Abstract

In computer vision and graphics as well as in mobile robotics one is often interested in capturing 3D real world scenes. In the former, models are captured for such purposes as photorealistic rendering. In the latter, models are captured to generate navigation maps for robot control. The objective in both fields is similar: How can the modeling of existing scenes be achieved?

This thesis investigates alternative ways of capturing the geometry and appearance of an indoor environment by using image-based modeling. Calibrated and uncalibrated methods are investigated and contrasted. The resulting models are validated by utilizing them in two robotics tasks. We first use the model as a navigation map to localize the pose of a robot and to track its motion based on images from a camera fitted on the robot. Second, we generate synthetic images by reprojecting and texturing the captured model given a desired camera pose.

The first approach is based on a panoramic image mosaic augmented with depth information and is built using calibrated cameras and range sensors (a trinocular device and a laser range-finder). Several methods for registering camera and range sensors were developed and compared. The model is segmented into planar pieces that can be reprojected in new positions.

The second approach uses an uncalibrated camera that samples a scene. By maintaining visual tracking of corresponding feature points, the geometry of the scene is reconstructed using stratified structure from motion. The geometric model is then bundle-adjusted and reprojected into the original images to acquire surface appearance. Surface appearance is represented not using a single traditional texture, but by acquiring a basis that captures the view dependency of the surface.

Contents

1 Introduction **1**
1.1 Panorama with Depth . . . 2
1.2 Geometric Model with Dynamic Texture . . . 3
1.3 Contributions . . . 4
1.4 Organization of the Thesis . . . 5

2 Image-Based Modeling and Rendering Techniques **6**
2.1 Image and View Morphing . . . 8
2.2 Plenoptic Sampling . . . 9
2.2.1 Image Mosaics . . . 9
2.2.2 Concentric Mosaics . . . 11
2.2.3 Lightfield and Lumigraph . . . 11
2.3 Depth-Based Reprojection . . . 13
2.3.1 Reprojection through Epipolar Constraints . . . 14
2.3.2 3D Image Warping . . . 14
2.3.3 Layered Depth Images . . . 14
2.3.4 Image-Based Objects . . . 15
2.3.5 Relief Textures . . . 15
2.4 Hybrid Geometric and Image-Based Models . . . 16
2.4.1 Volumetric Models from Images . . . 16
2.4.2 Polygonal Models using Structure from Motion . . . 18
2.5 Summary and Comparison . . . 19

3 Mapping in Mobile Robotics **21**
3.1 The Problem of Mapping in Mobile Robotics . . . 23
3.1.1 Aspects and Challenges . . . 23
3.2 Geometric Maps . . . 24
3.2.1 Occupancy Maps . . . 24
3.2.2 Topological maps . . . 25
3.2.3 Simultaneous Localization and Map Building . . . 26
3.2.4 Other Geometric Maps . . . 27
3.3 Image-Based Maps . . . 29
3.4 Summary of Mapping Algorithms . . . 31

4 Panorama with Depth **32**
4.1 Introduction to Computer Vision 33
4.1.1 Projective Camera 33
4.1.2 Camera Calibration 35
4.1.3 Image Projection of Geometric Primitives 35
4.1.4 Two View Geometry 37
4.2 Cylindrical Panoramic Mosaic 38
4.2.1 Depth Data 40
4.3 Depth from Stereo 40
4.3.1 Depth Panorama 40
4.3.2 Planar Patches 41
4.4 Depth from Laser Range-Finder 48
4.4.1 Range and Intensity Image Acquisition 48
4.4.2 Range-Intensity Image Registration 50
4.4.3 Robust Plane Fitting 52
4.4.4 Model Vertical Lines 54
4.4.5 The Navigation Image-Based Map 56

5 Applications of the Panoramic Model in Mobile Robotics **57**
5.1 Global Localization with Two Calibrated Panoramic Mosaics 58
5.1.1 Calibration of the Two Cylindrical Models 59
5.1.2 Localization Algorithm 59
5.1.3 Experiments with Synthetic Data 61
5.1.4 Experiments with Real Data 61
5.2 Global Localization using Stereo Panorama 63
5.2.1 Feature Selection 63
5.2.2 Planar Patch Features 64
5.2.3 Vertical Line Features 65
5.2.4 Point Features 66
5.2.5 Matching Planes and Lines 67
5.2.6 Experimental Results 68
5.3 Incremental Localization with Depth Enhanced Panorama 69
5.3.1 Angle Calibration and Matching Vertical Lines 71
5.3.2 Refinement of Matches in Image Space 71
5.3.3 Localization using Vertical Line Segments 73
5.3.4 Experimental Results 73
5.4 Mapping and Predictive Display for Remote Robot Environment . . 76
5.4.1 Predicted Scene Rendering 78
5.4.2 Experimental Results 80

6 Geometric Model with Dynamic Texture from Uncalibrated Image Sequences 83
6.1 Multi-View Geometry . . . 84
6.1.1 Non-Euclidean Camera Models . . . 84
6.1.2 Computation with Camera Models . . . 87
6.2 Affine Metric Structure . . . 90
6.2.1 Recovering the Affine Structure . . . 91
6.2.2 Recovering the Metric Structure . . . 92
6.3 Euclidean Structure for Perspective Projection . . . 93
6.3.1 Projective Factorization . . . 94
6.3.2 Bundle Adjustment . . . 95
6.3.3 Projective Ambiguity . . . 95
6.3.4 Self Calibration . . . 95
6.4 Dynamic Textures . . . 97
6.4.1 Texture Basis . . . 98
6.4.2 Geometric Texture Variation . . . 99
6.4.3 Photometric Variation . . . 102
6.4.4 Estimating Composite Variability . . . 102
6.4.5 Interpretation of the Variability Basis . . . 103
6.5 Combining Geometric Model and Dynamic Textures for Image-Based Rendering . . . 104

7 Dynamic Texture Model Implementation and Applications 108
7.1 3D SSD Tracking . . . 109
7.1.1 General Tracking Problem formulation . . . 110
7.1.2 Practically Useful Motion Models . . . 113
7.1.3 Experimental Results . . . 116
7.1.4 Discussion on the Tracking Algorithm . . . 118
7.2 IBR System Implementation . . . 120
7.2.1 Video Capture and Tracking . . . 120
7.2.2 Structure and Texture Editor . . . 121
7.2.3 Real-time Texture Blending . . . 124
7.3 Model Evaluation and Examples . . . 125
7.3.1 Geometric Model Accuracy . . . 126
7.3.2 Dynamic Texture Rendering Examples . . . 128
7.4 Tracking and Predictive Display for a Indoor Robot Environment . . . 134
7.4.1 Experimental Results . . . 134

8 Conclusions and Future Work 138
8.1 Image-Based Models . . . 139
8.2 Applications in Mobile Robotics . . . 141
8.3 Future Work . . . 142

List of Figures

2.1 Plenoptic function . . . 7
2.2 Image coordinates interpolation can produce incorrect views . . . 8
2.3 An example of a strip panorama - Peleg&Herman '97 [121] . . . 10
2.4 Concentric mosaics . . . 12
2.5 Lumigraph and lightfield parametrization . . . 12
2.6 (a) Setup used for lumigraph capturing (b) Example of a rendered view using lumigraph - Gortler *et al.* '96 - [55] . . . 13
2.7 (a) One of the input images of a scene(b) Different layers for the scene . . . 15
2.8 (a) Samples views of an object (b) Registered samples reprojected onto perpendicular image planes(c) Examples of rendered new views of the object -M. Oliveira '99 [116] . . . 16
2.9 (a) Image texture and depth map. (b) Pre-warped relief texture and final textured and rendered house. - M. Oliveira *et al.* [117] . . . 17
2.10 Illustration of the (a) visual hull and (b) photo hull recovered from 4 cameras (c) example of a rendered view from a volumetric model - Kutulakos&Seitz '2000 [86] . . . 17
2.11 Main steps in creating the Façade model - Debevec *et al.* '96 [35] . . . 19
2.12 Example of uncalibrated model developed by Pollefeys and van Gool [125] (a) geometric model and camera positions (b) view of the textured model . . . 20

3.1 Example of occupancy map (white - free space, black - obstacle; blue - unexplored space) [152] . . . 24
3.2 Example of map build with SLAM. The figure also shows the route that the robot followed while building. [52] . . . 26
3.3 (a) Example of sift features (b) View of the SIFT database [134] . . . 28
3.4 View of the VRML 3D model generated using range finders and a panoramic camera [94] . . . 29
3.5 Panoramic route representation [175] . . . 29
3.6 (a) Original locations and recovered trajectory (b) Example of a panoramic image (cylindrical projection) [75] . . . 30

4.1 Pinhole camera model . . . 34

4.2 From retinal coordinates to image pixel coordinates 34
4.3 A 3D line can be represented as $(\mathbf{v}, \mathbf{d})$. The image projection is represented by $\mathbf{m}$ 36
4.4 Points on a 3D plane are in general transfered to the image plane through a 2D projective transformation (homography). 36
4.5 Epipolar geometry 37
4.6 Calibrated stereo geometry 38
4.7 Example of a disparity map computed using the Triclops vision system. (a) left intensity image (b) disparity map (c) Triclops device . 39
4.8 (a) Cylindrical panoramic model of a research laboratory; (b) Depth map: dark - close objects; whither - far objects; black - no depth value; (c) Extracted planar patches 42
4.9 Flow chart for planar patch extraction algorithm 43
4.10 Planar region segmentation (a) Original image; (b) Edge detection and linking; (c) Constraint Delaunay triangulation; (d) Region growing (e) Extracted trapezoidal regions; (f) Vertical planar regions; 44
4.11 Spherical coordinates for normal to the plane 46
4.12 Point cloud for a planar region before (a) and after (b) the Hough Transform 46
4.13 Results of a range data segmentation algorithm 48
4.14 Image with the planar patches rendered from a different point of view 48
4.15 System configuration: the laser-range-finder with the camera attached on top of it is mounted on a pan-tilt unit 49
4.16 (top) Spherical representation of the range data from an 180° scan after filtering (bottom) Corresponding 180° panoramic mosaic. . . . 50
4.17 The projection of a space point $\mathbf{P}$ in the cylindrical image (θ, h) and the panoramic mosaic (u, v). We approximate the laser-camera transformation with a translation ΔY and a rotation over y axis. . 51
4.18 (top) Rendered image using only the global mapping (bottom) Rendered image with local alignment. Notice how misalignment from the top edge of the left monitor is compensated. 52
4.19 Planar region segmentation: (a) Original image; (b) Edge detection and linking; (c) Constraint Delaunay triangulation; (d) Merged planar regions 53
4.20 Model vertical lines: (a) Vertical edge segments in the panoramic mosaic; (b) 3D lines rendered on top of the model; 55

5.1 Two panoramic models and the planar image to be localized, each sowing 180° of the view. 59
5.2 Matching lines in image-based models and planar image 59
5.3 Variation of localization error error in X and Z directions with the number of triplets of line features; (a) without noise, (b) noise level $w = 3$. 61

5.4 Two cylindrical panoramic images (each formed using 25 planar images sampling 180°) 62
5.5 Planar image from in between the two panoramic images with the line features superimposed 62
5.6 (a) Recovered positions from the 9 images. The first panoramic model is situated at $(X, Z) = (0, 0)$ and the second one at $(X, Z) = (300, 0)$; (b) Error for the recovered orientation. The X coordinates show the position of the planar image regarding the first panoramic model. 63
5.7 Localization using planar patches 64
5.8 Localization using vertical lines 65
5.9 Localization using points 66
5.10 Robot positions for localization experiments. 69
5.11 Overview of the localization algorithm. **t** denotes robot's translation and α robot's orientation: (a) Edge detection and linking (b) Compute best match and calibrate angle α based on Hausdorff distance (c) Refine matches and eliminate duplicates based on image information (d) Localization using model-image correspondence . . 70
5.12 Illustration of matching algorithm: (a) model lines, (b) corresponding matched line after angle calibration (Section 5.3.1) and matched lines after refinement step (Section 5.3.2). Note that in the refinement step pair number 5 was eliminated as being a duplicate and the others are better aligned with the original lines in panorama. Numbers indexes lines in the model and show the matched pairs. . . 72
5.13 Recovered position (top) and orientation (bottom) for the model-based localization algorithm and a SFM point based algorithm . . . 74
5.14 Recovered positions for the first leg of the trajectory (straight line) with/without refinement. Note the errors in recovered position without refinement when a bad match is present. 75
5.15 Sensitivity of the localization algorithm with respect to features position in image vs. 3D. 76
5.16 Overview of the navigation system with predictive display. The image-based panoramic model is acquired and processed off-line using a camera and a laser range finder (a). The on-line localization system matches lines features in the model with the ones extracted from the current view (b). The robot location is send to the remote site where a predictive view is immediately rendered (c). The remote user can control the robot using the image-based interface. 77
5.17 Tele-operator interface where the user can view a 2D floor map with robot position and the predictive robot view. The vector in the small left window shows robot orientation. 81
5.18 Examples of predictive views using different texturing 81

6.1 Affine camera approximation 86
6.2 Different types of affine projection. 87
6.3 A general structure from motion algorithm extracts the structure and camera poses from a set of tracked points. 90
6.4 Texture parallax between two views 101
6.5 Comparison between spatial derivatives $\frac{\partial T_w}{\partial x}$ and $\frac{\partial T_w}{\partial y}$ (left two texture patches) and two vectors of the estimated variability basis $[\mathbf{B}_1, \mathbf{B}_2]$ (right) for house pictures. 104
6.6 A sequence of training images $I_1 \cdots I_t$ is decomposed into geometric shape information and dynamic texture for a set of quadrilateral patches. The scene structure X and views P_t are determined from the projection of the structure using a structure-from-motion algorithm. The view-dependent texture is decomposed into its projection $\mathbf{y}$ on an estimated basis B. For a given desired position, a novel image is generated by warping new texture synthesized from the basis B on the projected structure. On the web site is a compiled demo rendering this flower and some captured movies 105

7.1 Overview of the 2D-3D tracking system. In standard SSD tracking 2D surface patches are related through a warp W between frames. In our system a 3D model is estimated (from video alone), and a global 3D pose change ΔP is computed, and used to enforce a consistent update of all the surface warps. 111
7.2 **Top** Tracking individual patches using a homography. Not all regions can be tracked thought the whole sequence and occlusion is not handled. **Bottom** Through the 3D model each region motion is rigidly related to the model, and tracking succeeds through the whole sequence. The model also allows detection and removal of occluded regions and introduction of new regions. 116
7.3 **Top** Translation tracking of individual regions. Though the video sequence many patches are lost. **Middle** A projective 3D model is used to relate the regions, and provide more stable tracking through the whole sequence. **Bottom** An Euclidean model relates the region, and also allow the introduction of new regions. 117
7.4 Accuracy experiment. **Left** Scene image. Right current image and template superimposed indicate accurate alignment 118
7.5 Video capture and tracking user interface 121
7.6 Structure editor mode 122
7.7 The triangulation and texture editor 123
7.8 Synthetic room data and camera positions. Red dots indicate the camera positions and green lines indicate viewing direction 126

7.9 Reconstructed affine (top) and projective upgraded to Euclidean (bottom) structure under varying focal length. f = 50 (left), f = 100 (middle), f = 200 (right) 127
7.10 Accuracy of reconstructed affine and projective Euclidean structure as dependent of noise level with 10% outliers (top) and number of outliers with a noise level of 2 pixels (bottom). 127
7.11 Illustration of recovered structure from real data. (top) two of the original images overlapped with a Delaunay triangulation of the projected structure. (middle) recovered affine structure (same triangulation); (bottom) recovered projective Euclidean structure. . . 129
7.12 Left: Delauney triangulation of a captured house model. Note that triangles don't correspond well to physical planes. Right: Static texturing of a captured house produces significant errors. Especially note the deformations where the big and small house join due to several triangles spanning points on both houses. 130
7.13 Rendered novel views of a house by modulating a texture onto a coarse captured geometric model. Note the absence of geometric distortions compared to the previous figure. 130
7.14 Flower sequence: (left most) One of the original images and the outline of the quadrilateral patches 131
7.15 Examples of renderings for dinning room model from Figure 7.11. (top) example renderings of the affine model (bottom) example renderings of the projective model 132
7.16 Texturing a rotating quadrilateral with a wreath. Top: by warping a flat texture image. Bottom: by modulating the texture basis B and generating a continuously varying texture which is then warped onto the same quad. Demo on web site 133
7.17 Pixel intensity error when texturing from a close sample view (red) and by modulating the texture basis. For most views the texture basis gives a lower error. Only when the rendered view has the same pose as the one of the three source texture images (hence the IBR is a unity transform) is the standard view based texturing better . . 133
7.18 Overview of the tracking with predictive display system. 135
7.19 Recovered positions for the straight line trajectory (left) and the 2 perpendicular lines trajectory (left). The red line are the fitted 3D lines to each line segment. 136
7.20 Examples of predictive views (top row) and the corresponding actual images (bottom row) 137

List of Tables

5.1 Error for recovered position and orientation 69

5.2 Localization accuracy for the localization algorithm (with/without match refinement) and a point based SFM algorithm: error in distance between consecutive positions δ(cm) (ground truth 10 cm) and deviation from straight line ρ(cm) 75

7.1 Pixel reprojection error (pixels) and recovered motion range (degrees for rotation and original structure units for translation). The affine SFM algorithm recovers only an image plane translation that cannot be compared with the original 3D translation. 128

7.2 Average pixel jitter 132

Chapter 1

Introduction

Traditional computer graphics uses *geometry-based modeling and rendering* to create detailed 3D models of real environments. The model is either created manually using a modeler or generated from real 3D scene data obtained using range sensors [49]. Conventional 3D models are represented using polygonal, bicubic parametric curves, constructive solid geometry, or space subdivision (such as octrees). New views are rendered by reprojecting this model. Since the geometry and the reflection of real world objects is complex, efficient systems and realistic images are difficult to obtain using traditional geometry-based modeling and rendering.

A very similar problem exists in the field of *mobile robotics* where the navigation problem has been traditionally cast as acquiring a representation of the robot space in Euclidean coordinates. Achieving photorealism during navigation is not an issue, but the map is automatically generated and requires precise and detailed information that is used to track the robot's position. A navigation map can be described in various ways. The two extreme methods are a detailed geometric model and a scene graph representing the connectivity among topological elements in the environment. Classic vision-based navigation systems rely on the existence of a geometric model that contain precise metric measurements of the objects in the environment. Without such a priori knowledge it is in general difficult to create a metric map that contains a level of detail sufficient for robot localization.

Approaches to geometric modeling have been recently shown in *image-based modeling and rendering systems*. These systems combine computer vision and computer graphics algorithms to create a model directly from images and use this representation to generate new photorealistic views from viewpoints other than the original images.

We investigate in this thesis the applicability of image-based modeling techniques in mobile robotics. Our approach combines a sparse geometric model with image information to generate image-based models that are detailed enough to provide precise robot localization and that are easy to acquire and to generate. We developed two types of models: The first model (model 1) is a panoramic image-based model augmented with depth information built using *calibrated* computer vision techniques. The second model (model 2) is a sparse reconstruction of a geometric model that uses a special view-dependent texture that can be generated from *uncalibrated* video. The two models are described in more detail in the next sections.

1.1 Panorama with Depth

The first model is based on a panoramic mosaic augmented with range information. The mosaic is acquired by rotating a calibrated camera around the optical center, projecting then blending the images onto a cylindrical surface to obtain a continuous omnidirectional representation.

If no parallax is provided, the mosaic allows rerendering from only the same center of projection. To overcome this problem, we added depth information provided either by a trinocular vision system or by a laser range finder. In the latter case the intensity and depth information are generated from different sensors (camera and laser) so the data have to be registered in a common reference frame. We found that for moderate-precision tasks such as robot navigation, an image-based approach can be adopted for data registration [30]. Rather than determine the 3D rigid transformation between the sensors, which involves sensor calibration with respect to a common coordinate frame, we determine a 2D–2D image warp that linearly approximates the transformation between the generated panoramic mosaic and a similar panoramic representation of the laser range data. The registration is further refined by fitting local splines to a set of triangulated control points. The registered model is then segmented into planar pieces, which can be reprojected in any position.

The other type of range data provided by the trinocular vision system is inaccurate and noisy due to the small baseline of 10 cm relative to the size of the room (7–10 m). We developed an algorithm that combines intensity and depth data to obtain accurate patch segmentation [29].

To demonstrate the applicability of the proposed model for robot navigation, we developed localization algorithms that match an image taken from the current robot location with the model to compute the metric coordinates of the robot position. The first localization algorithm uses corresponding vertical lines in two panoramic mosaics for absolute robot localization [26]. The second uses the panoramic model with depth data from the trinocular device to calculate the absolute location of the robot by matching planar patches extracted from the current image with the ones from the model [28]. The accuracy of the localization was improved when global constraints were introduced for the match [21]. A similar approach was adopted for an incremental localization algorithm that uses 3D lines extracted from the registered panoramic model with laser range data [31]. Whereas the first algorithm does not consider the data association problem, the following two algorithms consider the image information contained in the model to provide a robust way for feature matching.

The delay in video transmission is a common problem in tele-robotics since it is essential to provide realtime feedback from the remote scene to the human operator. We use the final panoramic model (with depth from a laser rangefinder) for a system that integrates the incremental localization algorithm with predictive display. This system provides immediate synthesized video feedback representing the view of the command issued before the arrival of real video [72, 71].

1.2 Geometric Model with Dynamic Texture

For the second model a single uncalibrated camera is moved about a scene. Real time tracking is used to establish point correspondences between views and an affine or projective model is built and then upgraded to a metric model using the stratified approach. The resulting geometric model is *bundle-adjusted*, which consists of optimizing the accuracy over all views. The model is subsequently reprojected into the original images to acquire surface appearance. Surface appearance is represented not by using a single traditional texture, but by acquiring a *dynamic texture*, which is a basis that captures the view dependency of the surface [22, 25]. We show both mathematically and experimentally how this new type of texture model compensates for geometric inaccuracies, lighting, and nonplanarities. To render new poses the correct texture is modulated from the texture basis and then warped back to the projected geometry.

The geometric model is incorporated into a region-based tracking algorithm that relates spatial and temporal image derivatives to update the 3D camera position (full rotation and translation). The geometry of the scene is first estimated from uncalibrated video and then used for tracking. The tracking algorithm was integrated with a predictive display system. That system uses the ability of the dynamic texture model to be reprojected from any position in order to generate immediate visual feedback for a remote operator.

The dynamic texture model is also used for other augmented reality applications [24, 23], which are not included in this thesis since they are not directly related to robotics applications.

1.3 Contributions

The main contributions of this thesis are:

1. We contrast calibrated and uncalibrated methods for building models of scene geometry and appearance from images. To generate the image-based models we develop:

- A planar patch segmentation algorithm that integrates both depth and intensity data provided by a trinocular stereo system. The noisy depth data returned by the trinocular device cannot be used directly in the segmentation algorithm, but is combined with intensity information for a robust planar patch segmentation (for model 1).

- A new image-based algorithm for integrating range and intensity information from separate sensors. We performed a comparative study of performance for traditional geometric registration techniques and image-based registration techniques. This study demonstrated the advantages of the image-based registration in the field of mobile robotics (for model 1).

- A new type of view-dependent texture - dynamic texture - that represents surface appearance not using a single traditional texture, but rather by modulating a basis which captures the view dependency of the surface (for model 2).

2. We demonstrate the use of the models as navigation maps in mobile robotics with specific applications in localization, tracking, and predictive display.

- We develop localization algorithms that use the first image-based model and a single camera image to compute the position of the robot. The rich information contained in the image data is used for solving the data association problem (application of model 1).

- We develop a tracking and localization algorithm by extending the 2D image patch-based real-time tracking algorithms to 3D model-based tracking. In our method a 3D geometric model is both estimated from uncalibrated video and used to track rotational and translational camera scene motion from image differences (application of model 2).

- We integrate the models' ability to generate novel views from any position with the developed localization and tracking algorithms in a predictive display application for tele-robotics, where synthesized immediate visual feedback replaces the delayed real video from a remote scene (application of model 1 and 2).

1.4 Organization of the Thesis

The thesis starts by presenting the current state of the art in the two related research areas - image-based modeling in Chapter 2 and the problem of mapping in mobile robotics in Chapter 3. Chapter 4 presents the panoramic model with depth from stereo or laser rangefinder and Chapter 5 the model's applications in robot localization and predictive display. Chapter 6 describes the methods for obtaining geometric models from uncalibrated images and introduces the theory of dynamic textures. Chapter 7 evaluates model performance and presents applications in tracking and predictive display of the second model. Finally, we present our conclusions, possible improvements, and new directions in this research.

Chapter 2

Image-Based Modeling and Rendering Techniques

Image-based rendering systems combine computer vision and computer graphics algorithms to create a model directly from images and use this representation to generate photorealistic novel views from viewpoints different from the original ones. Image-based modeling and rendering (IBMR) encompasses a wide range of theories and techniques. The field is still young, and giving it a precise definition at this point would be premature. However, a common characteristic of IBMR methods is that images play a central role. While traditionally computer graphics has focused on transforming 3D data into 2D image projections, image based rendering (IBR) techniques shifts this emphasis to 2D - 2D image transforms. Using image-based approaches the model is very easy to acquire, and the cost of rendering is independent on scene complexity, because the sample images contain the complexity of the scene. The realism of the rendered images is directly dependent on the input images, hence it is possible to produce photorealistic renderings. A significant problem with current image-based rendering systems is how to accommodate changes in the scene (new objects, changes in lighting). Another problem is that their internal representation requires a lot of memory because of data redundancy, but considering the evolution of computer systems, this is not so challenging in our days.

This chapter will present some of the image-based modeling systems that are characteristic at the current state of the art. The idea behind several image-based rendering (IBR) systems is that they try to sample the *plenoptic function* [106] for the desired scene. The plenoptic function represents *the intensity of the light rays passing through the camera center at every location, at every possible viewing angle* (see figure 2.1). In general this is a 5D function, but depending on scene constraints it can have fewer degrees of freedom.

Based on the modality of encoding the plenoptic function and pixel transfer, IBR techniques can be classified into four categories: *image and view morphing*, *plenoptic sampling*, *depth based reprojection* and *hybrid methods*. The following sections will briefly describe each of these categories.

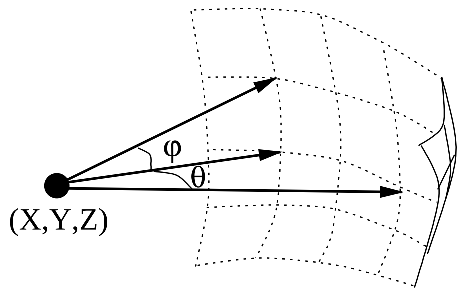


Figure 2.1: Plenoptic function

2.1 Image and View Morphing

Image morphing techniques generate intermediate views by interpolation in image coordinates, without considering the 3D geometry of a scene. Hence synthesized views are typically not geometrically correct (see figure 2.2). A well known technique is Beier and Neely's feature based image metamorphosis [9]. They used corresponding line segments to smoothly transform a source image into a destination image using simple pixel interpolation. A more advanced technique, called view morphing, was developed by Seitz and Dyer [135]. Assuming that the motion of the camera is restricted to a line that connects the two centers of projection, they generate new views of a scene that represent a physically-correct transition between two reference views. The intermediate views are created by first prewarping the images in order to align the projection planes, then morphing using interpolation, and then postwarping the interpolated images. Manning and Dyer extend this idea by creating dynamic view morphing [98]. Lhuiller and Quan [91] describe a view morphing technique which also produces interpolated views that correctly reproduce the motion of salient points in the scene. They describe a scheme for triangulating the input images in such a way as to respect intensity discontinuities. A similar approach, called view interpolation was created by Chen and Williams [19]. Arbitrary viewpoints, but with constraints on the viewing angle are generated by interpolating images stored at nearby viewpoints. Their technique requires calibration and a full correspondence map between the sample images. This is very difficult to obtain from real images, so their algorithm was tested with synthetic images.

Image morphing techniques are very fast and simple, but they produce nonrealistic and geometrically incorrect renderings approximated from the sample images. View morphing techniques demonstrate that it is possible to produce correct interpolated images from a set of correspondences, but, except in Chen's method, the camera motion is limited to a lines connecting the camera centers.

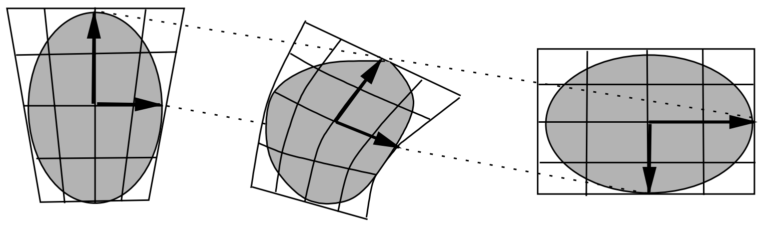


Figure 2.2: Image coordinates interpolation can produce incorrect views

2.2 Plenoptic Sampling

In this class of techniques, a subsample of the plenoptic function is captured from a set of input images. New views are generated by interpolating the appropriate rays from the plenotic function. One of the first techniques in this category are *image mosaics* that sample a 2D plenoptic function under some particular viewpoint or scene constraints. The next subsections will first present some mosaicking techniques and then the more general plenoptic sampling techniques - lightfield and lumigraph.

2.2.1 Image Mosaics

One of the simplest image-based modeling techniques is image mosaicking. The term "mosaic" refers to the combination of at least two images to yield a higher resolution or larger image. This is a very old technique and was developed long before the age of digital computers. It appeared shortly after the photography was invented in 19*th* century, when images acquired from balloons or tops of the mountains were manually pieced together to create maps. Today mosaics are used in many applications like white-board and document scanning, approximation of 3D scenes [147, 148, 141], video compression [68], architectural walkthroughs, virtual museums, cartoons [169], telepresence, tele-medicine, and astronomy.

In order to register the images into a larger unified image, they are related by a linear projective transformation (homography). This is possible when the images either sample a planar surface, or are taken from the same viewpoint. The general form of a projective transformation is:

$$s \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{H}_{13} \\ \mathbf{H}_{21} & \mathbf{H}_{22} & \mathbf{H}_{23} \\ \mathbf{H}_{31} & \mathbf{H}_{32} & \mathbf{H}_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

where (u, v) and (u', v') are corresponding pixels in two images, s is a scale factor and $\mathbf{H}$ is a non-singular matrix (defined up to a scale factor).

Creating an image mosaic involves three subproblems: **projecting** the images on the desired surface (planar, cylindrical, spherical), and correcting geometric deformations caused by different types of lenses, **registering** the images into a common coordinate system, and **correcting** resulting small errors from the registration process e.g. by smoothing.

There are different types of image projection depending on the desired application and acquisition technique. The simplest set of images to mosaic are pieces of a *planar* scene such as a document or a white-board. As mentioned before, these pieces are related by a linear projective transformation, so they can be registered together into a larger planar image [147]. While planar mosaics are convenient representation for relatively small field of view (less 180°), they become problematic for wider scenes. In those circumstances either *cylindrical* or *spherical* representations are more suitable giving panoramic or panospheric mosaics. Cylindrical panoramic mosaics are created by projecting images taken from the same viewpoint but with different viewing angles onto a cylindrical surface [141, 148]. They can be used for fixed location visualization (Quick Time VR) [18] or variable location visualization by interpolating several panoramas (plenoptic modeling) [106] and for recovering 3D structure of the environment [80]. Spherical panoramic mosaics can be created either by projected planar images taken from a fixed center of projection onto a sphere [147], or using special lenses and mirrors [114].

Images with parallax, which do not satisfy any of the two conditions mentioned above, can also be composed into a mosaic. Irani *et al* [68] used a polynomial transformation with more than eight degrees of freedom to compensate for nonlinearities due to parallax. An interesting approach is presented by Dornaika *et al.* [40] where an intermediate image is used for registering two images under arbitrary camera motion. Another solution is to use a one dimensional camera to scan scenes. This can be realized using conventional cameras by combining strips taken from a sequence of neighboring images. In this way Peleg [121] and Tsuji [175] created a panoramic mosaic from images along an arbitrary path. Figure 2.3 shows an example of a strip panorama.

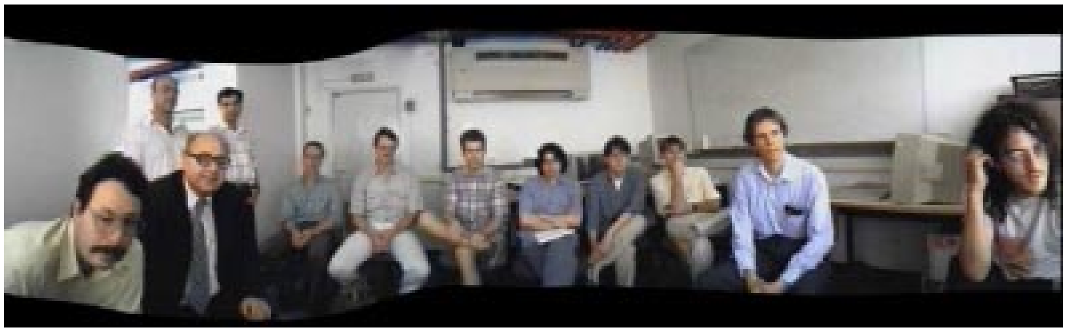


Figure 2.3: An example of a strip panorama - Peleg&Herman '97 [121]

In order to create the image mosaics, images have to be registered or matched based on aligning overlapping areas. Carefully calibrated cameras prior to the acquisition process can eliminate this step, but this is cumbersome and inconvenient for the user. The image registration techniques used in the literature include:

- **Manual registration** methods where the operator has to manually align the images.
- **Feature based methods** that manually or automatically detect specific features in the images, compute correspondences, and then estimate the camera motion.
- Finding the motion that will best align the images by **exhaustively searching** all the possible motions. This can be computationally extremely expen-

sive. Another possibility is to **iteratively adjust** the motion parameters by minimizing the differences between the overlapping areas. This method can lead to a local minimum unless a reliable initial estimate is provided.

- **Frequency domain** techniques compute the image displacement from phase correlation. These methods require significant overlap between the images.

After image alignment, usually the mosaic has to be further processed in order to eliminate remaining distortions and discontinuities where input images overlap. These errors are caused by changes in the illumination, imperfect registration, dynamic scenes, etc. The lighting problem can be reduced using histogram equalization or locally smoothing the mosaic at the intersection lines [148]. For compensating small errors introduced by motion parallax Szeliski and Shum [148, 141] developed a local alignment (*deghosting*) technique which warps each image based on the results of pairwise local registration.

Image mosaics are easy to build, but the rendered images must satisfy the same constraints as the input images, so it is not possible to create arbitrary new views. Adding more information like depth or disparity can overcome this problem.

2.2.2 Concentric Mosaics

Concentric mosaics are a generalization of cylindrical panoramas that allow the viewer to explore a circular region and experience horizontal parallax [140]. Concentric mosaics are created by composing slit images taken at different locations along a circular path. Thus a cylindrical panorama is equivalent to a single mosaic for which the axis of rotation passes through the camera's center of projection. Novel views are rendered by combining the appropriate captured rays (see figure 2.4). This procedure assumes that all images are infinitely far away. Since this assumption is not satisfied in most cases, the algorithm can introduce vertical distortion in the rendered images. In practice, dense concentric samples are obtained using a regular camera moved along a circular path. The camera can be oriented either tangent or normal to the path and multiple concentric mosaics are created through resampling. A similar approach is described in [120] where a *stereo panorama* is created using a camera with one left and one right slit that is rotating along a circle.

2.2.3 Lightfield and Lumigraph

This class of methods first build up a lookup table of light rays by taking many image samples of an object or a scene and then reconstruct images from arbitrary viewpoints by interpolation in the stored table. This lookup table is an approximation of the plenoptic function. A significant advantage of these methods is that pixel correspondence is not necessary and rendering is fast, but they require

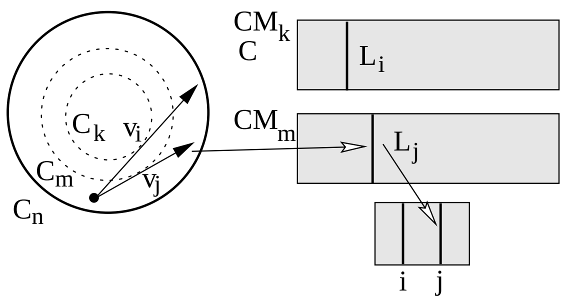


Figure 2.4: Concentric mosaics

extensive data acquisition, knowledge about the camera viewpoint during data acquisition, and high memory requirements. That is why they are more suitable for synthetic scenes.

Appearing simultaneously in '96 *light field rendering* [90] and the *lumigraph* [55] both use a 4D parameterization of the plenoptic function where the scene is restricted to a bounding box (see figure 2.5). In this representation, a ray is parametrized by its intersection with 2 planes, structure known as *light slab*. The interpolation scheme used by Levoy and Hanrahan [90] approximates the resampling process by interpolating the 4D function from nearest samples. Lumigraph [55] is reconstructed as a linear sum of the product between a basis function and the value at each grid point. Figure 2.6 shows the setup used for capturing a lumigraph type model from real images (a) and an example of rendering using the model (b).

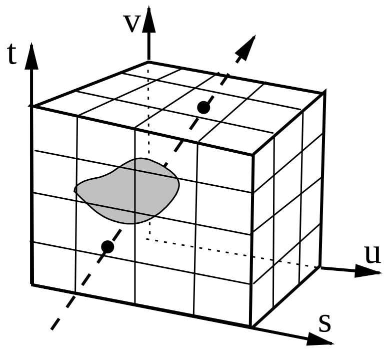


Figure 2.5: Lumigraph and lightfield parametrization

A generalization of the lumigraph is presented by Buehler *et al.* [14]. Their approach eliminates the lumigraph requirement of having the cameras on a regular grid by allowing unstructured set of cameras and using variable information about the scene geometry. The technique is similar to the view dependent texture mapping [35]. The same idea of acquiring dense images taken from a free-moving hand held camera for sampling plenoptic function is presented by Koch and Pollefeys in

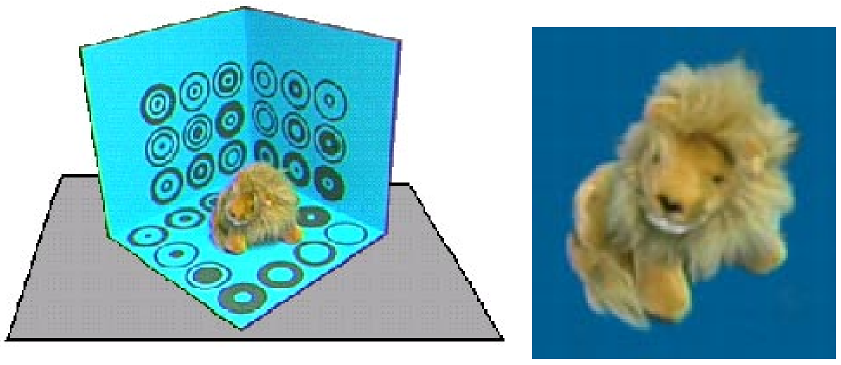

(a) (b)

Figure 2.6: (a) Setup used for lumigraph capturing (b) Example of a rendered view using lumigraph - Gortler *et al.* '96 - [55]

[81]. They use uncalibrated structure from motion algorithms for recovering camera positions. Novel views are generated by piecewise mapping and interpolating nearest viewpoints.

If a detailed geometry of the scene is known, *surface light fields* [168] can be efficiently used to encode the rays leaving every point on the known surface (lumispheres). A detailed geometry of the scene is acquired using laser scans which are registered with photographs. This technique is related to the methods in section 2.4 since it uses a known geometry of the scene, but the idea of sampling the plenoptic function belongs to this class of IBM.

By imaging an object from several viewpoints, the lumigraph and lightfield techniques can produce detailed and realistic models. However the large number of input images required to avoid blurring of synthesized views often make them unpractical compared to other IBR techniques.

2.3 Depth-Based Reprojection

This large class of IBR techniques use a relatively small number of images with depth or disparity values to reproject image pixels at a given camera viewpoint. The rendered images are geometrically correct, but an accurate disparity or depth map is very difficult to recover from images alone. Depth can be evaluated from stereo vision or using other sensors such as laser range-finders or sonars and then combined with intensity and color information provided by images to form an image-based model. Disparity is usually computed from image correspondence. Some researches include the depth-based techniques in the hybrid geometric and image-based methods (Section 2.4). Here, considering that there is no explicit geometric model of the scene reconstructed from the images, they are listed as a separate class.

The next subsections will briefly summarize some of the well known depth-based reprojection techniques, more precisely: Laveau and Faugeras' disparity based reprojection [87], 3D image warping [105], layered depth images (LDIs) [137], image-based objects [116], and relief textures [117].

2.3.1 Reprojection through Epipolar Constraints

One of the first IBR techniques using a collection of images with disparity values to generate new views was presented by Laveau and Faugeras [87]. They used fully calibrated images with disparity maps and epipolar constraints to compute a novel view through a process similar to raytracing.

2.3.2 3D Image Warping

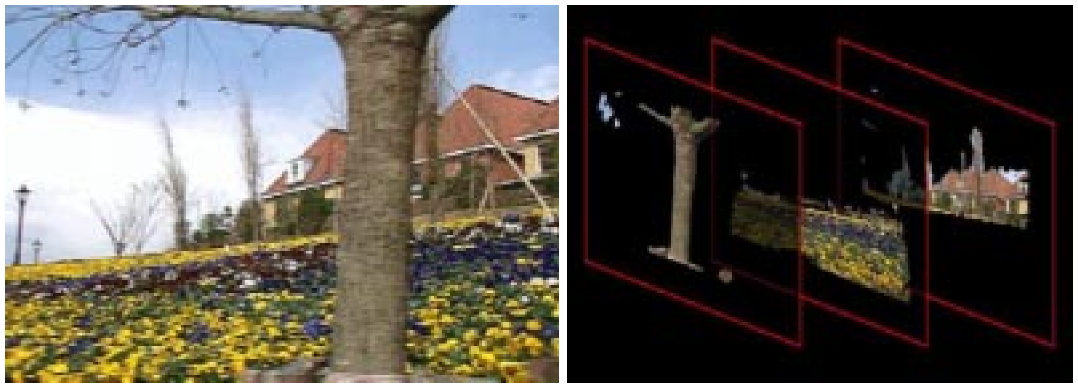
Three dimensional image warping [105] is a geometric transformation that correctly maps a source image with depth to an arbitrary target view. The 2D to 2D warping normally used in texturing models is only physically correct when model facets are true planes. If the texture has 3D structure, using a 2D image alone for texturing is at best an acceptable approximation, but in some cases (such as close up views or views at a grazing angle to the surface) it will give a flat, unnatural appearance to the rendered views. In 3D warping a depth map is used to correctly "rearrange" the texture image into a new view. The target image can be obtained by applying a planar projective transformation to the source image followed by a per-pixel shift proportional to the *generalized disparity* in the direction of the epipole of the target image (this corresponds to a factorization known as *plane-plus-parallax* in the computer vision literature). Normal texture mapping is a special case of 3D warping when all pixels in the source image have the same disparity values. The visibility problem is solved using a list-priority algorithm [105].

The 3D warping approach is generalized to cylindrical mosaics in plenoptic modeling [106].

2.3.3 Layered Depth Images

A *layer depth image* (LDI) [137] is a view of the scene from a single camera view point, where multiple pixel values are stored for each line of sight. In this case, each element of the image consists of an ordered list of samples. For rendering with LDIs, the 3D warping equation can be applied in occlusion compatible order. In this way, LDIs can reduce the occlusion artifacts while retaining the efficiency of the 3D warping equation. This property in combination with polygonal techniques is exploited in [104] to produce efficiently high resolution rendering of large environments. Figure 2.7 illustrates the idea of layer depth images.

The LDIs idea is further improved in [17] by introducing the LDI tree - a combination of a hierarchical space partition scheme with the concept of the LDI.



(a) (b)

Figure 2.7: (a) One of the input images of a scene(b) Different layers for the scene

The fixed resolution of the LDI may not provide an adequate sampling rate for every either reference (input) image or rendered video. The LDI tree preserves the sampling rate of the reference images by adaptively selecting an LDI in the tree.

2.3.4 Image-Based Objects

While image with depth and LDIs can be warped to produce correct reprojections of represented surfaces, to render all views of a 3D object, samples from several viewpoints are required. *Image-based objects* (IBOs) [116] provide a compact image representation of 3D objects. They are constructed by acquiring different views of an object that are registered and projected from a single center of projection onto the facets of a parallelepiped that represents a bounding box of the object (figure 2.8). Thus each IBO is represented by six LDIs that are warped in occlusion-compatible order.

2.3.5 Relief Textures

Relief textures [117] are an extension of conventional textures that supports the representation of 3D surface detail and view-motion parallax. This effect is achieved by pre-warping the relief textures and mapping the resulting images onto flat polygons. More intuitively (see Figure 2.9), using a rectangular geometry model of a house and texturing it with a flat 2D texture would give un-natural renderings for many views. However, using a texture composed of both an image and depth map, and relief-texturing the same geometry recreates the correct views of the house fine structure. In the figure particularly this is evident for the roof and dormers.

Relief textures can be used in the context of image-based objects for generating an object representation as six relief textures associated with the object bounding box. New views of the object can be obtained by pre-warping the relief textures and mapping the resulting images onto the facets of the box.

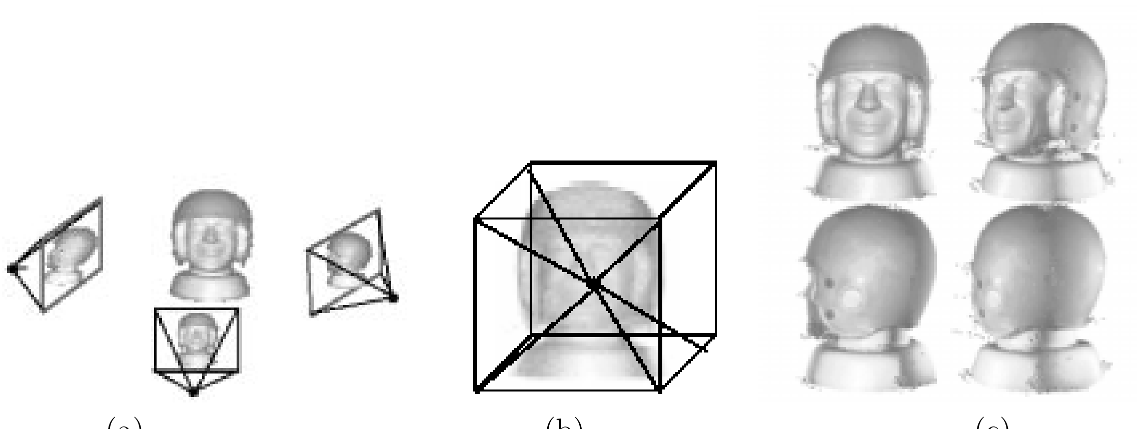


Figure 2.8: (a) Samples views of an object (b) Registered samples reprojected onto perpendicular image planes(c) Examples of rendered new views of the object -M. Oliveira '99 [116]

Depth-based reprojection techniques are popular among IBR methods because they produce high quality and correct renderings. While they are most practical for synthetic models, their practicality is still limited by the difficulty of acquiring depth maps for real scenes.

2.4 Hybrid Geometric and Image-Based Models

This last class of IBR techniques is closer to the traditional graphics modeling by having a geometric model as the based structure of the image-based model. The difference is that the geometric model is obtained and textured directly from images. Currently, there are two different groups of techniques - those that reconstruct a volumetric model from images and those that reconstruct architectural models using structure-from-motion (SFM) techniques (see Section 6.1). Next subsections will give examples in both categories.

2.4.1 Volumetric Models from Images

Virtualized reality. A volumetric model of an object or scene is a geometric representation of the volume occupied by the object. T. Kanade's *virtualized reality* [129] uses a collection of calibrated cameras in conjunction with multi-base line stereo techniques to compute a volumetric model of the scene. The model is first represented as a collection of 3D points that is converted by ISO surface extraction to a polygonal mesh that is textured from images. This model is applied to dynamic scenes.
Volume carving [136, 86] is a common method that uses silhouette edges to carve unoccupied regions from an explicit volumetric representation. The resulting shape

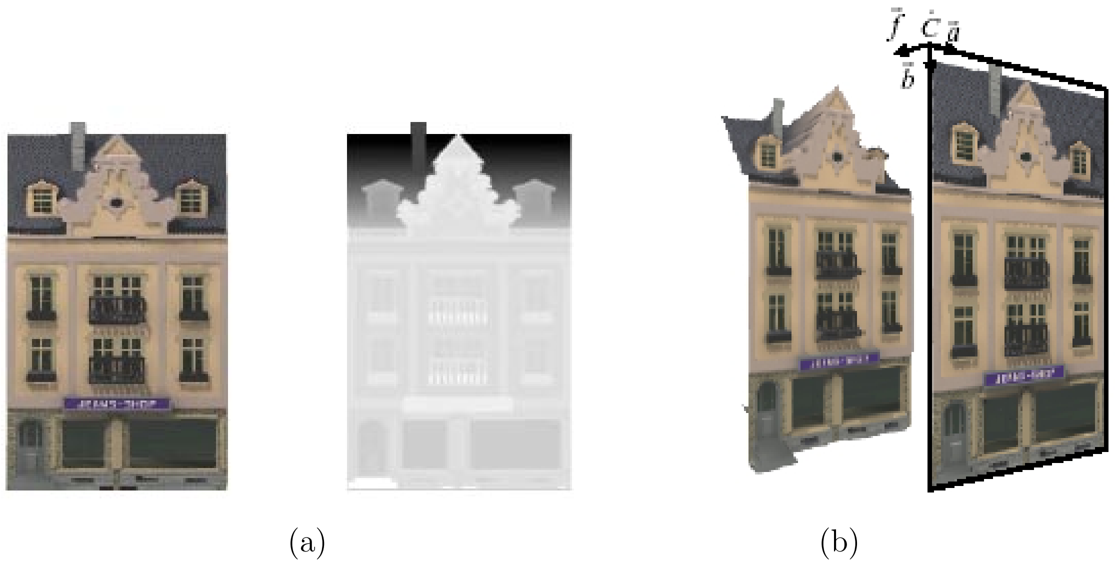


Figure 2.9: (a) Image texture and depth map. (b) Pre-warped relief texture and final textured and rendered house. - M. Oliveira *et al.* [117]

- called *visual hull* is a tighter fit than the object's convex hull that encloses the scene. All voxels falling outside the projected silhouette are eliminated from the volume. The *photo-hull* or *maximal photo-consistent shape* [86], contained inside the visual hull, is obtained using color consistency check on the visual hull. See Figure 2.10 for an illustration of the visual hull (a) and photo hull (b) concepts. The resulting volume is a quantized representation of the visual hull according to the given volumetric grid. Figure 2.10 (c) shows an example of a rendered view from a volumetric model.

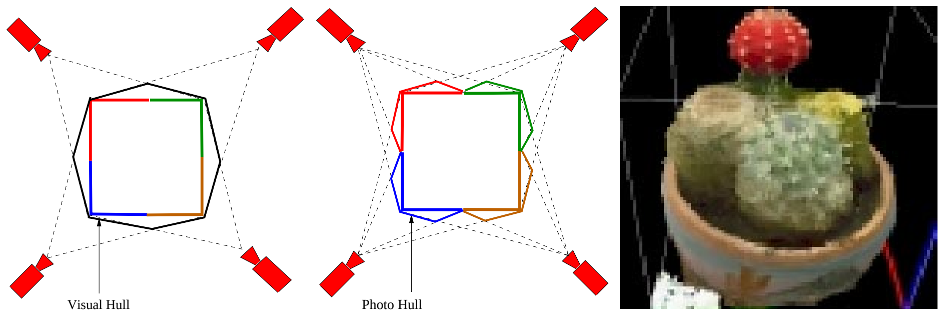


Figure 2.10: Illustration of the (a) visual hull and (b) photo hull recovered from 4 cameras (c) example of a rendered view from a volumetric model - Kutulakos&Seitz '2000 [86]

Image-based visual hulls. A more efficient representation of the visual hull [103] uses silhouette edges to compute the visual hull without constructing an auxiliary geometric or volumetric representation. The visual hull is stored as intervals along rays originating from each camera center for the sample data. The rendering algorithm takes advantages of epipolar geometry and McMillan's occlusion compatible order [105] to achieve real time performance.

2.4.2 Polygonal Models using Structure from Motion

A polygonal model of the scene can be reconstructed using images and structure from motion techniques (see section 6.1). A structure from motion algorithm takes as input a collection of corresponding image features (points, lines) and reconstructs their position in a 3D space. Depending of the level of calibration and camera model the resulting representation can be Euclidean (metric), projective or affine. The metric model is the most suitable for rendering with traditional graphics techniques. Most of the current systems focus on the more constrained problem of architectural reconstruction for which the abundance of planar surfaces, parallel and perpendicular lines make the reconstruction problem simpler and more robust.

A well known system - Façade [35] proposed a non-linear optimization algorithm to reconstruct 3D textured models from photographs. Their system is built from two components: the first is a photogrammetric modeling system that recovers the basic geometry of the scene and the second a model-based stereo algorithm that refines the geometric model to conform with its actual appearance from a set of photographs. For rendering they present a view-dependent texture-mapping that produces new images by warping and composing multiple views of the scene. The main steps in the algorithm are illustrated in figure 2.11. A similar system presented by Coorg and Teller [32] uses calibrated spherical mosaics to reconstruct a large set of 3D buildings. When dense range data is available from range-finders, it can be combined with the image data to form a detailed 3D model [144] of a large scene. The algorithm first segment the data into planar surfaces and then reconstruct buildings' edge segments as intersection of planes.

Recent advances in uncalibrated structure from motion [59] made the model acquisition more efficient and convenient for large-scale applications. Theoretically, using self-calibration constraints, the geometric model of the scene can be reconstructed from a set of uncalibrated images, e.g. taken with a hand held camera [93, 125, 8, 166]. The model is then textured from the sample images. Most of these systems are developed by European groups that pioneered the use of projective geometry in recovering the scene structure and camera motion. Most of the practical systems impose external constraints on the scene geometry (planarity, parallelism, perpendicularity) to make the self calibration more robust. Figure 2.12 shows an example of a reconstructed model and camera positions using uncalibrated structure from motion and a view of the textured model.

The hybrid geometric and image-based methods are very promising in practical applications. Beside being compact, their 3D geometry allows most traditional computer graphics processing (e.g. light, shadows). Recent development of SFM

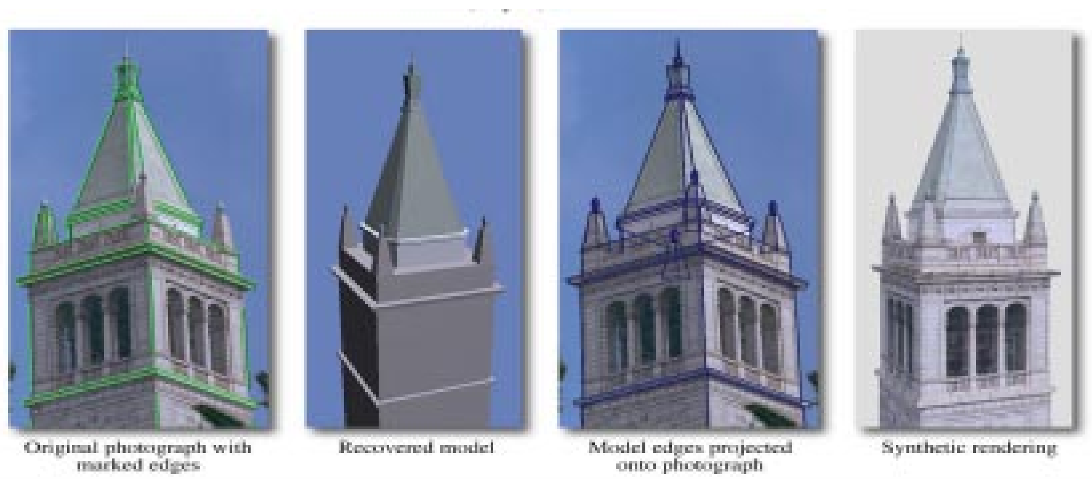
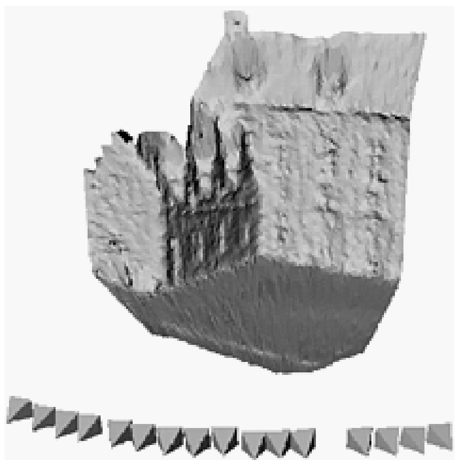

Original photograph with marked edges
Recovered model
Model edges projected onto photograph
Synthetic rendering

Figure 2.11: Main steps in creating the Façade model - Debevec *et al.* '96 [35]

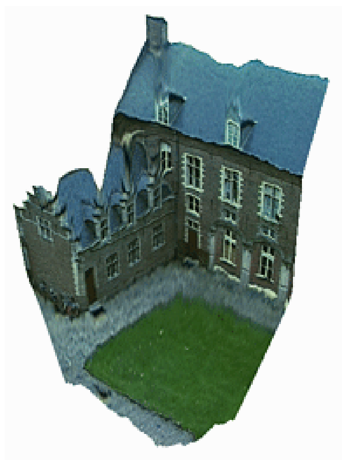
techniques made the geometric model acquisition easier by not requiring strong calibrated cameras. Despite having an approximate geometric model of the scene, the desired realism can be achieved by using an image-based texture. Our second, uncalibrated model (see Section 6) offers a very efficient and innovative example of a hybrid model.

2.5 Summary and Comparison

This chapter provided an overview of image-based rendering techniques. They are divided in four main groups depending on the way of representing and using the image data in modeling: image and view morphing, plenoptic sampling, depth-based reprojection and hybrid geometric and image-based methods. Given the current development of IBR techniques, the survey intended to provide an intuitive understanding of these methods focusing on their fundamental aspects. There are two very good surveys in the area that should be mentioned - one by Sing Bing Kang [78] and a more recent one by Manuel Oliveira [115]. Despite the existing challenges, image-based methods offer many advantages over traditional graphics modeling techniques in creating realistic models. An experimental study and comparison of non-euclidean image-based methods is presented in [20].



(a)



(b)

Figure 2.12: Example of uncalibrated model developed by Pollefeys and van Gool [125] (a) geometric model and camera positions (b) view of the textured model

Chapter 3

Mapping in Mobile Robotics

People have always wanted to build robots that can function in every day life and help without being helped too much. One of the most difficult problems is understanding the surrounding environment and freely navigating through it. Mapping in mobile robotics addresses the problem of acquiring a model of the physical environments through mobile robots. This problem is regarded as one of the most important tasks in autonomous navigation. Map building has been of considerable interest to humans over the last 4000 years. The basic process of distance measurement, correlation and triangulation was known by Phoenicians who built the first maps of the Mediterranean area. Today navigation is studied from different aspects and is applied in many areas like maritime, aviation, industrial or military applications. Despite the significant progress in robotic mapping in the last two decades, there are still many challenges.

The robot understands the environment through its sensors. There are different sensor technologies considered in the literature: odometry; ultrasonic, laser range and infrared sensors; vision sensors; radar, compasses and GPS. Considering vision as the main sensor for acquiring the model, the mapping problem becomes closely related to image-based modeling problem (discussed in the previous chapter). Being viewed by two different and distinct research communities, the solutions to these two related problems are quite different. The work presented in this thesis integrates two types of image-based maps in mobile robotic systems and analyzes the advantages of the image-based representation in different tasks - localization, tracking, and predictive display.

This chapters presents a short survey of existing robotic mapping techniques. Considering the amount of work in the area the goal is not to enumerate all the existing systems and techniques, but more to present the main existing approaches, differences and challenges in mapping indoor environments. Some very good surveys exist in the area - Thrun's very comprehensive presentation of mapping based on a probabilistic approach [151], a survey on vision based navigation approaches that includes both techniques for indoor and outdoor environments [37], and some older reviews presented by Boresnstein [11] or Cox [34].

From a mapping point of view, the most natural classification would consider the type of information that is represented. Most of the systems use a geometrical representation of the navigation space - *geometric-based maps* where the 2D or 3D physical space is mapped according to the absolute geometric relationships between objects. The level of detail in geometric maps varies - most of the systems use point features or occupancy grids while other use mode complex features (lines, polygons, 2D or 3D objects) or a complete CAD like model of the environment. The second category, which is seldom used but very related to the presented work, are *image-based maps*. In this case the map is simply a collection of images from known locations or along a path followed by the robot, that sample the environment without necessarily extracting an explicit geometric model. Some would consider another distinct category - *topological maps* that represent the environment based on topological relationships between features using a graph or a net. Because the topological approaches rely on geometric information, they can be seen as a higher resolution representation of a geometric map, and are mostly use for path-planning purposes. This survey starts with a short definition and description of the mapping problem and the aspects that have to be considered when designing a robotic mapping algorithm, followed by a summary of each of the above mentioned types of maps: geometric and image-based maps.

3.1 The Problem of Mapping in Mobile Robotics

Mapping involves acquiring a model of robot's environment that is usually used in navigation (*e.g.* localization). A map is usually formed by a collection of landmarks that are then identified by the robot in order to relate (localize) itself with respect to the map. By a closer look to the mapping problem, we see that, without an a priori map, the problems of mapping and localization are dependent on each other. Mapping is therefore like a *chicken and egg* problem: if the robot position can be tracked around the environment, building a map becomes simple; conversely, if the map is already known, robot position can be computed at any point in time [11]. The problem becomes more complex, if both localization and mapping are solved concurrently. This is refereed to in the literature as *simultaneous localization and map building* (SLAM) and is a central topic in mobile robotics. It will be later discussed in subsection 3.2.3. Next subsection presents most important problems in mapping as classified in [151].

3.1.1 Aspects and Challenges

Mapping and localization rely on sensor measurements. Any realistic navigation system has to account for *measurements noise*, caused by sensor limitations, uncertainties and inaccuracies. The most natural way to incorporate noise in the mapping process is to use probabilistic algorithms. Perceptual noise is complex to model and accumulates in time. Some of the common approaches for modeling robotics mapping (see subsection 3.2.3) are Kalman filters, expectation maximization or a combination of these two.

A second important and difficult problem in mapping, which is often ignored, is the *correspondence problem*, also refereed to as the *data association problem.* The correspondence problem involves relating current robot measurement with existing map features. The most robust systems address this problem using vision sensors. We believe that image information, which can easily describe complex aspects of the environment, is a very promising way of dealing with the data association problem (see chapter 4).

Another challenging problem is the fact that *environment can change* in time. The change can be cause by actual spatial changes in the environment (moving chairs, people), occlusions or by changing illumination. The latest type of change is only valid when working with vision sensors. Most of the systems rely on static objects and small changes are modeled by noise.

A last problem is *robot exploration*. When the map is unknown, or partially known, the robot has to be guided in order to continue its exploration. Most of the approaches are heuristic, but the majority of the systems do not address this problem.

Next sections will present the most important techniques in geometric based maps (occupancy, topological, SLAM and others) and image-based maps.

3.2 Geometric Maps

3.2.1 Occupancy Maps

Historically, the first maps were represented as *occupancy grids* - a 2D array of cells, each containing a value that represents the algorithm's certainty about the existence of obstacles in that cell (see Figure 3.1 for an example of an occupancy map). After the pioneering work of Giralt et al. [51] and later Moravec and Elfes [110, 111, 43] this approach was adopted by many researchers [12, 173, 172, 101]. Borestein and Koren [12] refine this approach by introducing histogram grid which allows dynamically modeling the data when the robot is moving.

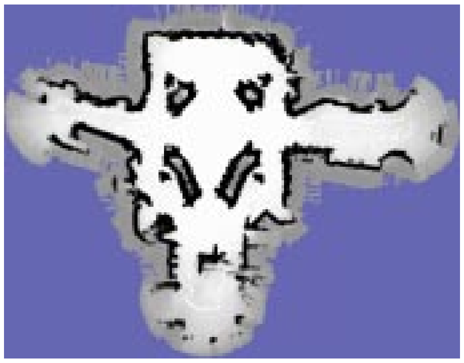


Figure 3.1: Example of occupancy map (white - free space, black - obstacle; blue - unexplored space) [152]

Most of the systems based on occupancy maps use sonar or laser sensors, but certainty values can also be extracted from vision sensors. Murray and Jennings [112] propose a trinocular stereo system to build an occupancy map from range values. Matthies *et al.* [118] build a 3D occupancy map using an omnidirectional stereo system. Thrun [152] combines sonar and vision sensors to create a grid-based occupancy map. The vision occupancy map is estimated from disparity values. In [164] an edge map and a color map extracted from an omnidirectional vision sensor are integrated with a sonar map to produce an exact occupancy map.

The central problem in generating occupancy maps is incorporating noisy and incomplete measurements, assuming known robot position. The algorithms are often based on one of the probabilistic approaches described in subsection 3.2.3.

3.2.2 Topological maps

An alternative type of map - *topological map* [100, 107, 83], represents the navigation space by a graph, where nodes correspond to key places of the environment (*e.g.* corners, doors) and arcs correspond to connectivity of moving from one place to another. The graph does not reflect the geometry of the space, but the relations between key places of the environment that can be used for path planning or navigation. That is why usually topological maps are enriched with local metric information [152] or image-base information [130] to facilitate localization or to establish correspondence between current and past locations. Sometimes the topological maps are built from the geometrical maps [152, 172], or from image-based maps [69].

Voronoi diagrams can also represent the connectivity of free space in a very compact way by storing a collection of points equidistant from two nearest obstacles. They can be used to generate optimal paths that are equidistant from obstacles. Canny and Donald [15] use simplified Voronoi diagrams to generate collision free paths among obstacles. In [69] and [130] image-based information is added to the graph in order to help the localization process. Ishiguro *et al.* [69] generate a topological map, called T-Net, that approximates local areas with straight paths which are memorized with pairs of feature points. The robot is moving along the T-Net by tracking feature points. Visual information, in the form of omnidirectional visual and range data and a panoramic route representation, is associated with the graph. In [130], the robot explores the space, stores the sequence of unfamiliar images, and associates them with an arc. It will close a loop when it finds a familiar image. The appearance of an unfamiliar image is marked as a new node in the graph and a new arc begins from this node.

Topological maps are mostly used for path-planning, so accuracy in robot position is not so important. Many topological maps rely more on external sensors (sonars, cameras, GPS) than on internal sensors (odometry). Without any additional information (metric, images), it is very hard to localize the robot. One solution is to use odometric information and to correct the estimated location after each node [84]. Another solution is to keep a short history of the measurements. If additional information is stored in each node, traditional feature-based approaches can be used for robot localization [152, 150, 130, 69]. When images are used to identify arcs or nodes in the graph, matching is done by traditional correlation techniques in image or transformed space (for example histogram matching [163]).

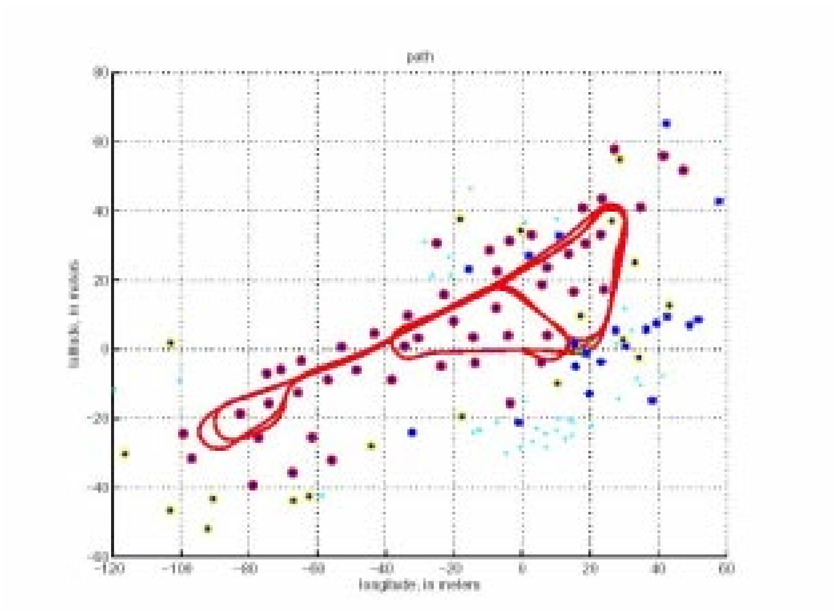


Figure 3.2: Example of map build with SLAM. The figure also shows the route that the robot followed while building. [52]

3.2.3 Simultaneous Localization and Map Building

The process of simultaneous tracking the position of the robot relative to its environment and, at the same time, building a map of the environment is referred in the literature as *SLAM* (*simultaneous localization and map building*) [38] or *concurrent mapping and localization* [153]. The stochastic approach was introduced in the 1990s with a series of seminal papers by Smith et al.[143] that put the basis of the statistical framework for formulating SLAM.

Most of the systems use sonar or laser range-finders to observe landmarks (significant point features) that will form the map. Figure 3.2 shows an example of a map and the path that was followed while building. Some vision-based systems use low level features such as vertical edges [16], but they have data association problems as good features are difficult to extract and match. There are two main families of algorithms that are presented in the next subsections, one based on Kalman filter and the other on expectation maximization.

Kalman Filter

A classical approach for generating maps is based on *Kalman Filter*, that was initiated by Smith et al. [143] and later developed by numerous researchers, most notably by a group of researchers located at University of Sydney, Australia.

In the original formulation, a single filter was used to maintain robot position, landmark locations and covariances between them. The computational complexity is $O(n^2)$, where n is the number of features in the map, so it increases with map size. In a recent paper, Thrun [109] developed an algorithm -FastSLAM - that reduces the problem complexity to $O(\log n)$ using particle filters for robot path estimation. Another approach for improving complexity is to decompose the problem into multiple smaller ones [52].

One advantage of Kalman filter based approaches, which made them popular in the mobile robotics community, is the capability of estimating a full posterior probability of robot map, pose and uncertainties in the map. Another advantage is that it was shown - in a formal theoretical study [38] of the evolution of the map, and uncertainty - that the approach converges to the true map up to a residual uncertainty distribution.

One of the main disadvantages of the algorithm is the assumption about the measurement noise as being independent and Gaussian, which can introduce practical limitations. Another is that in general Kalman filter approaches are unable to cope with correspondence problem. This problem can be eliminated by combining the incremental Kalman based approach with a maximum likelihood estimator [96, 154].

Expectation Maximization (EM)

An alternative to Kalman filter solution to SLAM is based on *expectation maximization* or shortly *EM* [153]. EM algorithms are one of the best current solutions to the data association problem. The E-step estimates robot location at different points based on the currently best available map and the M-step estimates a maximum likelihood map based on locations computed in the E-step. Being not incremental, as data has to be processed several times in order to find the most likely map, it can not run in real time.

Recently, real time algorithms have been developed by integrating probabilistic posteriors with ML estimates, known as *incremental maximum likelihood method* [154, 173]. The algorithm computes a single incremental map, by computing posterior probability over all robot poses, and not just the last measurements. The posterior estimate can be implemented using particle filters [36] , a version on Bayes filters using stochastic sampling. The hybrid approaches however cannot work in real time for large environments as the cost of computing posterior probability grows.

3.2.4 Other Geometric Maps

We include here some feature based maps that do not belong to any of the above presented categories and are mostly acquired using vision sensors. *Image patches*, with a distinct signature, are a good way to identify point features in the environment. Sim and Dudek [142] proposed learning visual features using principal component analysis. A tracked landmark is a set of image thumbnails detected in the learning phase. An interesting and robust system that tracks SIFT landmarks (image features invariant to translations, rotations, scaling) was developed by See, Lowe and Little [134]. The distinctiveness of these features makes them easy to track over longs periods of time. An Extended Kalman Filter is used for keeping track of robot and landmark positions and uncertainties. Figure 3.3 shows an example frame with the SIFT features marked and the SIFT database map (right).

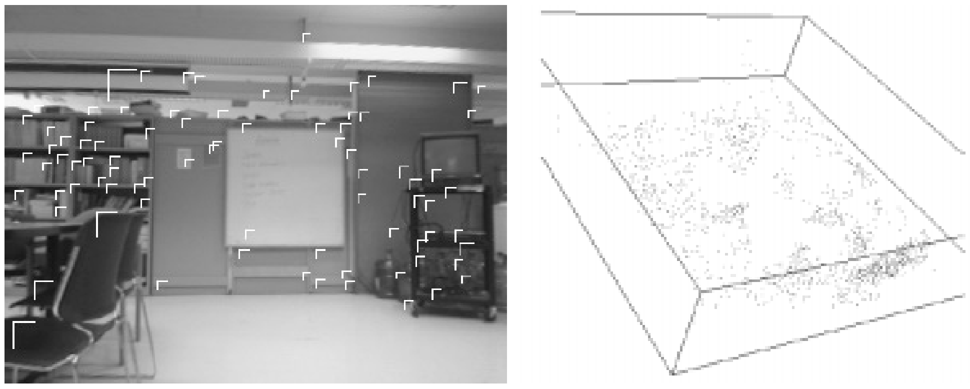

(a) (b)

Figure 3.3: (a) Example of sift features (b) View of the SIFT database [134]

More complex features - shape and objects such as lines, ceilings, doors, walls can be considered when building the navigation map. These maps are sometimes referred as *object maps* [151] and the techniques are borrowed from computer vision and photogrammetry literature. They have the advantage of being more compact, more accurate and closer to human perception, but in general harder to build. Most of them model flat and square surfaces that are common in man-made environments, neglecting the complex aspects of the real world. We believe that by using techniques from image-based modeling for creating object maps, can make them not only easier to acquire but also more realistic.

Different types of sensors (laser range-finders [33, 16], stereo vision [2, 174], or omnidirectional vision sensor [171]) can be used to extract the line segments. Forsberg [50] designed a navigation system in clustered rooms where direction and distances to the walls are determined using range-weighted Hough transform over range data provided by a laser range-finder. Faugeras *et al.* [46] used stereo vision for extracting road direction and the location of curbs. Their algorithm has good performance in indoor as well as outdoor environments. In [89] more complex primitives like planes, cylinders, corners and edges are tracked and matched with an *a priori* geometric map. A complete polygonal model generated from range data using a non-redundant triangular mesh and texture mapping is created in [42]. Thurn *et al.* generated a textured compact 3D model [94]. The algorithm employs the expectation maximization to fit a low-complexity planar model to 3D data collected by range finders and a panoramic camera. Figure 3.2.4 shows a view of the model (data file is in VRML format, which makes it possible to synthesize arbitrary views) .

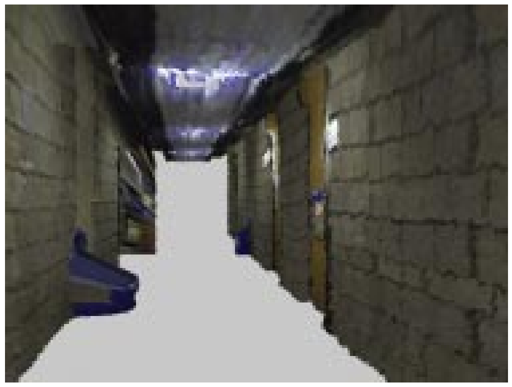


Figure 3.4: View of the VRML 3D model generated using range finders and a panoramic camera [94]

3.3 Image-Based Maps

Images contain rich information about the surrounding environment. That is why a set of images organized in a meaningful way can be used as a map for robot navigation. This map is very easy to build, but it has the disadvantage that it is hard to extract information from images. There are two different approaches in image-based maps. One is to memorize images along a path that should then be repeated by the robot, and the other is to memorize images at fixed locations as reference points in the navigation environment. An omnidirectional image contains views in all directions at a location, so most of the systems use it as a signature of that area. Localization is usually done by matching stored images with the current one observed by the robot.

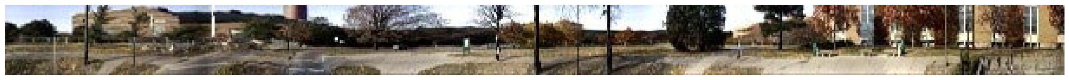
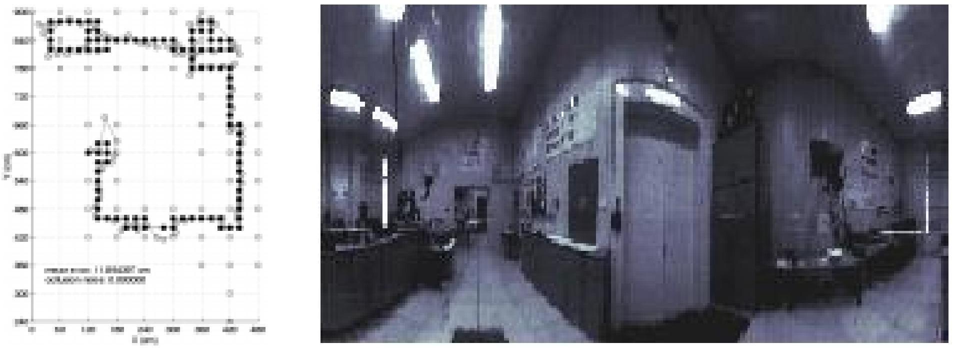


Figure 3.5: Panoramic route representation [175]

One of the earliest works from the first category (route representation) was created by Tsuji [175] where a panoramic representation of the route is obtained by scanning side views along the route. Figure 3.5 shows an example of route panorama. The robot uses the panoramic representation recorded in a trial move and the current one for locating itself along the trial route. In a later work [92] landmarks are extracted from the panoramic representation by segmenting objects based on color and range information. The current landmarks are matched with the stored ones using dynamic programming. In [102] the model contains a sequence of front views along the route. The robot memorizes, at each position, an image obtained from a camera facing forward, and the directional relation to the next view. An interesting approach is presented in [170] where the route is memorized as 2D Fourier power spectrum of consecutive omnidirectional images at the horizon. The robot position is determined by comparing patterns from memorized Fourier power spectrum with the principal axis of inertia.



(a) (b)

Figure 3.6: (a) Original locations and recovered trajectory (b) Example of a panoramic image (cylindrical projection) [75]

The problem with route representation approaches is that the robot has to move along the same pre-stored route. To overcome this problem, omnidirectional images are stored in fixed places of the environment [70, 64, 167, 75]. This representation is very suitable for homing applications where the robot has to move toward a target location. The omnidirectional images used to represent the space are very similar and require a lot of memory space, so they are processed and compressed. Ishiguro [70] transforms the images into the Fourier space; Hong [64] uses a one dimensional signature of the image assuming that the robot is moving on a plane. Winters [167] and Jogan [75] use an eigenspace representation of panoramic images. The former one compresses the images using Principal Component Analysis (PCA), and represents the map as a manifold in the lower dimensional eigenspace obtained from PCA. Johan uses Zero Phase Representation (ZPR) to project differently oriented panoramic images into one representative image and then Singular Value Decomposition (SVD) to find a lower dimensional representation of a set of images in terms of linear combinations in the eigenspace. Figure 3.6 shows a map with the locations of the panoramic images and recovered trajectory (left) and an example of a panorama (right). For eigenspace approaches, localization is done by projecting the representation of the current image into the eigenspace. Other techniques compare the current image with the stored ones and find the optimal position. Ishiguro [70] uses a spring model to arrange the observation points according to the environment geometry. A simpler approach to the homing problem is found in

[5, 6] where the robot is moved toward a goal position specified by a planar image. The translation and rotation that will align the robot with the desired position are computed from the current image and the target image by recovering the epipolar geometry.

3.4 Summary of Mapping Algorithms

This chapter presented a short overview over existing robotic mapping techniques. There are two main categories of maps - geometric and image-based maps. Most of the systems are using geometric maps that contain 2D or 3D information about the navigation environment. They can be further subdivided into occupancy, topological maps, SLAM (probabilistic point feature type) maps and object maps. Image-based maps, more seldom used, use images to represent the navigation space without an explicit geometric model.

The current trend in mobile robotics is probabilistic maps. They are robust to noise, can work in real time, and handle large scale environments. Despite being very promising, there are still problems, and one of the harder ones is data association. We believe that using the image information contained in an image-based map can lead to a solution to this problem. In addition, considering how much information we, people, have about environment, mostly acquired using vision, the mapping alternative presented in this thesis might be a natural way to represent a space. Object maps are along the same idea, but the present solutions are usually very tedious. Another advantage of an image-based type model is that it can very easily and naturally be interpreted by a human operator (e.g. for controlling a mobile robot in a remote location). The operator commands can be given in image space (e.g. by dragging the model) as opposed to the abstract Cartesian coordinates provided by a geometric map.

Chapter 4

Panorama with Depth

This chapter presents the first method for constructing image-based navigation maps using traditional calibrated techniques. The map is formed by a panoramic model enriched with depth information. In the beginning some basic computer vision techniques are summarized for a better understanding of the theory behind the model. The cylindrical panoramic model is then introduced, followed by a description of the two modalities of acquiring depth information (trinocular stereo and laser range-finder) and how depth data is integrated with intensity data. In particular, an innovative depth-intensity image-based registration algorithm is used for registering the cylindrical panoramic image with depth data acquired by the laser range-finder. In the next chapter, for demonstrating the use of the model in mobile robotics, we present applications in robot localization and predictive display.

4.1 Introduction to Computer Vision

This section presents some basic problems and concepts in compute vision starting with the projective camera model and calibration, projections of geometric primitives and, in the end, an introduction to two view geometry in particular the parallel stereo configuration. More advanced geometric vision notions about the modern formulation of multi-view geometry are presented later in Section 6.1.

4.1.1 Projective Camera

Computer vision studies the process of image formation and the world proprieties that can be recovered from images or video. The mathematical image formation abstraction is the camera matrix K, which projects 3D world points to 2D image points through a focal point $\mathbf{C}$ (see Figure 4.1). When representing the points in homogeneous coordinates, K is in general a 3×4 matrix. For the moment we introduce the simplest and most common camera model - the *pinhole model.* This is a *finite camera*, meaning that the center of projection is a finite point. Another class of cameras, infinite cameras, in particular *affine cameras*, that have the center of projection at infinity, are presented in Section 6.1.1.

As Figure 4.1 shows, if the center of projection is chosen as the world coordinate system, the projection equation can be written as follows:

$$x = \frac{fX}{Z} \qquad \text{and} \qquad y = \frac{fY}{Z}, \tag{4.1}$$

The nonlinear equation becomes linear when representing points in homogeneous coordinates:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{4.2}$$

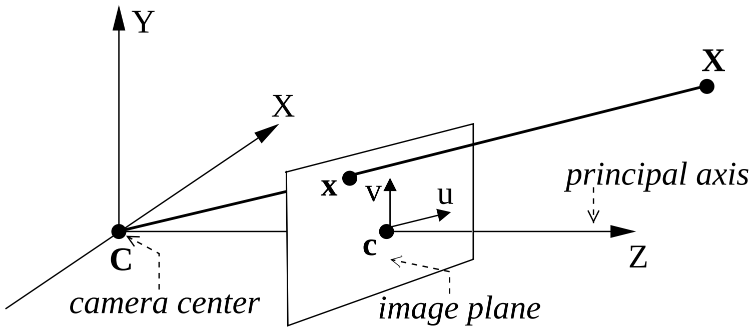


Figure 4.1: Pinhole camera model

where we denoted $\sim$ the point equivalence in homogeneous coordinates.

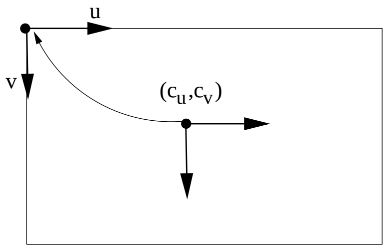


Figure 4.2: From retinal coordinates to image pixel coordinates

Using this projection the image points will be indexed with respect to the principal point (intersection of image plane with the optical axis). In real cameras image pixels are typically indexed with respect to the top left corner of the image and, in addition, the coordinates of the pixel do not correspond to the coordinates in the retinal plane but depend on the physical dimensions of the CCD pixels in the camera. The 2D transformation that transforms retinal points into image pixels is (see figure 4.2):

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} \frac{1}{p_x} + c_u \\ \frac{1}{p_y} + c_v \\ 1 \end{pmatrix} = \begin{bmatrix} \frac{1}{p_x} & 0 & c_u \\ 0 & \frac{1}{p_y} & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{4.3}$$

where (c_u, c_v) represents the principal point location (not always the center of the image) and p_x, p_y the 2 pixel scale factors along horizontal and vertical directions respectively. Combining equation 4.2 and 4.3, we obtain the projection equation in component and standard matrix form as:

$$\mathbf{x} = \begin{bmatrix} \frac{f}{p_x} & 0 & c_u & 0 \\ 0 & \frac{f}{p_y} & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{X}_{cam} = \begin{bmatrix} \frac{f}{p_x} & 0 & c_u \\ 0 & \frac{f}{p_y} & c_v \\ 0 & 0 & 1 \end{bmatrix} [I|\mathbf{0}]\mathbf{X}_{cam} = K[I|\mathbf{0}]\mathbf{X}_{cam} \tag{4.4}$$

Here K is called the calibration matrix and is formed from camera *internal parameters*. More parameters can be introduced to model different real cameras (e.g. skew in the case when the pixels are not rectangular). This parameters are in general related to a particular physical camera and can be determined ahead in the process of camera calibration [161] (see Subsection 4.1.2).

In a general configuration, when the world coordinate system is different than the camera coordinate system, an additional 3D transformation is required:

$$\mathbf{x} = K[R|\mathbf{t}]\mathbf{X} = P\mathbf{X} \tag{4.5}$$

The 3×4 matrix P is called the *camera projection matrix* and has in general 11 DOF. If P can be decomposed like in equation 4.5 (3×3 left hand submatrix M is nonsingular) it is called *finite camera*. If M is singular the camera is called *infinite camera*.

4.1.2 Camera Calibration

Camera calibration [161] deals with computing the camera matrix given images of a known object and is equivalent to the *resection* problem discussed in Subsection 6.1.2. The calibration object is usually built as a 2D or 3D pattern with easily identified features (e.g. checkerboard pattern). The camera matrix is then decomposed into internal and external parameters, assuming a certain camera model. The linear camera model discussed so far can be a bad approximation for a very wide lens camera and radial distortion has to be considered. The radial distortion is small near the center but increasing toward the periphery, and can be corrected using:

$$\hat{u} = u_c + L(r)(u - c_u) \tag{4.6}$$
$$\hat{v} = v_c + L(r)(v - c_v) \tag{4.7}$$

where $(\hat{u}, \hat{v})$ denotes the corrected point for (u, v) and (c_u, c_v) is the principal point of the camera and $r^2 = (u - c_u)^2 + (v - c_v)^2$. $L(r)$ is the distortion function and can be approximated with $L(r) = 1 + k_1 r + k_2 r^2 + ...$ The coefficients of the radial correction k_1, k_2 are considered part of the internal camera parameters.

4.1.3 Image Projection of Geometric Primitives

A *point* in 3D is projected on the image plane on a 2D image point using equation 4.2. The image point can geometrically be obtained by intersecting the ray defined by the 3D point and the camera center of projection with the image plane.

A *line* in 3D projects to a line in the image plane. Geometrically, as illustrated in Figure 4.3, the line projections obtained by intersecting the plane defined by the 3D line and the camera center with the image plane. Algebraically, the 3D line can be represented by a point $\mathbf{d}$ and a direction $\mathbf{v}$. Denoting with $\mathbf{m}$ the normal of

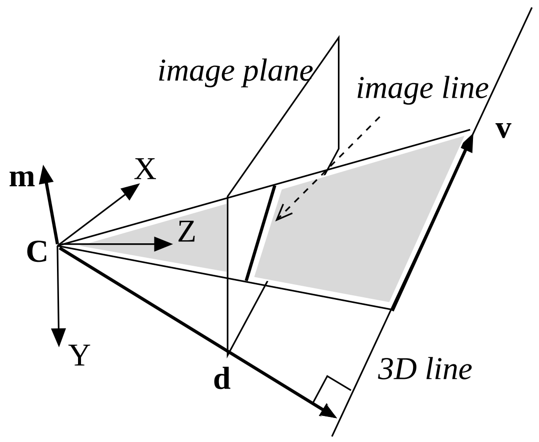


Figure 4.3: A 3D line can be represented as $(\mathbf{v}, \mathbf{d})$. The image projection is represented by $\mathbf{m}$

the plane defined by the camera center and the 3D line, the equation of the image line is:

$$m_x x + m_y y + m_z z = 0 \tag{4.8}$$

where $\mathbf{m} = (m_x, m_y, m_z) = \mathbf{v} \times \mathbf{d}$. If the camera is registered with the reference coordinate system and has focal length 1, $\mathbf{m}$ represents the projective coordinates of the image line. Note that in 2D projective space a line has 3 coordinates (same as the point). Different formulation of line projection equation can be obtained using Plücker representation [59].

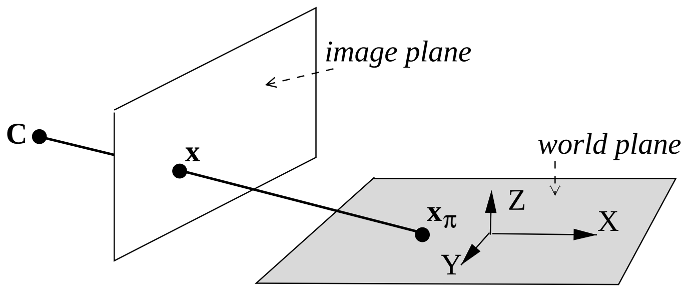


Figure 4.4: Points on a 3D plane are in general transfered to the image plane through a 2D projective transformation (homography).

Points on *plane* π can be represented as $(X, Y, 0, 1)^T$ when choosing the reference coordinate system aligned with the plane (see Figure 4.4). The projection equation 4.5 can then be rewritten as:

$$\mathbf{x} = P\mathbf{X} = \left[\begin{array}{cccc} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \end{array}\right] \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \left[\begin{array}{ccc} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_4 \end{array}\right] \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = H\mathbf{x}_\pi \tag{4.9}$$

where $\mathbf{p}_i$ denotes column i in the projection matrix. So, in general, the map between points of the plane $\mathbf{x}_\pi = (X, Y, 1)^T$ and their image $\mathbf{x}$ is a 3×3 matrix named planar homography or 2D projective transformation. Considering projections of points from the same physical plane on two images, it can easily be shown that, they are related by the same type of transformation (2D homography).

4.1.4 Two View Geometry

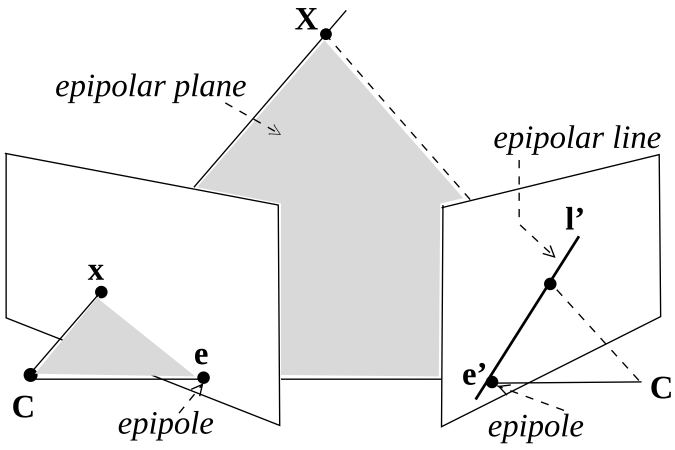


Figure 4.5: Epipolar geometry

Epipolar geometry

Without having any information about the position of a 3D point, given its projection in one image, its projection is restricts in a second image to a line that is called the *epipolar line*. This is the basic geometric constraint for the two view geometry and is illustrated in Figure 4.5. Geometrically it can be easily be proven by considering the plane that connects the camera centers $\mathbf{C}, \mathbf{C}'$ and the point $\mathbf{x}$ in the first image - *epipolar plane*. This plane intersects the second image plane on a line that is the epipolar line. Note that all epipolar lines in an image have a common point - the projection of the second camera center. This point, is called the *epipole* and is denoted $\mathbf{e}$ and $\mathbf{e}'$ respectively for first and second camera in Figure 4.5.

Algebraically, the epipolar constraint is formulated using the *fundamental matrix* F [47]. F is a 3×3 matrix that has rank 2 (the epipole $\mathbf{e}$ is the nullspace of F). A lot of effort has been put in robustly estimating F from a pair of uncalibrated images (e.g [156]). It can be estimated linearly given 8 or more corresponding points. A nonlinear solution uses 7 corresponding points, but the solution is not unique. Here are some important properties of the fundamental matrix that allows the computation of the epipolar lines and epipoles.

$$\mathbf{l}' = F\mathbf{x} \quad \mathbf{l} = F^T\mathbf{x}' \tag{4.10}$$

$$F\mathbf{e} = 0 \quad F^T\mathbf{e}' = 0 \tag{4.11}$$

Calibrated stereo

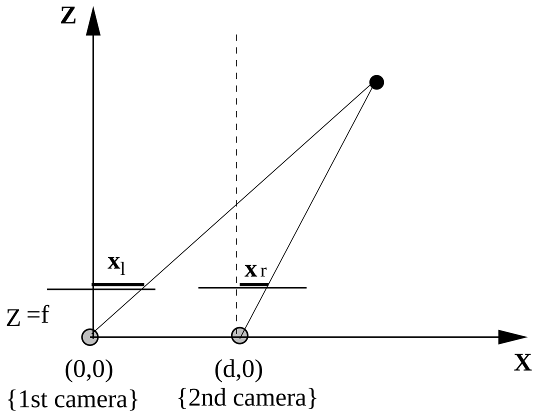


Figure 4.6: Calibrated stereo geometry

A special case of the two view geometry arises when the cameras are calibrated and aligned (the image planes are parallel with the line connected the camera centers). A 1D illustration of this configuration is presented in Figure 4.6. In this case the epipolar lines correspond to image rows so corresponding points in two images are on the same scanline. The horizontal displacement between two corresponding points is called *disparity*. This special case has practical application in computing dense depth from images. Correspondence problem is simplified (corresponding point is on the same scanline) and there is a simple solution for depth, knowing the focal length of the cameras f and the distance between the camera centers d:

$$Z = \frac{df}{x_l - x_r} \tag{4.12}$$

where $x_l - x_r$ represents the disparity (horizontal distance between corresponding calibrated image points). Note that the cameras have to be calibrated and x_l, x_r normalized with respect to image scale and center. An example of a depth map computed using a trinocular vision system (from Point Gray Research [131]), used for one of the models presented in this thesis in Section 4.3, is presented in Figure 4.7.

4.2 Cylindrical Panoramic Mosaic

Image mosaicking, as described in Section 2.2.1, deals with merging a collection of images into a bigger image. A panoramic mosaic captures a 360° degree view of

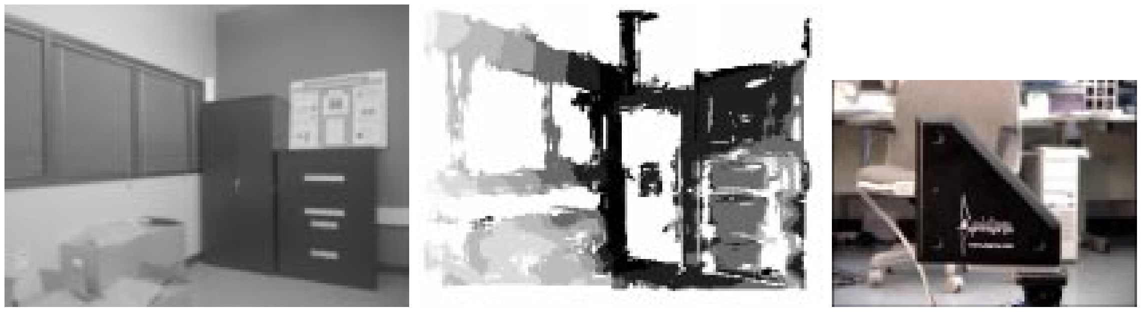


Figure 4.7: Example of a disparity map computed using the Triclops vision system. (a) left intensity image (b) disparity map (c) Triclops device

the environment and can be constructed by composing planar images taken from a single point of view. This mosaic is geometrically correct because the input images are related by a 2D projective transformation (homography).

In this research, a CCD camera was mounted on a pan-tilt unit that could be programmed in order to control the amount of rotation. The camera was fully calibrated using Tsai's algorithm [161]. The rotating camera took an image every 5-10 degrees and they were blended into a mosaic as described next. Using the camera model from equation 4.4,

$$K = \begin{bmatrix} a_u & 0 & c_u \\ 0 & a_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{4.13}$$

a pixel $\mathbf{x} = (u, v)^T$ backprojects on the 3D ray $\mathbf{d} = (d_x, d_y, d_z)^T$, according to:

$$d_x = \frac{\mathrm{u} - u_c}{a_u} \quad d_y = \frac{-\mathrm{v} + v_c}{a_v} \quad d_z = -1 \tag{4.14}$$

Each acquired image was projected onto a cylinder with the radius equal to the focal length of the camera f through the mapping:

$$\theta = \tan^{-1}(\frac{d_x}{d_z}) \quad v = f\frac{d_y}{\sqrt{d_x^2 + d_y^2}} \tag{4.15}$$

The projected images "stitched" or correlated in order to precisely determine the amount of rotation between two consecutive images. In the cylindrical space a translation becomes a rotation, so we can easily build the cylindrical image by translating each image with respect to the previous one. To reduce discontinuities in intensity between images we weight the pixels in each image proportionally to their distance to the edge [148]. Figure 4.8 (a) presents a 360° cylindrical panorama constructed using the presented technique. More examples are shown in the following sections.

4.2.1 Depth Data

The panoramic model does not contain any depth information (parallax), so a single model can only be reprojected from the same viewpoint where it was acquired, and has limited use in any robotic application when the robot should be able to freely move in the navigation space. A solution would be to consider multiple panoramic image from relatively distant viewpoints. A major problem in this case is the *correspondence problem*[1] that is very difficult for planar and relatively close images and becomes very challenging for the case of distant panoramas. This option is investigated in a robot localization application in Section 5.1. An alternative solution is to enrich the model with dense depth data that can then be use to generate correct re-projections. Two modalities of acquiring depth data are described in the following subsections: a trinocular vision system and a laser range-finder.

4.3 Depth from Stereo

4.3.1 Depth Panorama

The depth panorama is acquired using a trinocular vision system from Point Grey Research [131] (see Figure 4.7). This system consists of three cameras and produces a real time disparity map. The disparity information was used along with the intensity data to produce a "stereo" panorama. A similar approach is presented in [80], with the difference that they used two panoramas to produce the depth map, while, in this work, the disparity information is provided by the trinocular system for each of the images to be composed in the panorama. In this way, there are significant quantization errors in the generated disparity map because of the small baseline, although the correspondence problem is easier to solve (see Subsection 4.3.2).

The trinocular system is rotated around the optical center of the reference camera. Along with the intensity cylindrical panoramic image, we also build the corresponding depth map. The result of the mosaicking technique is presented in Figure 4.8(a) and the corresponding "depth" map in Figure 4.8(b).

It is known that depth from stereo has physical limitations due to calibration of cameras, quantization on CCD and a relatively small baseline compared to object's depth - as easily derived from equation 4.12. Some other algorithm related source of errors like establishing correspondences might introduce more noise. For filtering depth data, but also to provide strong features for the localization algorithm (Section 5.2) that is using the model, we developed an algorithm for extracting planar patches that will enrich the current panoramic model. Next subsection

[1]The correspondence problem is the problem of relating/identifying geometric primitives or features that correspond to the same physical entity.

presents a segmentation algorithm based on both intensity and range data. The approach was developed on planar images and then applied to cylindrical mosaics.

4.3.2 Planar Patches

Most of the current algorithms for extracting planar regions are based on range image data. Several algorithms are known in the literature for segmenting range images in planar regions. They can be classified into three main categories: *region growing*, *split-and-merge* and *clustering* methods. An experimental comparison of the range image segmentation algorithms [44] has shown that this problem is far from being "solved".

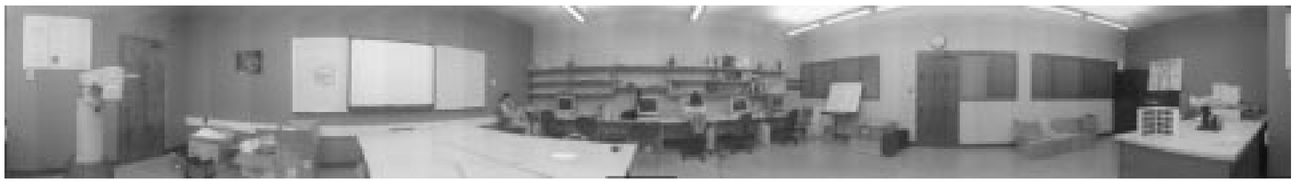
Region growing approaches start with a fine segmentation of the initial image and then bigger regions are grown based on similar plane equations. The initial segmentation can be obtained by fitting a plane to each pixel based on an $N \times N$ window [44], or by generating a mesh from the original 3D points using traditional graphics techniques like Delaunay triangulation [47, 79]. The approach presented in [74, 113] is based on the observation that, in the ideal case, the points on a scan line that belong to a planar surface form a straight 3D line segment. Therefore one can first divide each scan line into straight line segments and then perform region growing using the set of line segments. In [79] the boundary of the final regions is further simplified in order to accommodate noise in the range data.

In [119], the split-and-merge approach originally proposed by Horowitz and Pavlidis [66] for intensity images was extended to range images. Schmitt and Chen [133] propose a new split-and-merge algorithm that uses a Delaunay triangulation as the basis for an adaptive surface approximation technique.

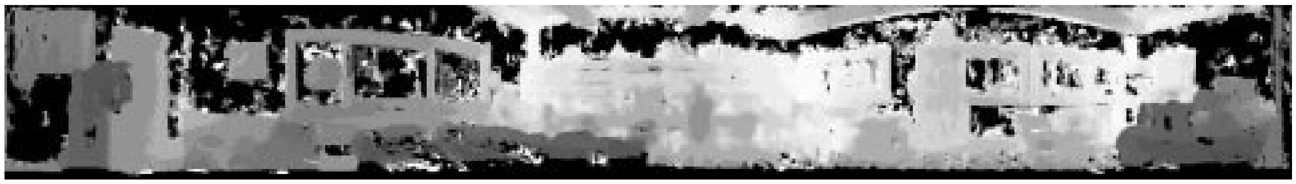
Clustering represents another class of algorithms for segmenting range data into planar regions. The feature space in which the clustering takes place differs from one algorithm to the next and can be a pixel [76], or a pixel and the estimated normal [44].

All these algorithms use a best fitted plane to range data to verify the planarity of a region. An interesting texture-based planarity constraint is presented in [122]. Having an initial triangulation of the scene in two images, the planarity of each triangle is determined based on the correlation between the texture of matched triangles after rectifying them to a common viewpoint.

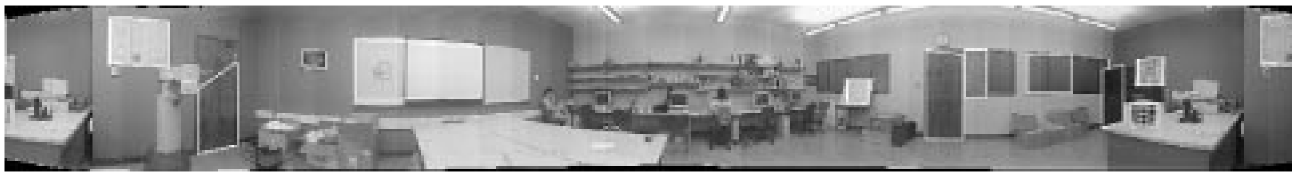
Most of the current systems segment the range image data into planar regions, which are then integrated into a 3D model. They rely on dense and accurate depth information that is usually acquired using laser or structured light range finders. As mentioned before, depth from stereo has errors caused by limitations due to calibration and CCD quantization or other algorithmic inaccuracies like mismatched correspondences (for a performance comparison of stereo algorithms see [82]). As an example, considering the Triclops baseline (10 cm), the error in depth for a distance of 10 m is about 0.5 m. Another characteristic of stereo algorithms is that they generates disparity only for textured ares (where correlation



(a)



(b)



(c)

Figure 4.8: (a) Cylindrical panoramic model of a research laboratory; (b) Depth map: dark - close objects; whither - far objects; black - no depth value; (c) Extracted planar patches

makes sense). The developed image-based panoramic model is designed for indoor robot navigation, where most of the planar patches, like doors, walls, cabinets, either have regular or do not have any texture, but they are visually distinctive in the intensity image. So using disparity information alone in extracting planar patches would not be appropriate in our case. Instead, we designed a planar patch extraction algorithm that uses a combination of intensity and depth information.

The main observation that led us to the current algorithm is that in a typical indoor environment, most of the planar regions have an intensity distinct from the surrounding regions. So the first step in the segmentation algorithm is a region growing approach based on average intensity. This algorithm is summarized in Subsection 4.3.2. Next, depth information is used to segment the regions generated by the region growing algorithm, based on a planarity test. To compensate the errors in depth data, we use a generalized Hough transform to eliminate the bad points. Subsection 4.3.2 describes this planar patch selection approach. Subsection 4.3.2 presents some experimental results for evaluating the robustness of the algorithm.

Intensity based segmentation

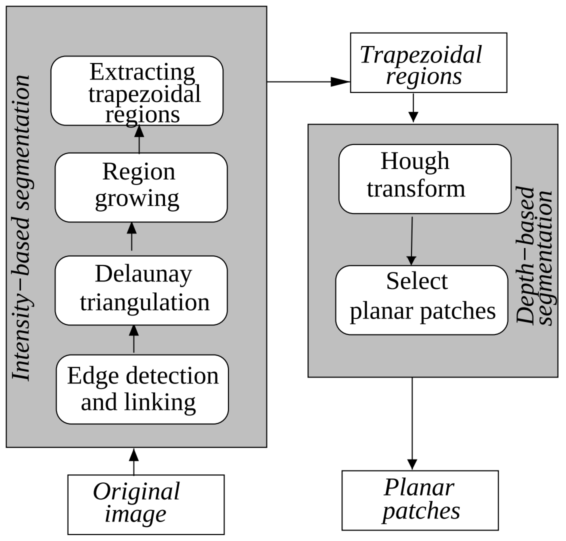


Figure 4.9: Flow chart for planar patch extraction algorithm

The flow chart of the segmentation algorithm is presented in Figure 4.9. The fundamental structure used by the global region growing algorithm is a triangular mesh. The segmentation algorithm takes place in the image domain so the mesh is also generated in pixel space. We choose a constrained Delaunay triangulation [139] based on edge segments to construct our 2D mesh because it generates a connected mesh with disjoint triangles. The edge segments input to the triangulation

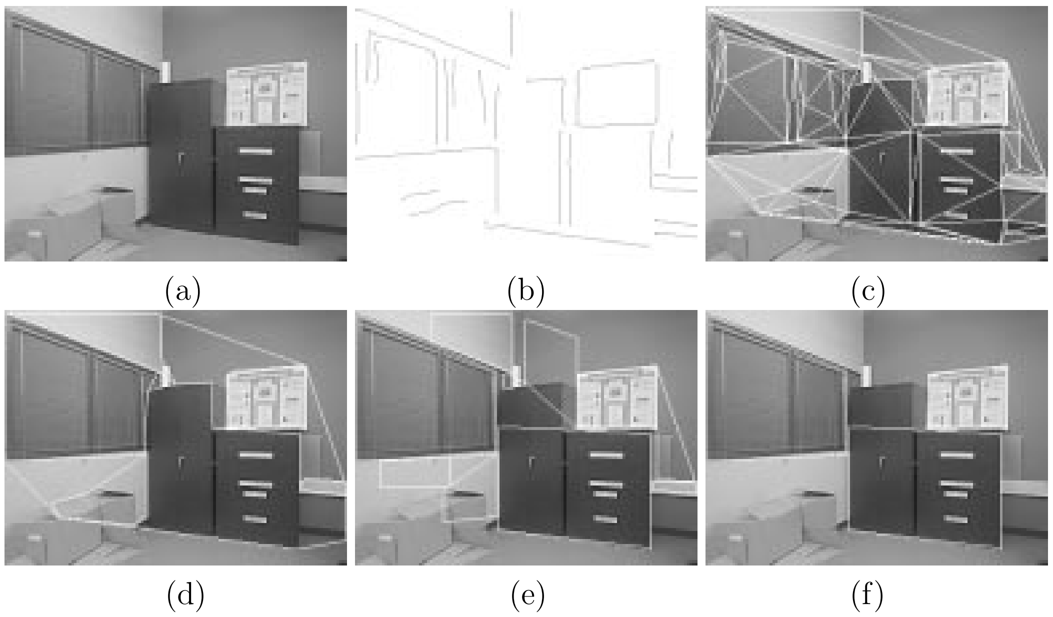


Figure 4.10: Planar region segmentation (a) Original image; (b) Edge detection and linking; (c) Constraint Delaunay triangulation; (d) Region growing (e) Extracted trapezoidal regions; (f) Vertical planar regions;

algorithm are edges of the resultant mesh. The segmentation algorithm extracts regions with distinct average intensity that should have also distinctive edges.

For edge extraction and linking we used the code provided by Dr. S. Sarkar at University of South Florida [132]. Their edge detection algorithm is an adaptation of the optimality criterion proposed by Canny to filters designed to respond with a zero crossing. For edge linking, they segment an edge chain into a combination of straight lines and constant curvature segments. Figure 4.10(b) presents the edge image after the edge detection and linking algorithm is applied to original image (Figure 4.10(a)), and Figure 4.10(c) presents the result of constrained Delaunay triangulation with the edge segments.

The global region growing algorithm starts with the triangular mesh and merges the initial triangular regions into larger ones that have similar average intensity. The process stops when a threshold in the number of regions or total mesh error is passed. We used a modified version of the region growing algorithm presented in [47, 79].

From the initial triangular regions, *region adjacency graph* is created, where the vertices represent the regions and the edges indicate that two regions are adjacent. Each edge is weighted by the error given by

$$E_{ij} = \sum_{u,v \in R_i} \frac{I(u,v)}{N_i} - \sum_{u,v \in R_j} \frac{I(u,v)}{N_j} \tag{4.16}$$

where R_i and R_j are the adjacent regions that share the edge, I is the initial image to be segmented, and N_i represents the number of pixels from region R_i. Larger regions are grown from the initial mesh by merging adjacent regions. At each iteration the two regions that produce the smallest error E_{ij} are merged. This guarantees that the total error grows as slowly as possible. After each merge the adjacency graph is updated.

There are two thresholds for stopping the region growing process. One is the total number of regions and the other is an upper bound for the total error. In our case the first one works better.

The resulting regions are presented in Figure 4.10 (d). Final regions usually have irregular shapes that can be either concave or convex. We developed a heuristic algorithm that extracts the biggest trapezoid out of a region. The algorithm proceeds by first filling all the small interior holes and then finding the biggest rectangle included in the original region. For easily testing if a certain pixel belongs to the current region or not, we created a black and white image that contains only the current region. We then detect the bigger interior rectangle - R_h - by horizontally scanning the image. The initial rectangle is the longest vertical scan scan line of the current region. This rectangle is extended in both left and right directions till its area stops growing. The procedure is repeated for vertical scan to obtain R_v. The final rectangle is the biggest one between R_h and R_v. This is then expanded up and down into a trapezoid to fit the original region shape. The result of this algorithm is shown in Figure 4.10 (e).

Planar region selection

The trapezoidal regions that result from the intensity based segmentation algorithm are distinct regions not necessary planar. This section describes the algorithm that thresholds these regions based on a planarity error measure and some properties of the corresponding 3D plane.

The trinocular system provides 3D information for some of the interior points in each trapezoidal region. This depth data is very noisy, so, before calculating the best fitted plane to region points, we eliminate the outliers using a generalized Hough transform [54].
Consider the plane equation

$$X \sin\phi \cos\theta + Y \cos\phi + Z \sin\phi \sin\theta + \rho = 0 \tag{4.17}$$

where ρ is the distance to the plane from the origin, and ϕ and θ are the spherical coordinates of the normal (see Figure 4.11). There is an infinite number of planes that pass through the point $(X, Y, Z)^T$, but they all satisfy equation 4.17. By fixing X, Y, Z and considering ϕ, θ, ρ as the parameters of equation 4.17, we will have a surface in the parameter space corresponding to the point $(X, Y, Z)^T$. All the points belonging to a certain plane will have the corresponding surfaces in the parameter space intersecting at one point $(\phi, \theta, \rho)^T$ which represents the parameters of that plane. We subdivide the parameter space in cells according to expected

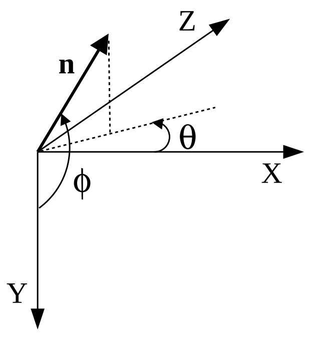


Figure 4.11: Spherical coordinates for normal to the plane

ranges of ϕ, θ, ρ and then compute the corresponding cells for each 3D point by incrementing ϕ and θ and calculating ρ from equation 4.17. The range for ϕ and θ is $[0°, 180°]$. The parameters of the cell that contains the largest number of points give an approximation of the plane fitted to the region points, so we will keep only these points for computing the exact equation of this plane. Figure 4.12 shows a point cloud before (a) and after (b) the Hough transform.

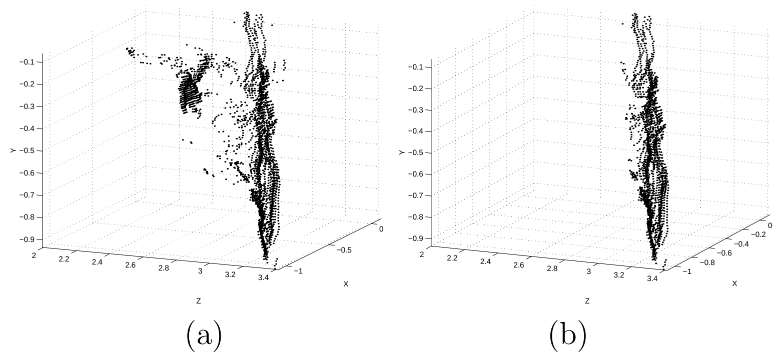


Figure 4.12: Point cloud for a planar region before (a) and after (b) the Hough Transform

For computing the plane equation of a planar patch, we compute the best fitted plane that approximates the points selected by Hough transform. The plane parameters $(\mathbf{n}, d)$ are determined by minimizing the error measure

$$E = \sum_{i=1}^{N} (\mathbf{n}^T \mathbf{X}_i + d)^2 \tag{4.18}$$

where $\mathbf{X}_i$ are the points in the region, N is the number of points, $\mathbf{n}$ in the unit normal of the plane, and d is the distance from the origin to the plane. This is a classical non-linear optimization problem [47] and the solution for the plane normal $\mathbf{n}_{min}$ is an eigenvector of length one of the covariance matrix $\mathbf{\Lambda}$ associated with the smallest eigenvalue λ, which is also the minimum error. The covariance matrix is given by

$$\mathbf{\Lambda} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{A}_i\mathbf{A}_i^T, \ \mathbf{A}_i = \mathbf{X}_i - \mathbf{X} \ \text{ and } \ \mathbf{X} = \frac{1}{N}\sum \mathbf{X}_i$$

The minimum distance to the best fitted plane is given by

$$d_{min} = -\frac{1}{N}\sum_{i=1}^{N}\mathbf{n}_{min}^T\mathbf{X}_i$$

After computing the plane that best approximates the points in each region we decide if this is a real planar patch by thresholding the plane error (equation 4.18), number of points with disparity relative to patch size, and patch dimension in the image space. For the robot navigation application we are interested in extracting only planes that are almost vertical so another threshold criterion is the normal angle with the horizontal plane. Figure 4.10(f) shows the extracted vertical planar patches.

We used this segmentation algorithm to extract planar patches out of the cylindrical panoramic model. For each planar patch we store its position in the panoramic image and the corresponding plane equation. The result is presented in Figure 4.8(c). The **panoramic model with depth from stereo** is formed from intensity and depth panoramic mosaics and vertical planar patches represented as trapezoidal regions in image space and with the plane equation in 3D space. This represents our first developed image-based map for robotics and was applied for a localization algorithm that is described in Section 5.2.

Evaluation of the plane segmentation algorithm

To compare the performance of our algorithm with others, we implemented Kang and Szeliski's range-based segmentation method [79]. The result is presented in Figure 4.13 that shows the algorithm's inability to handle noisy range data. For example the poster from the right part of the image is considered to belong to the same plane as the cabinet below that is about 0.5 m in front of it. This is because of the noise present in the depth data resulting from the trinocular system. Figure 4.10(f) shows the output of our segmentation method where the poster and the cabinet are correctly separated into two distinctive regions.

After applying the segmentation algorithm, for evaluating the correctness of the plane equation for the extracted planar patches, we produce an image rendered from a different view point than the original one. For producing the rendered image, we first project the corresponding 3D points for the corners of each rectangular patch on the evaluated plane, and then re-project them from a new viewpoint. All the interior points of the rectangular region from the original image belong to the same plane so, by texture mapping the interior of each patch we obtain a geometrically correct image. Figure 4.14 shows an example of a rendered image for the three extracted planar patches from Figure 4.10(f).

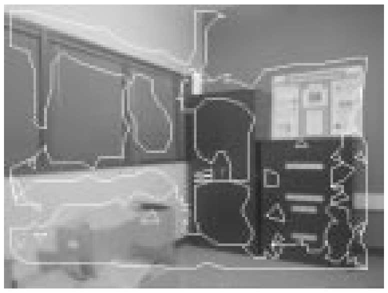


Figure 4.13: Results of a range data segmentation algorithm

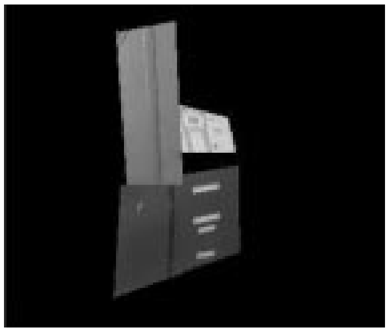


Figure 4.14: Image with the planar patches rendered from a different point of view

4.4 Depth from Laser Range-Finder

An alternative modality to acquire depth data uses a laser range-finder. This section describes the steps involved in building a cylindrical panoramic model enriched with depth data from a laser range-finder. The main difference from the approach presented in the previous section is that, an additional registration step is required for aligning the depth and intensity data. Beside this, the resolution of the depth data is sparser than the resolution of the intensity data, so only some pixels in the panoramic mosaic will have depth values. The first step in model building involves data acquisition where both intensity and range data are obtained and mapped to a cylindrical coordinate system. Subsequently, the two data sets are registered under a common coordinate system. For further refining the model, planar patches are fitted to registered data and vertical lines are extracted as intersections of vertical planes.

4.4.1 Range and Intensity Image Acquisition

The data acquisition system consists of a *laser range-finder* (Acuity Research, Inc.) and a CCD *camera*, mounted on a *pan-tilt unit (PTU)* (see Figure 4.15). We use the pan axis of the PTU to rotate the camera in order to build a cylindrical or panoramic image model. The same pan axis of the PTU and a rotating scanner attached to the laser range finder produce two degrees of freedom, sufficient for spanning a unit sphere to acquire complete range information. Once two separate images are obtained, they will be registered to generate the final image-based model enhanced with 3D geometric features.

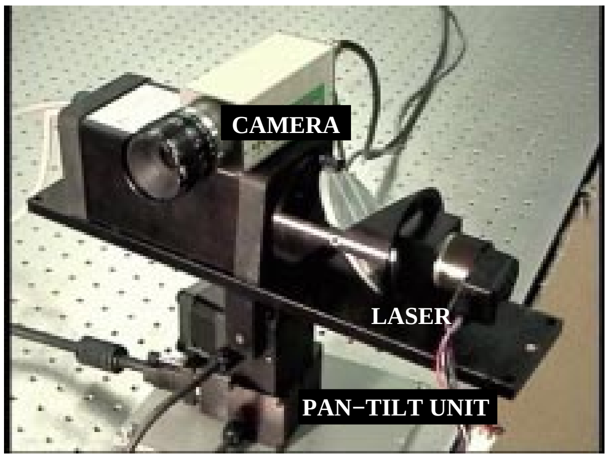
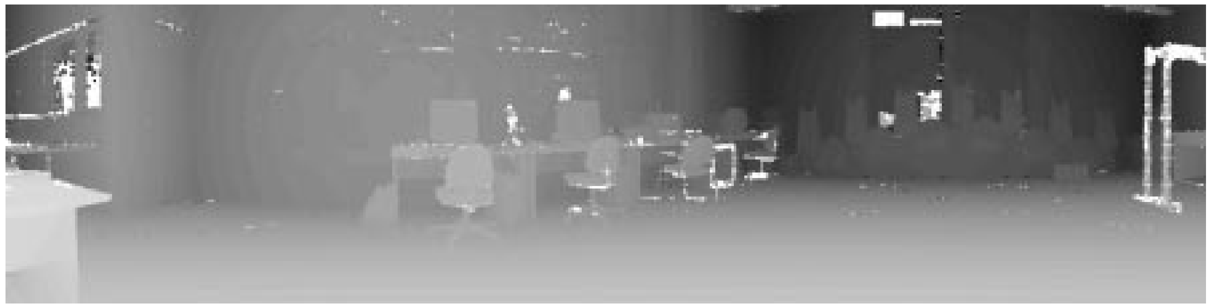


Figure 4.15: System configuration: the laser-range-finder with the camera attached on top of it is mounted on a pan-tilt unit

The data returned for each sample of the laser range finder consist of a range r, an amplitude a, and angular position of the rotating mirror θ of the scanner. The amplitude a corresponds to the strength of the returned signal, and is related to, among other things, both r and the gray-scale value of the reflecting surface. Since the pan angle ϕ of the PTU is also known, each sample can be expressed as a quadruple (r, a, θ, ϕ), and it can be considered as two images sampled on a spherical surface: a range image r and an amplitude image a. We ignore the small translation between the mirror center of rotation and the pan-tilt unit center, and apply a filter to eliminate outliers in the range data. Finally, because the laser range finder often does not generate uniformly sampled data, due to a variety of reasons such as noise and non-reflecting surface, a 3×3 median filter was applied over the neighboring samples to fill the missing values and create a uniform grid. The top figure in Figure 4.16 shows a scaled spherical range image, r, representing 180° scan of our navigation environment. In our scheme for representing depth values, the close objects appear in lighter shades than far objects. The corresponding 180° panoramic mosaic constructed using the algorithm described in Section 4.2 is shown in the bottom figure of Figure 4.16.



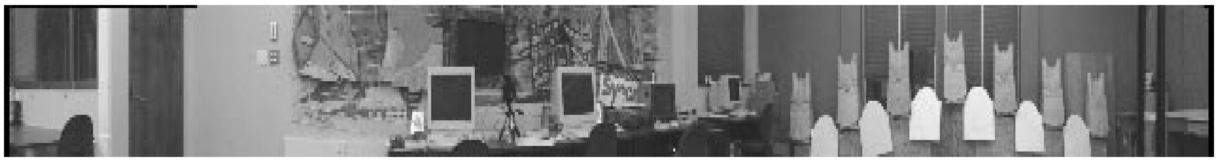


Figure 4.16: (top) Spherical representation of the range data from an 180° scan after filtering (bottom) Corresponding 180° panoramic mosaic.

4.4.2 Range-Intensity Image Registration

The registration of volumetric and intensity data is an important problem especially in the fields of model building and realistic rendering. Most of the proposed solutions are recovering the rigid transformation between the sensors using point or line features [144, 85]. This is in general a non-linear problem and requires an initial estimate to converge. In contrast, image-based techniques compute a direct mapping between the points in the data sets to recover the transformation. The accuracy of this method depends on the original assumptions about the physical system (*e.g.* affine camera or planar scene) but in general they are good for locally recovering a mapping between data sets. In [30] we compared the two approaches and find that image-based methods are fast and adequate for application that does not require a high precision alignment. In the setup presented here the two sensors are very close to each other an image to image warp is suitable for aligning the two data sets.

The first step in the registration algorithm is to project the spherical range data into a cylindrical representation with the radius equal with the focal length of the camera. This mapping is given by

$$\mathbf{X}(r,\theta,\phi) \mapsto \mathbf{X}(r,\theta,f\tan\phi) = \mathbf{X}(r,\theta,h) \tag{4.19}$$

where r represents the distance from the center of the cylinder to the point, h is the height of the point projected on the cylinder, θ is the azimuth angle and f the focal length of the camera. Again, this data is sampled on a cylindrical grid θ, h and represented as a cylindrical image. The same procedure is applied to the amplitude data and get the cylindrical amplitude image.

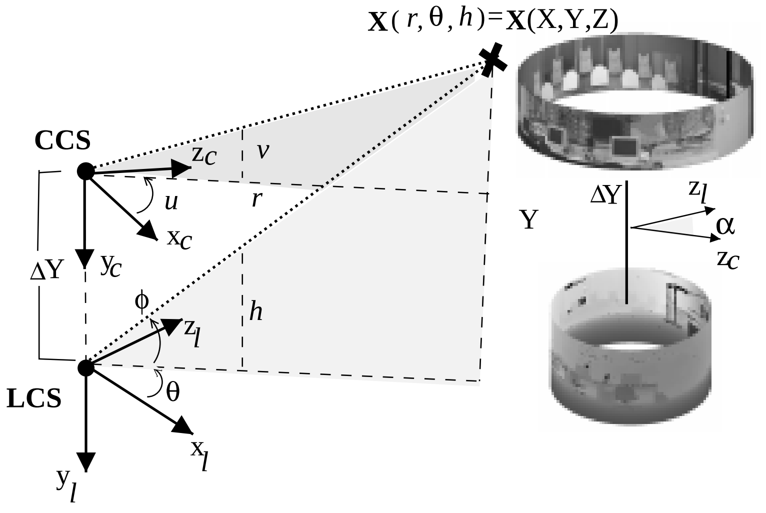


Figure 4.17: The projection of a space point $\mathbf{P}$ in the cylindrical image (θ, h) and the panoramic mosaic (u, v). We approximate the laser-camera transformation with a translation ΔY and a rotation over y axis.

From the intensity and range data in similar cylindrical image representations, we compute a *global mapping* between them. The physical configuration of the sensors is approximated, as in Figure 4.17, assuming only a vertical translation ΔY and a pan rotation between the two reference coordinate systems LCS (laser coordinate system) and CCS (camera coordinate system). For a point $\mathbf{x}_l(\theta, h)$ in the cylindrical laser image, its corresponding point in the panoramic mosaic $\mathbf{x}_c(u, v)$ is

$$\begin{array}{l} u = a\theta + \alpha \\ v = f\frac{Y-\Delta Y}{r} = f\frac{Y}{r} - f\frac{\Delta Y}{r} = bh - f\frac{\Delta Y}{r} \end{array} \tag{4.20}$$

where a and b are two warp parameters that will account for difference in resolution between the two images, α aligns the pan rotation and $Y = r\frac{h}{\sqrt{f^2+h^2}}$ is the height of the 3D point $\mathbf{X}(r, \theta, h)$. For the presented setup, $f = 1000$ pixels, $\Delta Y = 5$ cm and the range of the points is $r = 5 - 8$ m, so $f\frac{\Delta Y}{r} = 6 - 10$ pixels and it can be approximated to a constant $-\beta$. The general warp equations are:

$$u = a\theta + \alpha, \quad v = bh + \beta \tag{4.21}$$

The warp parameters (a, b, α, β) are computed from two or more corresponding points in the two images using a least square approach.

After the global mapping, the two data sets are only approximately aligned with a misalignment of $5 - 7$ pixels. We perform a *local alignment* using a set of corresponding control points. The local map "stretches" the range data to fit the intensity data using cubic interpolation based on a 2D Delaunay triangulation of the control points.

After the data registration algorithm, each range data point acquired by the laser range-finder has a corresponding pixel value in the panoramic image. Because the resolution of the panoramic mosaic is bigger than the resolution of the range image, this relation does not hold the other way, more explicitly, there is no corresponding range value for every pixel in the panorama. To solve this problem the range values are interpolated for the points in-between.

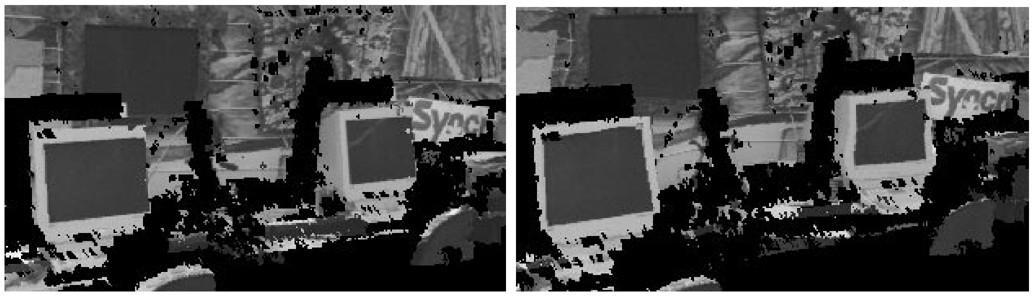


Figure 4.18: (top) Rendered image using only the global mapping (bottom) Rendered image with local alignment. Notice how misalignment from the top edge of the left monitor is compensated.

For visualizing the performance of the registration algorithm, the navigation room was rendered from different positions than the one where the model was taken. The registered range points on the panoramic image are triangulated using a 2D Delaunay triangulation and the the corresponding image triangles are rendered using OpenGL. In a complete mesh, some of the triangles might not represent physical planes but are artifacts of occluding contours, and in most of the cases this appears at silhouette edges [104] where points from the object are connected with background points. To avoid this phenomenon,all the triangles within a threshold that are parallel to the viewing direction are eliminated. Figure 4.18 shows the rendered view after global mapping (top) and with additional local alignment (bottom). Notice that the misalignment of the texture on the computer monitor is corrected with the local mapping technique. The bad quality of the rendered images is due to some inaccuracies present in depth data. Part of these will be eliminated by the plane fitting algorithm presented in the next subsection.

4.4.3 Robust Plane Fitting

Man-made environments are mostly composed from planar regions. A common technique in data modeling [144, 148] is to segment the range (3D data) into planar regions. In most of the cases, the goal is to create a 3D CAD-like model of the

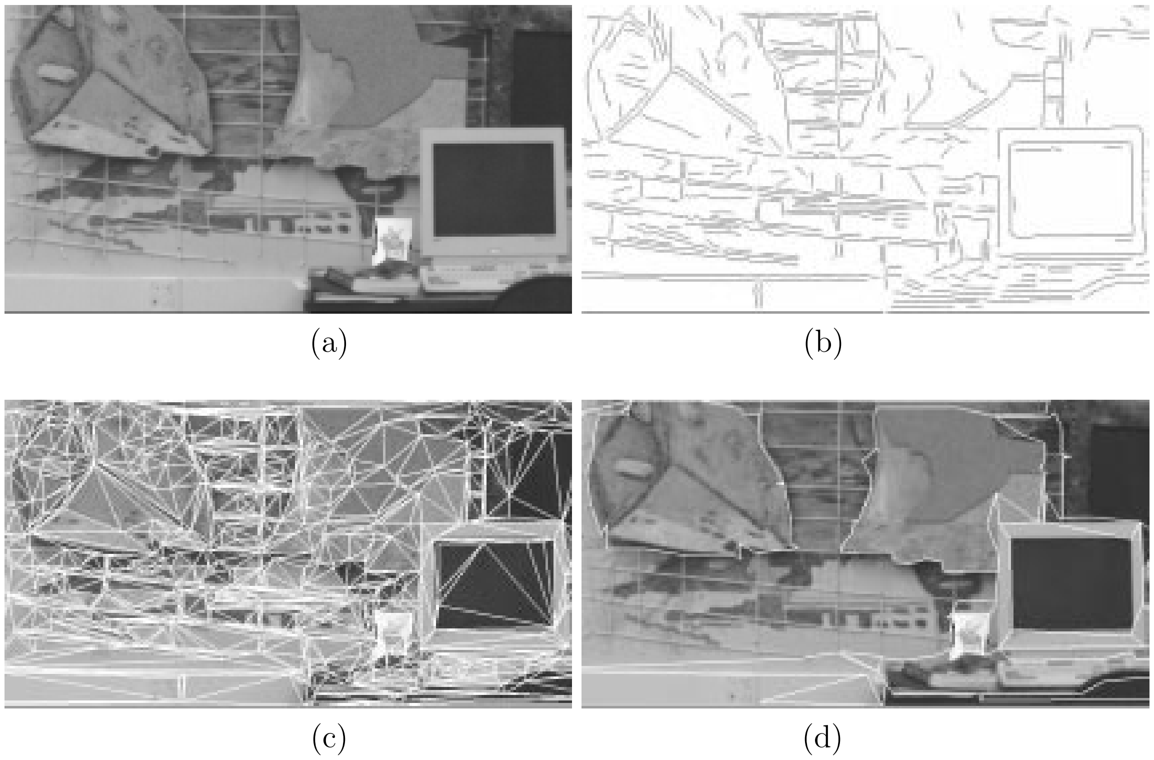

(a) (b) (c) (d)

Figure 4.19: Planar region segmentation: (a) Original image; (b) Edge detection and linking; (c) Constraint Delaunay triangulation; (d) Merged planar regions

scene, composed of planar regions that are suitable for rendering. In our case, the goal is not to reconstruct a full 3D model of the scene, but to extract a sparse set of robust 3D features (e.g. lines) that are required by the localization algorithm (see Section 5.3). So, the plane fitting algorithm was used to eliminate bad data and create a cleaner model. For planar region detection, the algorithm is very similar to the planar patch detection algorithm from Section 4.3.2 with the difference that the planes are fitted directly to the range data that is much more reliable when acquired by the laser range-finder. It starts with a set of triangles in image space that are merged into larger regions based on residual error from a best fitted plane to the corresponding 3D data points. We used the same observation that planar regions are often bordered by intensity discontinuities (edges in the image), for generating the starting triangular mesh for the region growing algorithm. The edges in the *region adjacency graph* are weighted with the residual error of a plane robustly fitted to the union of the 3D points corresponding to the adjacent regions R_i,R_j that share that edge.

$$E_{i,j} = \sum_{\mathbf{X}_k \in R_i, R_j} \alpha_k (\mathbf{n}^T \mathbf{X}_k + d)^2 \tag{4.22}$$

where $\mathbf{X}_k$ are corresponding 3D laser points for the regions and α_k is a weight factor (see the paragraph about plane fitting). Larger regions are grown from the initial mesh by merging, at every step, adjacent regions that produce the smallest error, and the algorithm stops when a threshold on the total number of regions is reached. At the end the 3D points in each region are projected to the corresponding fitted plane. Figure 4.19 illustrates the plane detection algorithm on a segment of the panoramic image, with the original image, detected edge segments, mesh triangles, and the final planar regions.

For determining plane equations, we used a weighted least square approach. It is well known that even a few outliers present in the data set can cause an erroneous estimation of the fitted plane. A robust fitting algorithm first estimates a plane as described in Section 4.3.2 (solving equation 4.18), and then assigns weights to the points depending on their residual:

$$\alpha_k = 1/(\mathbf{n}^T \mathbf{X}_k + d)^2$$

The plane is re-estimated by solving the weighted least square problem:

$$\min \sum_{k=1}^{N} \alpha_k (\mathbf{n}^T \mathbf{X}_k + d)^2 \tag{4.23}$$

In practice the points that have residuals above some threshold are eliminated by setting α to 0 and the plane is re-estimated with the remaining ones with $\alpha = 1$.

4.4.4 Model Vertical Lines

The incremental localization algorithm (see Section 5.3), designed for the image-based map described in this section, matches 3D vertical line features from the

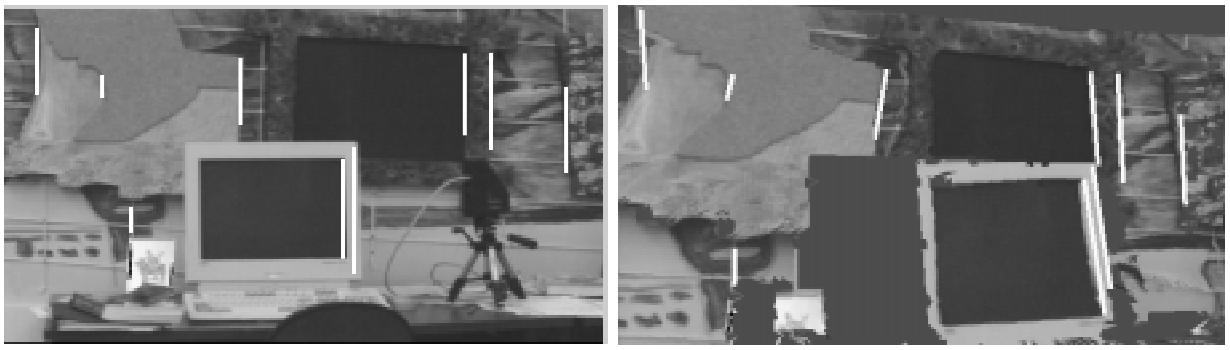

(a) (b)

Figure 4.20: Model vertical lines: (a) Vertical edge segments in the panoramic mosaic; (b) 3D lines rendered on top of the model;

model with detected vertical edges in the current image. To calculate the model lines' parameters, vertical edges are first detected in the panoramic model, and then, the line equation is calculated from the corresponding 3D points of the registered range data. There are three major categories of discontinuities in 3D that can generate an edge in the image:

(a) lines on the same surface that are discontinuities in intensity (color),
(b) surface discontinuities, and
(c) occluding contours on a smooth surface.

The lines in the first category will lie on the same plane, the ones in the second at the intersection of two planes and the ones in the third belong to a planar surface and it separates it from a background surface. For the last two cases, the estimated 3D line points might lie on two surfaces. A robust estimation of the line equation should consider the line as the intersection of the planar surfaces in case (b) and, in case (c), should first eliminate the points that belong to the background surface and estimate the line from the points lying on the foreground surface. The proposed algorithm simply eliminates the last two cases and considers only the first one where most line points belong to the same planar region.

A line in 3D can be represented in terms of a unit vector $\mathbf{v}$, that indicates the direction of the line and a point $\mathbf{d}$ on the line (see Section 4.1.3). Having N points on a line the line equation can be estimated in a similar way to the plane equation. Normalizing the points with respect to their centroid $\mathbf{X} = \frac{1}{N}\sum_{k=1}^{N}\mathbf{X}_k$, gives the zero mean points $\mathbf{A}_k = \mathbf{X}_k - \mathbf{X}$. The direction of the line is the eigenvector $\mathbf{v}_{max}$ corresponding to the biggest eigenvalue of the covariance matrix

$$\mathbf{\Lambda} = \frac{1}{N}\sum_{k=1}^{N}\mathbf{A}_k\mathbf{A}_k^T$$

The mean point was chosen as being the line point $\mathbf{d}_{max} = \mathbf{X}$. Each line is approximated with the closest vertical line. Figure 4.20 shows a rendering of the model with the vertical lines.

4.4.5 The Navigation Image-Based Map

Finally, to produce the composite model, the **panoramic model with depth from laser range-finder** consists of the panoramic image mosaic registered with a sparse, piece-wise planar 3D model, and a set of vertical line segments with corresponding 3D coordinates. This represents the second developed image-based navigation map that is used for robot navigation (Section 5.3) and predictive display (Section 5.4).

Chapter 5

Applications of the Panoramic Model in Mobile Robotics

This chapter presents some applications in mobile robotics of the panoramic image-based models described in the previous chapter. The general idea is that the model is used as a navigation map for robot navigation. The models are built off-line and the robot is carrying an on-board camera during navigation. The robot motion is assumed, planar which is reasonable for indoor environments where motion takes place on the floor, so the robot location is represented by a 2D translation and a rotation angle. The first few applications deal with the problem of robot localization. Section 5.1 presents an algorithm that uses two panoramic image mosaics to globally localize a robot that travels in the proximity of the models. The second application, described in Section 5.2 uses the "stereo" panoramic model from Section 4.3 for a global localization algorithm that uses a view taken with the same trinocular sensor. The last localization algorithm, described in Section 5.3 is designed for the panoramic model enhanced with depth acquired by the laser range-finder. This time we developed an incremental localization algorithm that is integrated into a predictive display system for a remote robot environment in Section 5.4. In general, the robotic applications demonstrate the applicability of the proposed panoramic image-based model in traditional tasks for mobile robotics (localization) integrated with less usual applications, like the predictive view generator, that make use of the image content of the model. The image-based map has to be regarded as an alternative to the traditional geometric (feature-based) maps and has the advantage over traditional geometric maps of being detailed but easily acquired and interpreted by an average user when controlling the robot. The data correspondence problem is also solved using image information.

5.1 Global Localization with Two Calibrated Panoramic Mosaics

Assume two panoramic image-based models constructed at known locations in a common environment, displayed as the two cylinders with the texture map of the environment in Figure 5.1. The global localization algorithm presented in this section calculates, from a single planar image captured by the robot, the position and orientation of the robot with respect to either one of the two panoramic models. The vertical axes of the two panoramic models are aligned and parallel to the image plane of the robot's camera. The localization algorithm uses a set of corresponding vertical lines as features for the two panoramic models and the planar image. Note that the vertical lines remain vertical when projected on a cylinder and are not transformed into curves like horizontal or arbitrary lines. The correspondence problem is not considered in this application, but is assumed that corresponding features already exist. For the experiments, the process of detecting corresponding lines is performed manually.

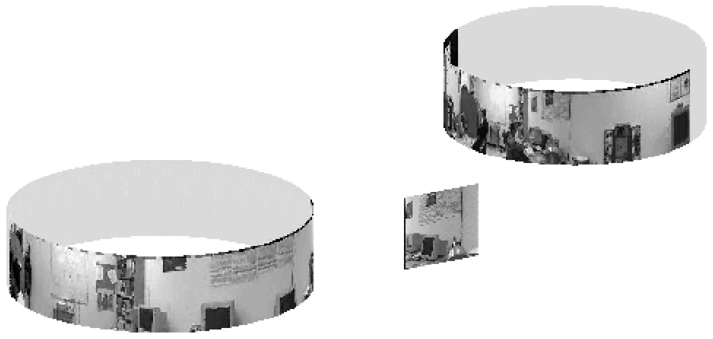


Figure 5.1: Two panoramic models and the planar image to be localized, each sowing 180° of the view.

5.1.1 Calibration of the Two Cylindrical Models

The localization algorithm requires the the exact position and orientation (starting angle) of the panoramic images with respect to the world coordinate system. For simplicity and without loss of generality, the center of first panoramic image was chosen as the origin of world coordinate system. We used McMillan's algorithm [106] to compute the relative position of the second cylinder and the rotational offsets which align the angular orientation of the two cylinders to the world coordinate system.

5.1.2 Localization Algorithm

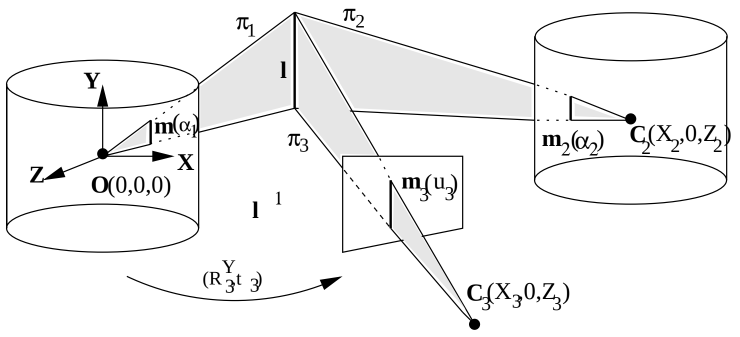


Figure 5.2: Matching lines in image-based models and planar image

The localization algorithm, depicted in Figure 5.2, uses pairs of vertical lines in the two panoramas and current robot view. The vertical line features are specified using only one coordinate - the *horizontal coordinate*, u, for the planar image and the *azimuth angle* , α, for the cylindrical models. The length of the line feature was not considered because this might not be a robust constraint due to occlusions.

Consider three corresponding lines $\mathbf{m}_1(\alpha_1)$, $\mathbf{m}_1(\alpha_2)$, and $\mathbf{m}_1(u_3)$ as depicted in Figure 5.2. These lines together with the centers of projection $\mathbf{O}, \mathbf{C}_2, \mathbf{C}_3$ determine three planes π_1, π_2, π_3 that should intersect on the same line $\mathbf{l}$. The normals to these planes, expressed in the normalized local coordinate systems, are given by

$$\begin{aligned} \mathbf{n}_1 &= [\cos\alpha_1, 0, \sin\alpha_1]^T \\ \mathbf{n}_2 &= [\cos\alpha_2, 0, \sin\alpha_2]^T \\ \mathbf{n}_3 &= [1, 0, u_3'] \end{aligned} \tag{5.1}$$

where $u_3' = \frac{u_3 - u_c}{a_u}$ is the normalized coordinate of u_3 considering the camera model given by (4.13). The projective coordinates of the normals in the coordinate system of the first cylindrical model are [47],

$$\mathbf{n}_1^p = \begin{bmatrix} \mathbf{n}_1 \\ 0 \end{bmatrix} \quad \mathbf{n}_2^p = \begin{bmatrix} \mathbf{n}_2 \\ \mathbf{t}_2^T \mathbf{n}_2 \end{bmatrix} \quad \mathbf{n}_3^p = \begin{bmatrix} \mathbf{R}_3^Y \mathbf{n}_3 \\ \mathbf{t}_3^T \mathbf{R}_3^Y \mathbf{n}_3 \end{bmatrix} \tag{5.2}$$

where $\mathbf{t}_2 = [X_2, 0, Z_2]$ is the displacement between the position of the second model and origin of the coordinate system, and $\mathbf{t}_3 = [X_3, 0, Z_3]$, $\mathbf{R}_3^Y(\theta)$ are the displacement of the current position and orientation of the robot with respect to the world coordinate system.

The planes π_1, π_2, π_3 should intersect on the same line. This is equivalent to the algebraic condition that the 4×3 matrix, formed with the normals $\mathbf{n}_1^p, \mathbf{n}_2^p, \mathbf{n}_3^p$ as columns, is of rank 2 [47]. This implies that all 3×3 determinants extracted from the matrix are equal to zero. For vertical planes, the second row from the matrix is zero, so we have only one determinant left,

$$D(\alpha_1, \alpha_2, u_3') = \begin{bmatrix} \cos\alpha_1 & \cos\alpha_2 & \cos\theta - u_3'\sin\theta \\ \sin\alpha_1 & \sin\alpha_2 & \sin\theta + u_3'\cos\theta \\ 0 & A & B \end{bmatrix}$$

where

$$A = -X_2\cos\alpha_2 - Z_2\sin\alpha_2$$

$$B = -X_3(\cos\theta - u_3'\sin\theta) - Z_3(\sin\theta + u_3'\cos\theta)$$

θ, X_3, Z_3, can be found by minimizing the following function with $N \geq 3$ triplets of corresponding lines:

$$\min_{X_3, Z_3, \theta} \sum_{i=1}^{N} \left(D(\alpha_1^{(i)}, \alpha_2^{(i)}, u_3^{\prime(i)}) \right)^2$$

This minimization problem is solved by the Levenberg-Marquardt non-linear minimization method [128]. Experimental results with both synthetic and real data are presented in the following sections.

5.1.3 Experiments with Synthetic Data

To verify and evaluate the robustness of the algorithm, we tested its sensitivity to noise. The noise can be caused by errors in selecting corresponding lines, by image quantization error, and by errors in model construction or calibration. 25 space vertical lines were selected and projected on two cylinders. Both cylinders have a radius equal to the focal length. The first cylinder shares the origin with the world coordinate system. The second one is three meters away from the first along the X axis. The space vertical lines were also projected on a planar image, using the calibrated camera model. To simulate erroneous line selection, noise uniformed distributed in the interval $(-w, w)$ (in pixels) was added to the coordinates of the line features in the cylindrical models and planar image. Then the algorithm was applied to recover the position and orientation of the third image captured by a robot between the two cylinders. For noise level $w = 3$ pixels the error was about 6.6 mm for position estimation, and 0.2° for orientation. The experiment was repeated for different numbers of lines, and the results show that the localization error decreases if the number of lines increases and, it stabilizes for more than 10 triplets of line features (see Figure 5.3).

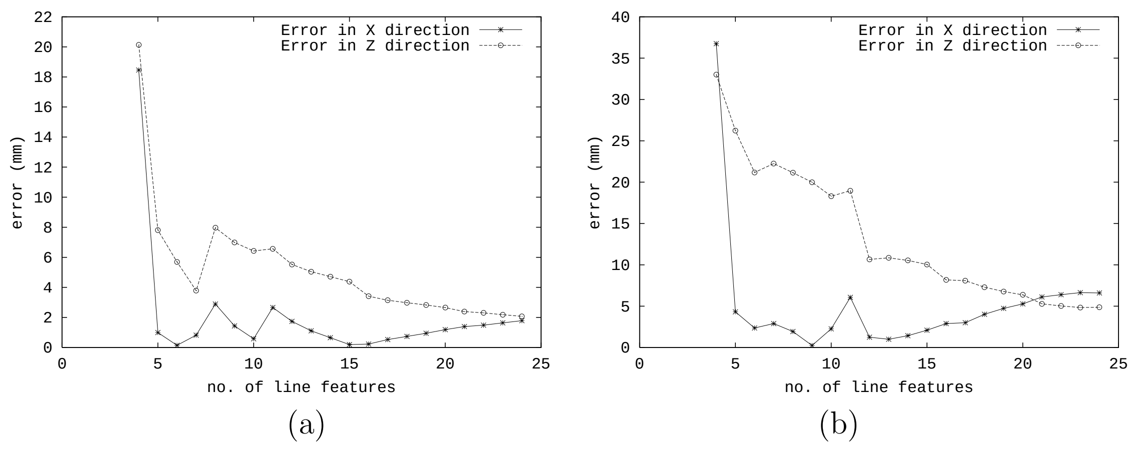


Figure 5.3: Variation of localization error error in X and Z directions with the number of triplets of line features; (a) without noise, (b) noise level $w = 3$.

5.1.4 Experiments with Real Data

For evaluating the algorithm with real data, we first built two panoramic cylindrical images as described in Section 4.2. The data acquisition setup was formed from a tripod and a pan-tilt unit on top of the tripod. For simplicity, the panoramas sampled only half of the environment, or 180° of each cylinder. The distance between the cylindrical models was about three meters. Then, the exact location and starting angles for the cylindrical models were computed using the method

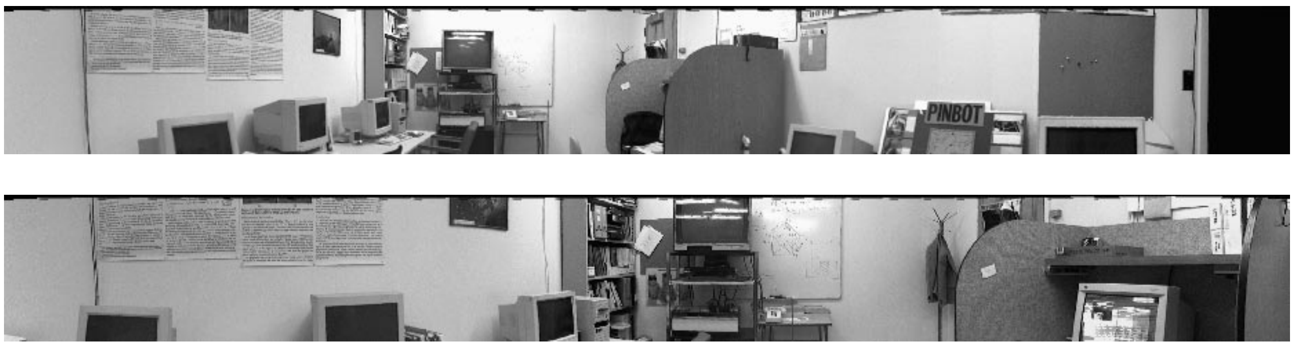


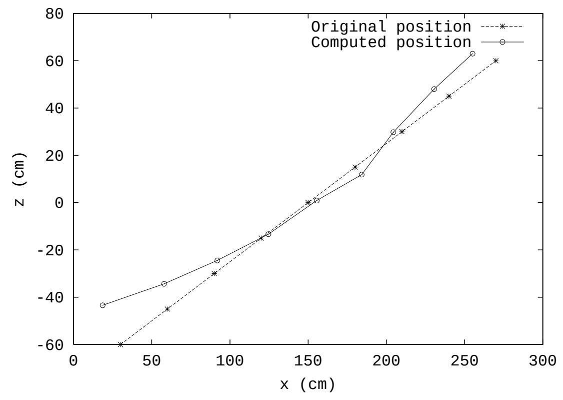
Figure 5.4: Two cylindrical panoramic images (each formed using 25 planar images sampling 180°)

described in Section 5.1.1. The calibrated distance and orientation of the cylinders are $\mathbf{C}_2 = (3000.04, 0, 3.05E-5)$(mm), and the rotational offsets $\beta_1 = -21.88°$ and $\beta_2 = -69.70°$. The results are shown in Figure 5.4, where the cylindrical images are unrolled.

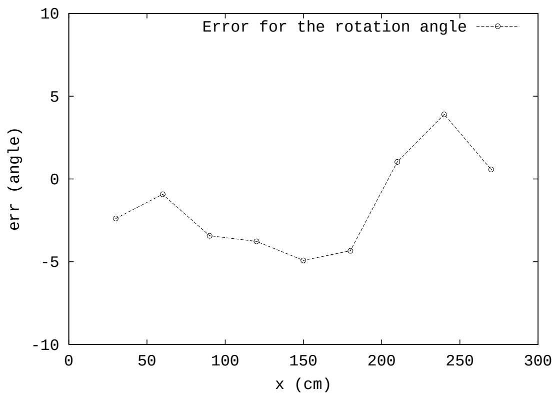
Figure 5.5: Planar image from in between the two panoramic images with the line features superimposed

Next, the localization algorithm was applied for images at nine locations between the two cylindrical models. The origin of the coordinate system was chosen at the origin of the first model, and the X axis along the line that connects the centers of the cylinders. Figure 5.5 shows one typical image. Figure 5.6(a) shows the actual and the estimated position, and Figure 5.6(b) shows the error in estimated angle for each position. The actual location of the robot was measured using a meter stick for the position, and a pan tilt unit for the rotation angle. The position error is measured in centimeters, and the rotation error in degrees. The average position error was around 5.5 cm in X and Z directions, and 3.5° for the rotation angle.

In addition to positions between the cylinders, we also tested the localization algorithm with images taken from other positions. The accuracy of the proposed method depends mostly on the viewing direction, and much less on the position of the robot. When the planes π_i generated by the line features from the current image and from one of the cylindrical models are very close to each other, the



(a)



(b)

Figure 5.6: (a) Recovered positions from the 9 images. The first panoramic model is situated at $(X, Z) = (0, 0)$ and the second one at $(X, Z) = (300, 0)$; (b) Error for the recovered orientation. The X coordinates show the position of the planar image regarding the first panoramic model.

localization error increases. This happens, for example, when the robot is close to the location of one of the models and the line features belong to objects that are far away from the camera. In comparison to an earlier point-based algorithm [27], however, this new line-based algorithm performs much better.

5.2 Global Localization using Stereo Panorama

The panoramic model with depth from stereo and extracted planar patches from Section 4.3 is used in a global localization algorithm designed for a robot that is carrying the same trinocular sensor. The robot's position and orientation with respect to the model's coordinate system are calculated from the current image and its corresponding disparity map.

5.2.1 Feature Selection

Three types of features were considered for the localization process - *planar patches*, *vertical lines*, and *points*. First, planar patches are extracted from the current image using the same segmentation algorithm described in Section 4.3.2. The robot's position and orientation are computed from pairs of corresponding features in the model and current image. Matching points and lines is a very difficult problem in computer vision especially when there is significant difference in scale between corresponding features. The proposed algorithm overcomes this problem by first matching the planar patches and then matching vertical lines that belong to corresponding patches. For point features, the corresponding pairs were manually extracted. Plane equations were extracted using depth information. From the plane equations, the 3D parameters of lines and points are calculated by intersecting image features with the corresponding plane. In this way the algorithm becomes more robust to errors present in the depth data.

All variations of the localization algorithm use first a linear method, and then the initial results are refined using non-linear minimization. The non-linear minimization problem was solved using a Levenberg-Marquardt non-linear minimization algorithm [128]. The following subsections briefly describe each localization algorithm.

We denote by $\mathbf{t} = [X_t, 0, Z_t]^T$ the robot's position and by $\mathbf{R}_y$ its orientation. Assuming planar motion, $\mathbf{t}$ is a translation in (X, Z) plane and $\mathbf{R}_y$ is a rotation about Y axis, both relative to model's coordinate system. The center of the panoramic image was chosen as the origin of the coordinate system, the Y axis along cylinder's axis, and Z axis pointing the starting angle in the cylindrical model.

5.2.2 Planar Patch Features

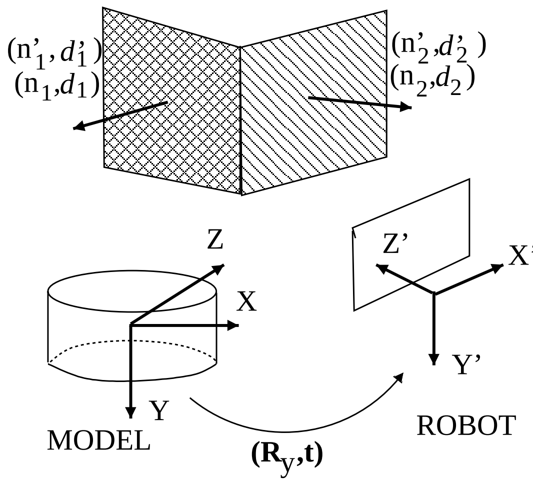


Figure 5.7: Localization using planar patches

Consider a plane that has parameters in the model coordinate frame $(\mathbf{n}, d)$ and in the image coordinate frame $(\mathbf{n}', d')$ (see Figure 5.7). The equations of this plane in the two coordinate frames and their relationship are the following:

$$\begin{aligned} \mathbf{n}^T\mathbf{P} + d = 0 \\ \mathbf{n}'^T\mathbf{P}' + d' = 0 \end{aligned} \tag{5.3}$$

and,

$$\mathbf{P} = \mathbf{R}_y\mathbf{P}' + \mathbf{t} \tag{5.4}$$

By substituting Equation (5.4) into the plane equations and comparing them we get the following constraints:

$$\begin{aligned} \mathbf{n} &= \mathbf{R}_y \mathbf{n}' \\ \mathbf{n}^T \mathbf{t} + d - d' &= 0 \end{aligned} \tag{5.5}$$

The rotation $\mathbf{R}_y$ can be computed from the first constraint (rotation about Y axis), but for the translation $\mathbf{t}$ we need another plane non-parallel with the first. So for completely localizing the robot, at least two non-parallel planar patches are needed. A least square algorithm was used to solve this problem if more than two planar patches are available. The results are further refined by minimizing

$$E = \sum_{i=1}^{N} \mathrm{dist}(\mathbf{n}, \mathbf{n}') + \sum_{i=1}^{N} |d - d'|$$

where N is the number of planar patches, and $\mathrm{dist}(\mathbf{n}, \mathbf{n}')$ is the distance between the normal vectors $\mathbf{n}, \mathbf{n}'$.

5.2.3 Vertical Line Features

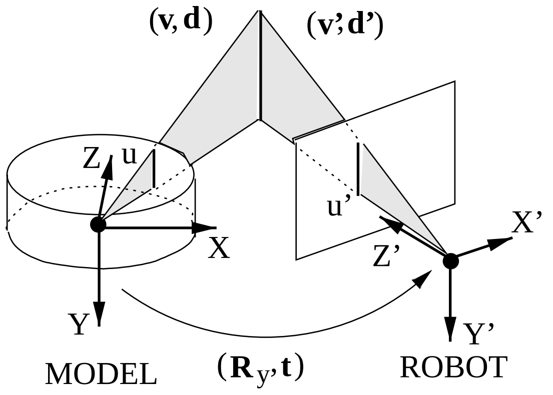


Figure 5.8: Localization using vertical lines

A 3D line can be represented in terms of a unit vector $\mathbf{v}$, which indicates the direction of the line, and a vector $\mathbf{d}$, which represents a point on the line that is closest to the origin [149]. In the case of vertical lines, $\mathbf{v} = (0, 1, 0)^T$, and $\mathbf{d}$ is the intersection of the line with the horizontal plane. A vertical line in the image plane can be characterized by its u coordinate.

Let's consider a vertical line that has parameters $(\mathbf{v}, \mathbf{d})$ in the model coordinate space, and $(\mathbf{v}', \mathbf{d}')$ in the camera coordinate space (see Figure 5.8). Denote by u and u' the parameters of the projection of this line on the model surface and image plane, respectively. As mentioned before, u and u' are used to compute the 3D equations of the lines, by intersecting the plane determined by the line projection with the corresponding planar patch. The relation between the two representations is the following:

$$\begin{aligned} \mathbf{v} &= \mathbf{v}' = (0,1,0)^T \\ \mathbf{d} &= \mathbf{R}_y\mathbf{d}' + \mathbf{t} \end{aligned} \tag{5.6}$$

The first equation does not bring any constraints on the robot position and is always true considering the way we extract line's equation (intersect two vertical planes). Using the second equation, which provides two constraints on $\mathbf{R}_y$ and $\mathbf{t}$, the robot can be localized with two or more corresponding vertical lines. The initial estimates are further refined by minimizing the distance between the projections of the estimated lines and the real ones.

$$E = \sum_{i=1}^{N}(u'_i - u''_i)^2$$

where N is the number of lines, and u''_i is computed by projecting $\mathbf{d}'_i$ on the image plane. In this case

$$\mathbf{d}'_i = \mathbf{R}_y^T(\mathbf{d}_i - \mathbf{t})$$

.

5.2.4 Point Features

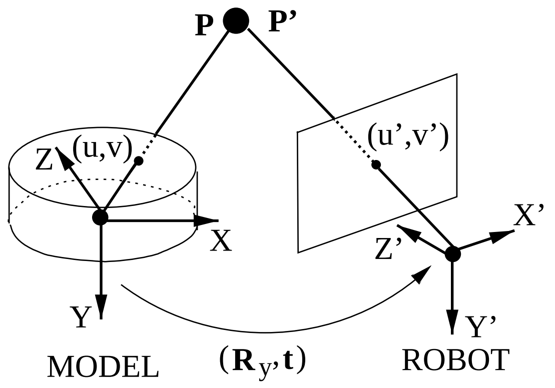


Figure 5.9: Localization using points

The point-based localization algorithm is very similar to the line-based localization algorithm. If $\mathbf{P}$ and $\mathbf{P}'$ are the 3D point coordinates in the model coordinate system and camera coordinate system, respectively (see Figure 5.9), the geometrical constraint between the two representation is the following:

$$\mathbf{P} = \mathbf{R}_y \mathbf{P}' + \mathbf{t} \tag{5.7}$$

With two or more corresponding points, the position and orientation of the robot can be calculated using linear methods. The non-linear algorithm minimizes the distance between the projection (u'', v'') on the image plane of the estimated 3D point

$$\mathbf{P}' = \mathbf{R}_y^T(\mathbf{P} - \mathbf{t})$$

and the real feature point (u', v'). The error function is

$$E = \sum_{i=1}^{N} \sqrt{(u'' - u_i')^2 + (v_i'' - v')^2}$$

5.2.5 Matching Planes and Lines

For the above localization algorithms to work, corresponding features must be identified in the image-based model and the current view. We developed a matching algorithm that computes the corresponding planar patches in the current image and the model. The algorithm first performs a *global registration* to find the position in the cylindrical model correspondent to the middle of the image. Then it matches patches that are around that position (within 300 pixels left or right considering that the cylindrical image width is about 1800 pixels). The global registration correlates a window from the middle of the image with the model.

All possible matches are generated and a score is computed for each candidate. Some of the patches from the model are occluded or simply not selected in the current image and vice versa, so not all the patches from the image or model should appear on the match. The *correlation score* is based on the difference in intensity between matching patches, relative distance between adjacent patches in the model and image, and number of patches present in the match. If $(\pi_i, \pi_i'), i = \overline{1, n}$ is a candidate match, where π_i represents a patch in the model, π_i' the corresponding candidate patch in the current image, and n is the number of patches in the match, the correlation score is:

$$score = \Delta int + dist + \frac{4}{n+1}$$

The difference in intensity can be computed as

$$\Delta int = \frac{1}{100n} \sum_{i=1}^{n} |\overline{I}_i - \overline{I}_i'|$$

where $\overline{I}_i$ $(\overline{I}_i')$ represents the average intensity in π_i (π_i'). For computing the relative distance between adjacent patches, both distance in the image plane and in 3D space is considered. Correlation is not included in the matching score because differences in scale between matching patches and occlusions can cause the algorithm to fail. Denoting by (u_i, v_i) $((u'_i, v'_i))$ the centroid of patch π_i (π'_i) in the image plane and by $(X_i, Y_i, Z_i)^T$ $((X'_i, Y'_i, Z'_i)^T)$ the centroid in the 3D space, the relative distance is:

$$\frac{1}{n-1}\left(\sum_{i=1}^{n-1} \frac{|du_i - du'_i| + |dv_i - dv'_i|}{\sqrt{du_i^2 + du'^2_i + dv_i^2 + dv'^2_i}} + \sum_{i=1}^{n-1} \frac{|D_i - D'_i|}{|D_i + D'_i|}\right)$$

where

$$\begin{aligned}
&du_i = u_i - u_{i+1}\\
&dv_i = v_i - v_{i+1}\\
&D_i = \sqrt{((X_i - X_{i+1})^2 + (Y_i - Y_{i+1})^2 + (Z_i - Z_{i+1})^2)}
\end{aligned}$$

and the same for du'_i, dv'_i and D'_i.

From corresponding planar patches, corresponding vertical lines are detected by matching the left and right borders of the planar patches. For matching a pair of vertical lines, the algorithm first computes their gradient in a window along the line to see if they are real edges. Then, the middle position of a pair of lines is correlated and decided if it is a match or not based on the correlation score.

We have tested this matching algorithm for about 20 images in different positions and orientations. The algorithm automatically detects matching planar patches in 80% of the cases. It may fail especially when there is significant difference in scale between the model and the image. If the column corresponding to the middle of the image is manually selected, the rate is increased to 90%. Having correct corresponding planar patches, the algorithm successfully detects corresponding vertical lines.

5.2.6 Experimental Results

For evaluating the model accuracy and the performance of the localization algorithm, we took images in seven random locations in the same room in which the panoramic model from Figure 4.8 was taken. Figure 5.10 shows the original locations and orientations, together with the positions of the planes that were used for localization. Planar patches and vertical lines between the panoramic model and the current image were automatically detected and matched. The matching algorithm usually gives 2 or 3 pairs of corresponding planes, and 3 to 6 corresponding lines. For the point-based algorithm, 6 or 7 points on the planar patches were manually chosen and matched.

Localization results using the three algorithms based on planar patches, lines and points, are summarized in Table 5.1. The algorithms based on point and line features perform better than the one using planar patches. This can be expected because the points and lines equations were computed based on plane equations, so there are no additional errors added, and the positions in the image form an extra constraint. For the localization algorithm using planar patches, only 3D equations of the corresponding planes were used, but not the boundary location in

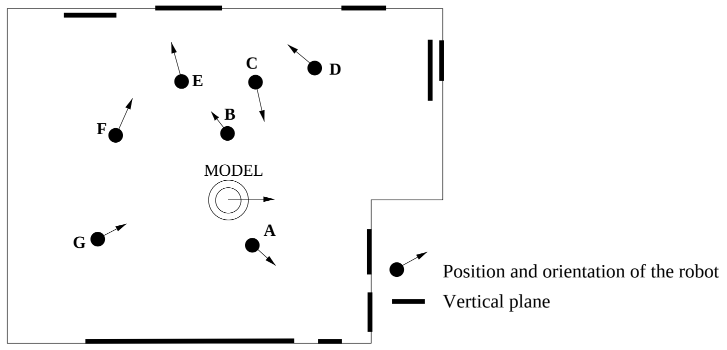


Figure 5.10: Robot positions for localization experiments.

the image that can be occluded or partially occluded. The dimension of the room is 10 m × 8 m. The average error for line and point-based algorithm is about 10 cm for position and 2.5° degrees for orientation. Both are acceptable for most navigation purposes. Ground through is estimated manually using a meter stick for distances and the pan unit for angles.

	Err. position (cm)			Err. orientation (deg)		
	Patch	*Line*	*Point*	*Patch*	*Line*	*Point*
A	70.5	25.6	6.4	20	0	4
B	51.8	4.0	2.0	15	1	1
C	19.1	5.3	17.1	7	4	5
D	30.0	12.2	18.0	9	2	2
E	50.0	12.5	9.0	6	3	1
F	25.5	9.8	3.6	8	4	2
G	52.5	3.0	30.4	3	1	1
Mean	42.7	10.3	12.3	9.71	2.1	2.2

Table 5.1: Error for recovered position and orientation

5.3 Incremental Localization with Depth Enhanced Panorama

This section presents a localization algorithm that uses the panoramic model with depth acquired by the laser range-finder described in Section 4.4. The localization problem involves finding the position and orientation of the robot with respect to the model (CCS) using the current robot view in terms of a 2D image.

An overview of the localization algorithm is presented in Figure 5.11. We perform an incremental localization where the current position is approximately

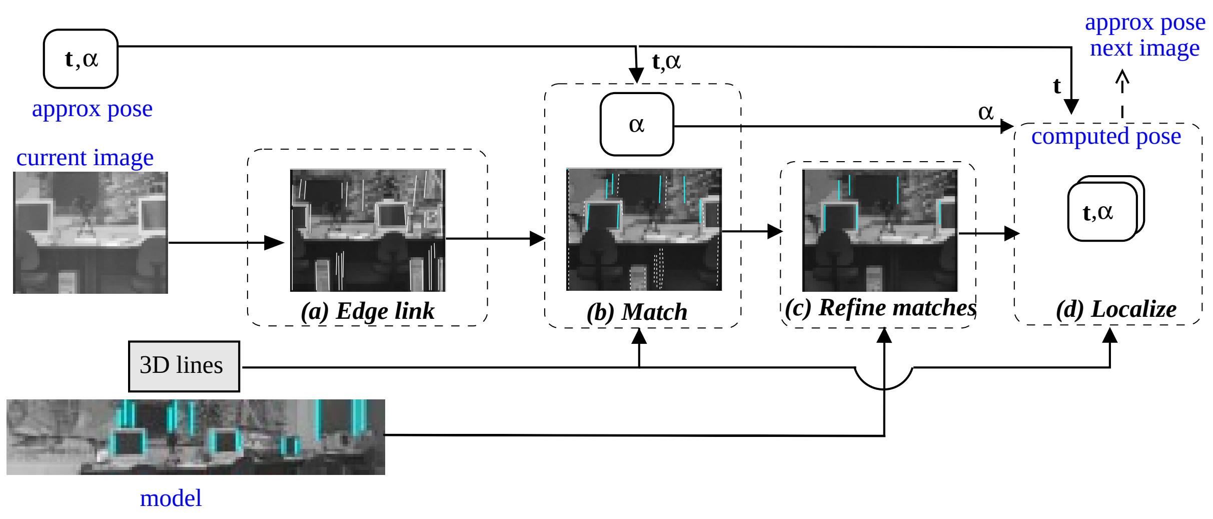


Figure 5.11: Overview of the localization algorithm. $\mathbf{t}$ denotes robot's translation and α robot's orientation: (a) Edge detection and linking (b) Compute best match and calibrate angle α based on Hausdorff distance (c) Refine matches and eliminate duplicates based on image information (d) Localization using model-image correspondence

known either from the previous position - assuming motion continuity - or from other sensors (odometry). An initial position and the height difference between the model location and the robot have to be estimated at the beginning using, for example, manually selected corresponding feature points. The first step (a) is to detect vertical line segments in the current image using standard edge detection and linking algorithms. The next step (b) is the angular calibration and detection of the best match based on the minimum Hausdorff distance between the projected model 3D lines and the detected vertical edges. The matching pairs of vertical lines are further refined and the duplicates are eliminated in step (c), based on the correlation score between the corresponding lines in the panoramic mosaic and the image line. Then, current position is updated from the corresponding model-image lines in step (d). The incremental localization approach assumes that the position is approximately known, limiting the matching search region, relative to a global localization approach. Next, some general notations are first introduced, followed by the general description that is used in the localization algorithm.

A 3D line is represented by its direction $\mathbf{v}$ and a point on the line $\mathbf{d}$ (Section 4.1.3). For a vertical line $\mathbf{v} = (0, 1, 0)^T$ and $\mathbf{d}$ can be chosen as the intersection of the line with the horizontal plane $\mathbf{d} = (X, 0, Z)^T$. Any point on the line can be expressed as $\mathbf{X}_l = (X, k, Z)^T$ where k is a real parameter that is restricted to an interval in case of a line segment. A vertical image line can be characterized by the column coordinate u, and a point on the line has the form $\mathbf{x}_l = (u, q)$, where

q is a parameter similar to the 3D case.

Let the unknown displacement between the model coordinate system (MCS) and the current image position (CIP) be $(R_y, \mathbf{t})$, where R_y is a rotation matrix over Y axis, and $\mathbf{t} = (t_x, H, t_z)^T$ (H is the height difference between the model and the robot). The vertical line points $\mathbf{X}_l = (X, k, Z)$ expressed in MCS are projected on the image using:

$$\mathbf{x}_l = K(R_y \mathbf{X}_l + \mathbf{t}) \tag{5.8}$$

where K is the camera matrix (equation 4.13). Using equation 5.8, the column coordinate of the projected line can be derived as:

$$u_{proj} = a_u \frac{X \cos\alpha + Z \sin\alpha + t_x}{-X \sin\alpha + Z \cos\alpha + t_z} + u_c \tag{5.9}$$

where α is the pan rotation angle.

In the following, assume that N vertical lines from the model are visible in the current view and M vertical edges are detected in the current image ($M > N$).

5.3.1 Angle Calibration and Matching Vertical Lines

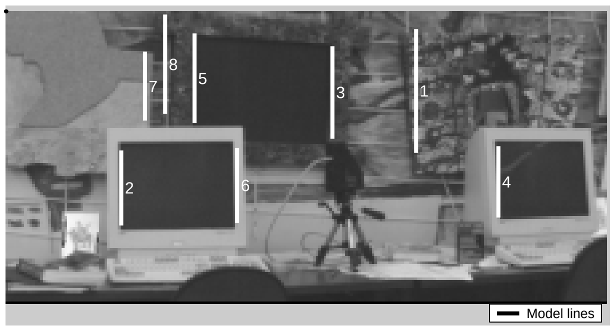
An error of a few degrees in the approximate orientation of the robot results in a big displacement between the projected lines and the corresponding detected image edges, making the matching process very difficult. The angular orientation is calibrated using a modified Hausdorff distance [67] between the projected and detected edges. The algorithm varies the orientation angle α in an interval of 10° around the given approximate position and compute the corresponding Hausdorff distance between the projected model lines $u^i_{proj}(\alpha)$, $i = 1 \ldots N$ (Equations 5.8 and 5.9) and detected vertical edges u^k, $k = 1 \ldots M$.

$$H(\alpha) = \sum_i K^{th}_{i=1 \ldots N} \{ \min_{k=1 \ldots M} d(u^i_{proj}, u^k) \} \tag{5.10}$$

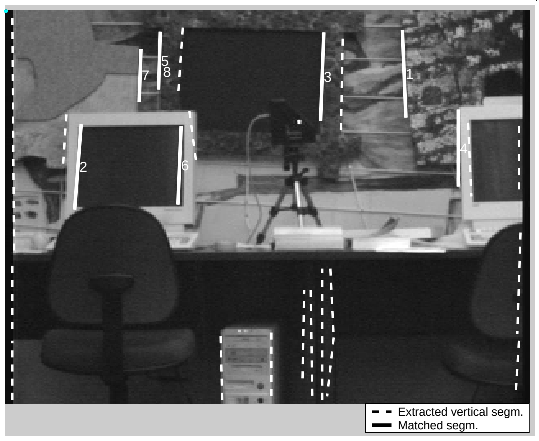
where $K^{th}_{i=1 \ldots N}$ denotes the K ranked values in the set of distances (one corresponding for each model line), $d(u^i_{proj}, u^k)$ represents the Euclidean distance between the center points of the line segments. Denote the edge segment that gives the minimum distance to a projected model line u^i_{proj} by u^{k_i}. The algorithm chooses the angle that has the smallest distance $H(\alpha)$. The corresponding set of line-edge pairs $(u^i_{proj}, u^{k_i}), i = 1 \ldots K$ from Equation 5.10, represents the desired matched model-image features. Figure 5.12 (b) shows the detected vertical edge segments, and the matched edges corresponding to the model lines in Figure 5.12 (a).

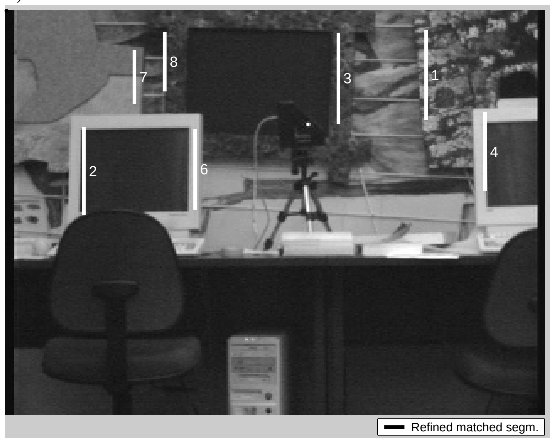
5.3.2 Refinement of Matches in Image Space

A further refinement step is considered to correct the bad matches and to eliminate duplicate matches. The algorithm takes advantage of the image information



(a)





(b) (c)

Figure 5.12: Illustration of matching algorithm: (a) model lines, (b) corresponding matched line after angle calibration (Section 5.3.1) and matched lines after refinement step (Section 5.3.2). Note that in the refinement step pair number 5 was eliminated as being a duplicate and the others are better aligned with the original lines in panorama. Numbers indexes lines in the model and show the matched pairs.

contained in the panoramic mosaic. This will give robustness to the matching algorithm, offering an appealing way to solve the data association problem, a very difficult issue in robotic mapping (see Section 3.1.1).

Consider a small patch Υ^i in the panoramic mosaic along each model line i that is present in a match pair. The refinement step searches for the best matching vertical line $\mathbf{x}_l^i = (u_l^i, q_l)$ in the current image, around the position of the original match $\mathbf{x}_l^{k_i} = (u^{k_i}, q)$. A small patch I^i (same size as Υ^i) is moved in a vicinity of the middle point of the line segment $\mathbf{x}_l^{k_i}$ and compared with Υ^i. Because correlation is not size-invariant and robust to occlusions, we define the matching criteria as being a combination of the correlation score and the intensity difference between left/right corresponding portions of the patches Υ^i and I^i (similar to the approach in Section 5.2.5). If the score is below a threshold the pair is eliminated. If there are more matches for the same model line, only the one with the biggest score is considered. Figure 5.12 (c) shows the refinement matches for the model lines in Figure 5.12 (a) . Note that pair number 5 was eliminated as being a duplicate and the others are better aligned with the original lines in panorama.

5.3.3 Localization using Vertical Line Segments

If L pairs of 3D model lines - 2D image edges - are available, the motion parameters (α, t_x, t_z) are computed by minimizing the displacement between the corresponding projected and detected lines.

$$(\alpha, t_x, t_z) = \min_{\alpha, t_x, t_z} \sum_{i=1}^{L} (u_{proj}^i - u_l^i)^2 \tag{5.11}$$

The non-linear least square problem is solved using Levemberg-Marquardt non-linear minimization algorithm.

5.3.4 Experimental Results

To evaluate the model accuracy and the performance of the localization algorithm, we took 26 images along a trajectory at positions that are 10 cm apart, and recovered their positions using the localization algorithm described in the previous sections applied to model from Figure 4.16. For the first position, we manually selected corresponding point features in the image-based model and the robot view at the position, then applied the point-based localization algorithm described in Section 5.2.4. This initialization step not only provided a reliable initial position of the robot for studying the incremental localization algorithm along the trajectory, but also recovered the difference in height, H, between the image-based model and the robot camera reference frame, which would remain constant throughout the experiments.

Subsequently, we performed the incremental on-line localization algorithm for the remaining 25 points. At each position, the location computed in the previous step is used as an approximate location for the current step. All the measurements were relative to the reference frame of the image-based model.

We measured the relative accuracy of the localization algorithm, in terms of the position error along the trajectory δ, assumed to be 10 cm apart, and the position error tangent to the trajectory, ρ. For the 26 positions, the two errors are found to 2.1 cm and 1.6 cm, respectively (see Table 5.2). Both of those errors are quite satisfactory. It is worth mentioning that errors are less where the 3D line features were more accurate.

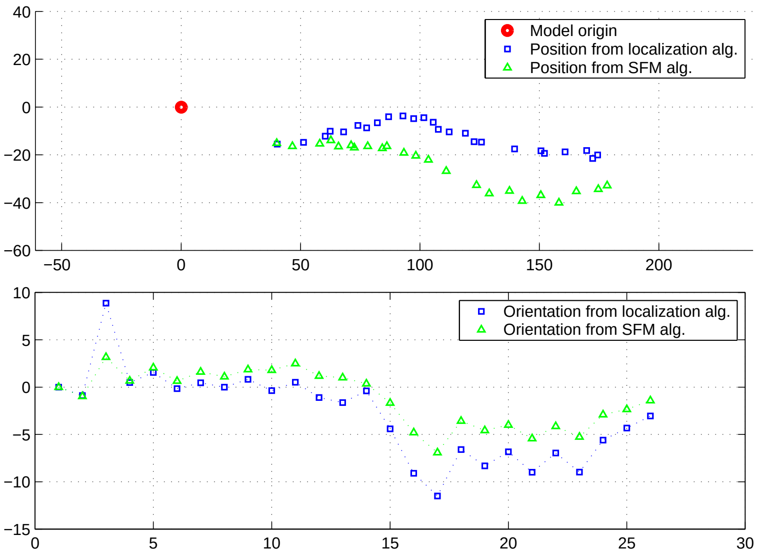


Figure 5.13: Recovered position (top) and orientation (bottom) for the model-based localization algorithm and a SFM point based algorithm

Having no absolute ground truth, we compared the performance of the proposed localization algorithm with the results from a point based structure from motion (SFM) algorithm (similar to the one presented in Section **??**). The feature detection and correspondence for the SFM algorithm was performed manually. Figure 5.13 plots the recovered trajectory (positions) (a) and orientations (b) for the two algorithms. The two algorithms are different in the sense that the SFM algorithm uses image information alone and the proposed model based algorithm uses the panoramic model composed from both image and laser range data. The similarity of the results proves the success of the proposed localization algorithm (compared to a classic algorithm). There is a consistent error in both measurements probably caused by some error in the model alignment. The accuracy of the SFM algorithm is superior to the accuracy of the proposed line based algorithm (Table 5.2 shows same measurements compared to relative ground truth about distance between sample images and deviation from line), but the automatic selection and matching of point features required by a robot navigation algorithm, is a very difficult problem in this case. For the proposed line based algorithm, both the 3D and image information contained the model was used to solve the data association problem (Section 5.3.1 and 5.3.2).

	No refinement	Refinement	SFM
Distance δ(cm)	3.8	2.1	1.6
Deviation ρ(cm)	2.9	1.6	0.6

Table 5.2: Localization accuracy for the localization algorithm (with/without match refinement) and a point based SFM algorithm: error in distance between consecutive positions δ(cm) (ground truth 10 cm) and deviation from straight line ρ(cm)

For illustrating the role of image information, used in the match refinement step (Section 5.3.2), compared to only 3D information, that is usually used in a traditional geometric based navigation approach, the recovered trajectories (first leg), in the two cases, were plotted in Figure 5.14. Note the errors in recovered position without refinement when a bad match is present. The numerical results in the two cases are presented in Table 5.2.

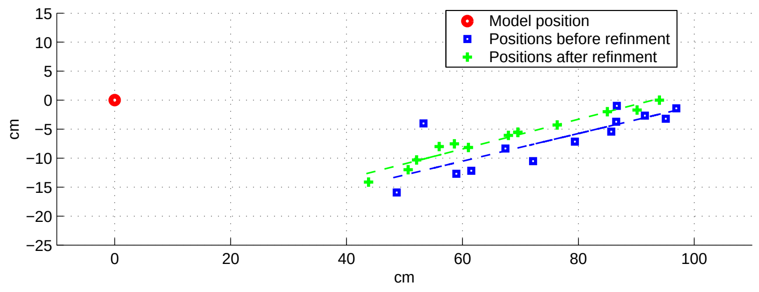


Figure 5.14: Recovered positions for the first leg of the trajectory (straight line) with/without refinement. Note the errors in recovered position without refinement when a bad match is present.

To examine the robustness of the algorithm with respect to the errors in the 3D line features of the model and the errors in 2D vertical line detection in the current image, different levels of uniform noise were added to the 3D lines and the manually selected feature points. We compute the average error between the reconstructed positions using the noisy data and the reconstructed position using noise-free data. Figure 5.15 plots the levels of perturbations in the 3D line and image line positions that will produce different errors in the estimated robot location. This result is useful to estimate the performance of the localization algorithm under varying sensor characteristics.

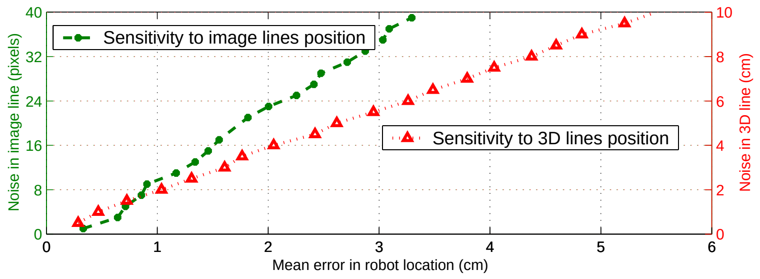


Figure 5.15: Sensitivity of the localization algorithm with respect to features position in image vs. 3D.

5.4 Mapping and Predictive Display for Remote Robot Environment

An important and challenging problem in tele-robotics is how to convey the situation at a distant robot site to a human operator. In real-world applications such as emergency response, unmanned exploration, and contamination cleanup, typically the remote scene geometry is unknown, and the data link between the robot and operator can have both delays and limitations on bandwidth. Hence, at the outset, the operator has not only little information about the remote scene, but also limited means of transmitting timely and high fidelity video and other data. Yet, research has shown that tele-operation performance is significantly degraded with delays of as small as 0.4 seconds [60]. This causes operators to lose direct intuitive connection between their control actions and effects at the remote site displayed in the video feedback.

Both quantitative and qualitative information from a remote tele-robotics scene can be improved through the use of geometric modeling. For quantitative robot localization, typically a *geometric map* is used as presented in Section 3.2. A different approach to geometric feature-based maps is the *appearance maps*, described in Section 3.3, that are created by “memorizing” the navigation environment using images or templates. One of the major drawbacks of these appearance-based maps is that robot motion is restricted to either a predefined route or positions close to the locations where the images were originally acquired. By contrast, in image-based computer graphics the objective is to generate models which represent the visual appearance of a scene from all view points. Chapter 2 presents a variety of techniques designed for modeling from images. In mobile robotics obviously different viewpoints need to be represented. However, current localization algorithms do not give precise enough pose to integrate the views from several positions into one lumigraph. We present an approach in-between image and geometry based modeling that uses the panoramic model with depth from laser (Section 4.4) for a predicted display system. The localization algorithm described in Section 5.3 is integrated with the system.

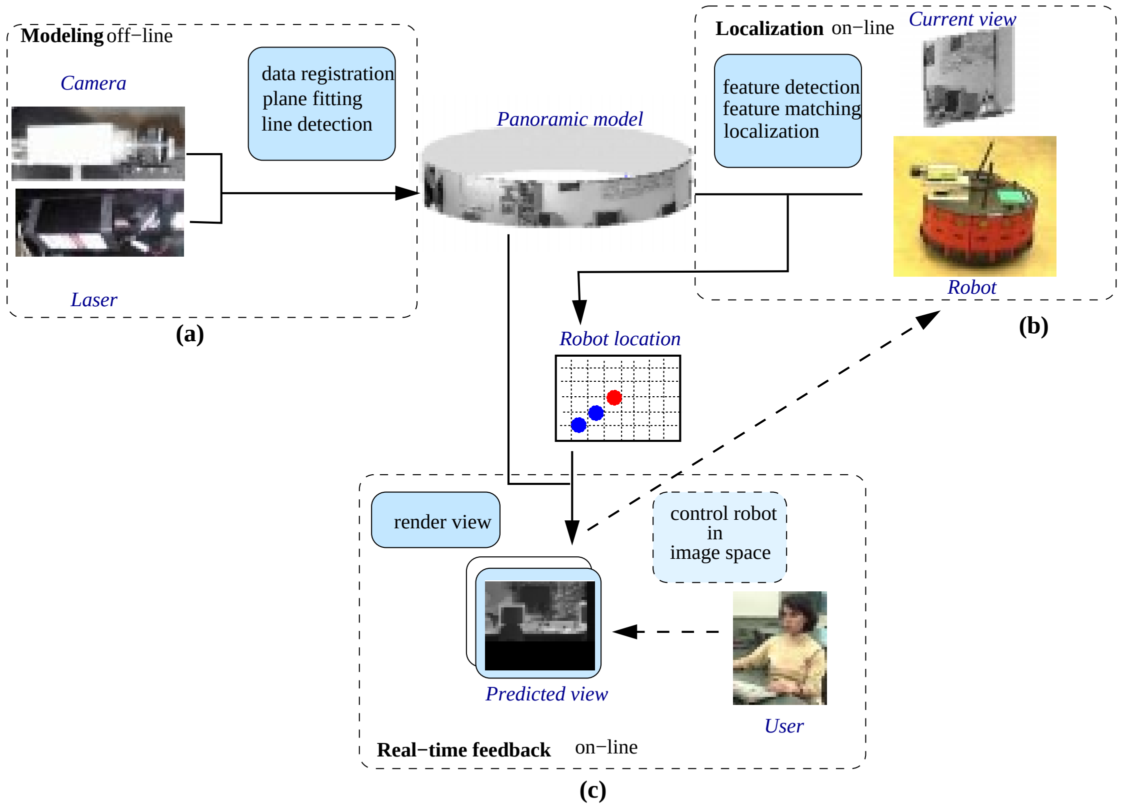


Figure 5.16: Overview of the navigation system with predictive display. The image-based panoramic model is acquired and processed off-line using a camera and a laser range finder (a). The on-line localization system matches lines features in the model with the ones extracted from the current view (b). The robot location is send to the remote site where a predictive view is immediately rendered (c). The remote user can control the robot using the image-based interface.

In predictive display two qualities of the model are essential: The ability to determine the correct view point (location) to render, and the support of high fidelity image rendering of that view point. Most current systems use geometric CAD models to support rendering and calibrated additional sensors (e.g magnetic or joint angle) to determine pose. For a review refer to [138]. However, this is impractical in unstructured environments often encountered in mobile robotics. A better method to align the predicted display is to register the remote camera image with the model [7]. A model free telepresence vision system, presented in [4], generates an approximate desired view by predicting the mouse movement in a panoramic image acquired by a sensor mounted on the robot. In our approach the model is obtained once through automatic sensing, then a single camera image is used to align the current view location with the model to support both localization and predictive display. An overview of our system is presented in Figure 5.16. Specifically, the model is formed by a panoramic image-based model augmented with a sparse set of 3D vertical line features (a) as described in Section 4.4. When the room map has been obtained, the incremental on-line localization algorithm from Section 5.3 is used to estimate to estimate the robot's position (b). After robot position has been obtained, it is sent to the remote location to generate a synthesized view using the same model (c) as described in Section 5.4.1. In this way, the operator who is controlling the robot has a user friendly interface as shown in the experiments, Section 5.4.2.

5.4.1 Predicted Scene Rendering

In tele-robotics a robot at a remote site is controlled by a human operator. Good operator and system performance depends in part on the ability of the tele-robotics system to visually immerse the operator in the remote scene. In particular, it is necessary to ensure that round trip delay between an operator motion command and the visual feedback received by the operator from the remote site must be minimized. It has been shown that longer delays prevent complex motion tasks, and operators adopt inefficient "move and wait" strategies to motion control [10]. While time delays are inherent in information transmission over a distance, the effect of the delays can be reduced by synthesizing estimated images based on our model, and showing this video for operator visual feedback. In the current implementation, we used the model to synthesize the current robot view from the acquired model ahead of the arrival of actual (delayed) scene video. Three components are needed to render a predicted view:

1. **A geometric model of the scene segment to be rendered.** We use the piece-wise planar model derived in Section 4.4.
2. **A viewpoint for the predicted image.** From our model we can derive the robot location as shown in Section 5.3, (showing current motion) and to

it add the operator motion command (predicting the outcome of the next action ahead of its completion).

3. **A texture image to warp onto the geometric model.** The predicted view can be textured either from the panorama or a previous camera image.

The image-based panoramic model with registered sparse range values is stored at both robot and tele-operator locations. We used as a geometric model of the scene the triangulation mesh formed by a constrained Delaunay triangulation with edge segments in the panoramic mosaic. This triangular mesh was the starting point of the region growing algorithm in Section 4.4.3 (see Figure 4.19 (c)). For each vertex in the panoramic image $\mathbf{x}_c$, we compute its corresponding 3D coordinate $\mathbf{X}$ (in model coordinate system) by interpolating the existing 3D laser points in its vicinity.

We developed an OpenGL hardware accelerated implementation that allows rendering in real-time. Given a viewpoint, the geometric model is projected in the virtual camera and textured with an image from the remote scene. The panoramic image contains the textures for all viewing directions, but from a single viewpoint. This provides a good approximation to textures also in nearby viewpoints. For rendering viewpoints far away from the model center, the panoramic texture has several drawbacks (e.g. significant quantization effects due to aliasing, occlusions, amplification of errors in the geometric model). A better alternative is to instead use the last received image from the robot site. Through the localization procedure, it is registered with the model and then re-warped into a new view. Since the motion of the real robot is continuous and smooth, most of the new view can usually be textured from a previous delayed image. Only during large rotational motions where the robot rotates with an angular speed approaching the transmission delay does the limited viewing angle of the real camera cause significant portions of the predicted view not to be textured. The process of generating the predictive view can be summarized as follows.

```
for each time step $t$
    robot site:
      (1) calculate robot location $(R_y(t), \mathbf{t}(t))$
      (2) send position to operator site
      (Asynchronously) Send robot camera image
    user remote site:
      (3) add current operator motion command
      (4) project scene model points $\mathbf{X}$
            $\mathbf{x}(t) = K(R_y(t)\mathbf{X} + \mathbf{t}(t))$
      (5) generate synthesized robot view with
            (a) texture from panoramic model image
            (b) or texture using delayed image from $t-1$
end for
```

In **Step 1**, the robot location (position and orientation) is calculated from the robot view in terms of a 2D image (as in Section 5.3). From corresponding pairs of line segments, we compute the robot orientation (pan angle) $R_y(t)$ and position on the floor plane (2D translation) $\mathbf{t}(t)$.

After the robot position has been calculated, it is sent to the remote site in **Step 2**, the current incremental operator motion command between $t-1$ and t is added in **Step 3**, and a predictive view can be generated by projecting the geometric model in the new location in **Step 4**.

In **Step 5**, texture mapping a geometric model assumes that it is decomposed in triangular planar surfaces. In a complete mesh, some of the triangles might not represent physical planes but are artifacts of occluding contours. In most cases, these appear as silhouette edges [104] where points from the object are connected with background points. To avoid this phenomenon, we eliminate all the triangles that (within a threshold) are parallel to the viewing direction.

Texturing the predicted view from the cylindrical panorama can be done directly, since the registration provides the texture coordinates for each model vertex. In the cases where a previous camera image $I(t-1)$ from a different view point is used the correct texture coordinates are obtained by projecting the geometric model using the robot location at time $t-1$, $(R_y(t-1), \mathbf{t}(t-1))$. In this way the delayed image $I(t-1)$ is warped into the new view location to generate a predictive view at time t. Assuming a continuous and smooth motion, we can generate realistic views in real-time along the entire robot trajectory.

5.4.2 Experimental Results

The localization algorithm and the predicted display are integrated into a common interface where the remote user can visualize both robot position on a 2D floor map and a see a predictive synthesized robot view (see Figure 5.17). In addition to forward predicting and rendering views one or several time steps ahead of the robot, using the model and panoramic image, the operator can also visualize entire simulated robot motion trajectories in the remote scene before executing the actual robot motion. The navigation system can be extended by offering the user the possibility to control the robot by dragging and clicking on desired locations in the rendered view. This will offer a natural and intuitive way to control the robot without having to interpret the geometric map.

To evaluate the model accuracy and performance of the localization algorithm with predictive display, we used the same data as in Section 5.3.4 applied to model from Figure 4.16. In evaluating the on-line forward prediction from delayed images, the 26 robot camera images along a trajectory were forward warped to viewpoint about 10 cm ahead on the trajectory and compared to a real image obtained at that same viewpoint. Figure 5.18 shows examples of rendered views along robot trajectory using texture from panoramic model (middle column) and previous (delayed) real image (left column). Notice that as mentioned in Section 5.4.1, in texturing

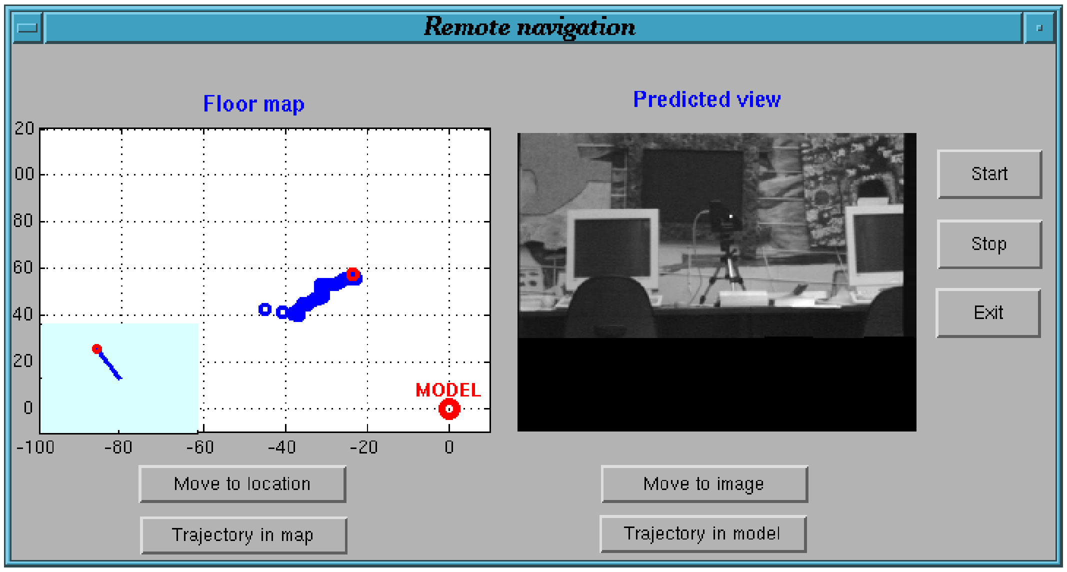
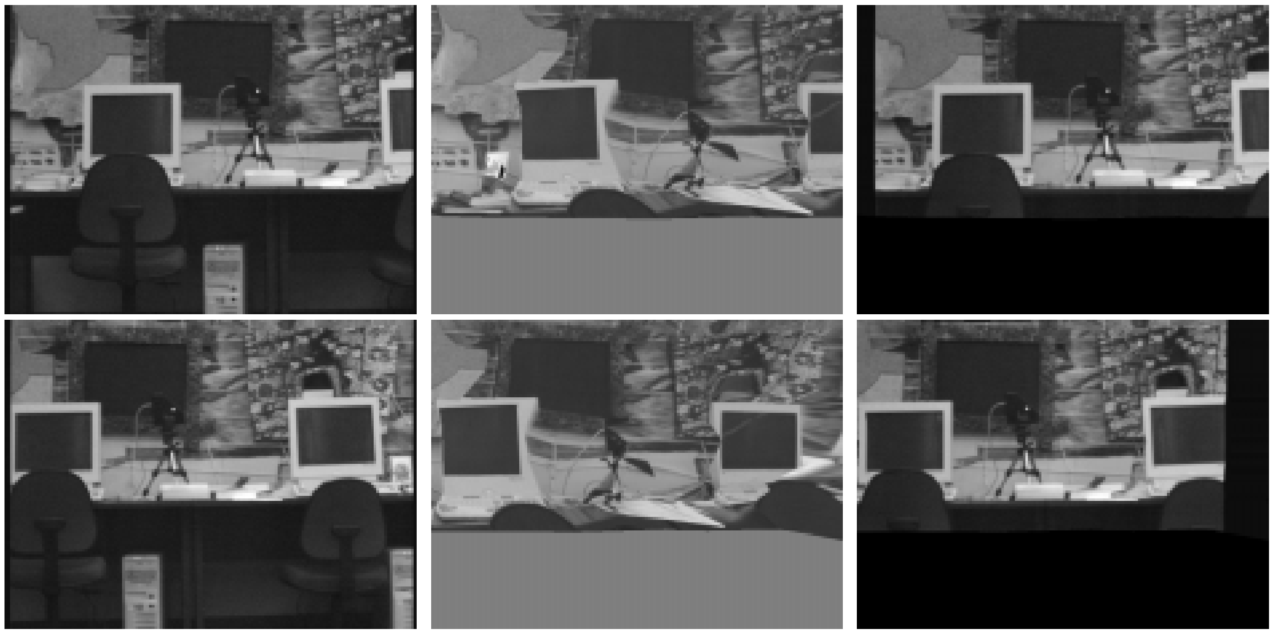


Figure 5.17: Tele-operator interface where the user can view a 2D floor map with robot position and the predictive robot view. The vector in the small left window shows robot orientation.



(a) texture from panorama (b) texture from previous image (c) ground truth (delayed image)

Figure 5.18: Examples of predictive views using different texturing

from the cylindrical panoramic image, any pan angle can be rendered, while when texturing from a delayed image, only the overlap in pan angle between the two views can be rendered. This accounts for the black stripe to the left or right of the predicted views in the middle column. The black stripe in the bottom is due to the cylindrical model being cut off at that level. Comparing the predicted with real views (in the right column), we notice that the rendering from delayed images produces better quality visual feedback than rendering from the panorama. This is to be expected because between two successive robot views there is only a small displacement, so distortions caused by texture or geometry errors are minor.

Chapter 6

Geometric Model with Dynamic Texture from Uncalibrated Image Sequences

This chapter presents the second type of image-based model, here developed using uncalibrated computer vision techniques. The model is formed from a sparse geometry computed using structure from motion techniques under either of two camera models - an affine (weak perspective) and a projective camera. For mobile robotics applications metric measurements are required in order to be able to give commands to the robot. For this, in both cases the structures are upgraded to metric using the stratified approach [48]. In a more general application (e.g. visualization, AR) the model can be less constrained. We present a comparison of geometric models for AR system in [23]. To compensate for the sparsity of the geometry and other unmodeled aspects such as lighting and nonplanarities of surface patches, we developed a technique based on a view dependent dynamic texture represented as basis of spatial filters. The first section introduces general multi-view geometry concepts that are used for building the geometric model, followed by a presentation of the computation for the two types of structures: affine and projective. The theory of dynamic textures is presented in Section 6.4 followed by a description of the image-based rendering system (Section 6.5). Next chapter presents the implementation details and an evaluation of the model from a graphics viewpoint as well as its mobile robotics' application in tracking and predictive display.

6.1 Multi-View Geometry

Researchers have made made significant progress in the past 10 years in developing the theoretical aspects of uncalibrated structure from motion. The goal is to acquire a realistic 3D model by freely moving a camera about a scene. This section will summarize some important aspects and problems in uncalibrated structure from motion. For a comprehensive and complete presentation refer to e.g. Hartley and Zisserman's book [59].

6.1.1 Non-Euclidean Camera Models

A camera maps points from the 3D space (object space) to 2D image plane. Section 4.1.1 introduced the projective Euclidean camera or calibrated projective camera that, in general, can be decomposed into the intrinsic parameters K and extrinsic parameters (camera position) - rotation R and translation $\mathbf{t}$.

$$P = K[R|\mathbf{t}]$$

In general, the 3×4 projection matrix P has 11 DOF. If the left 3×3 submatrix M is nonsingular the camera is called *finite*; if M is singular the camera is called *infinite*.

Some properties of the general projective camera $P = [M|\mathbf{p}4]$ can be derived without decomposing it into internal and external parameters. For example we can retrieve the camera center $\mathbf{C}$ as the 1D nullspace pf P ($P\mathbf{C} = 0$), for finite cameras:

$$\mathbf{C} = \begin{pmatrix} -M^{-1}\mathbf{p}4 \\ 1 \end{pmatrix}$$

and for infinite cameras:

$$\mathbf{C} = \begin{pmatrix} \mathbf{d} \\ 0 \end{pmatrix}$$

where $\mathbf{d}$ is the nullspace of M. This property provides an additional characteristic for finite/infinite cameras: in the case of a finite camera the center is an actual point, whereas for the infinite camera the center is a point in the plane at infinity. The principal ray (the ray passing through the camera center and which is perpendicular to image plane) can be expressed as:

$$\mathbf{v} = det(M)\mathbf{m}^3$$

where $\mathbf{m}^3$ is the last row of M.

Affine cameras

Affine cameras (or parallel projection cameras) is a class of infinite cameras of practical interest. An affine camera has the last row of the projection matrix of the form $(0, 0, 0, 1)$, they have an infinite center of projection so camera projection rays are parallel.

In general an affine camera can be written as:

$$P_\infty = K \begin{bmatrix} \mathbf{i} & t_x \\ \mathbf{j} & t_y \\ \mathbf{0}^T & d_0 \end{bmatrix} \quad R = \begin{pmatrix} \mathbf{i} \\ \mathbf{j} \\ \mathbf{k} \end{pmatrix} \quad \mathbf{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \quad d_0 = t_z \tag{6.1}$$

where d_0 represents the depth of the point along the principal direction and $\mathbf{i}, \mathbf{j}$ are the first two rows of the rotation matrix that related the position of the object with the camera pose.

The imaging geometry of an affine camera is depicted in Figure 6.1. It can be described as a two step projection. Consider a plane that passes through the world coordinate center and is parallel with the image plane. This plane can be thought as approximating the object, and for all the points on the plane the affine projection is equivalent to a perspective projection. The 3D point $\mathbf{X}$ is first projected on this plane to get $\mathbf{X}'$, and then $\mathbf{X}'$ is perspectively projected on the image plane. The deviation of the affine projection with respect to the projective projection can be expressed as:

$$\mathbf{x}_{aff} - \mathbf{x}_{persp} = \frac{\Delta}{d_0}(\mathbf{x}_{proj} - \mathbf{x}_0) \tag{6.2}$$

where $\mathbf{x}_0$ is the principal point and Δ is the distance from the point to the object approximation plane. From this we can see that the affine camera is a good approximation of the imaging geometry when:

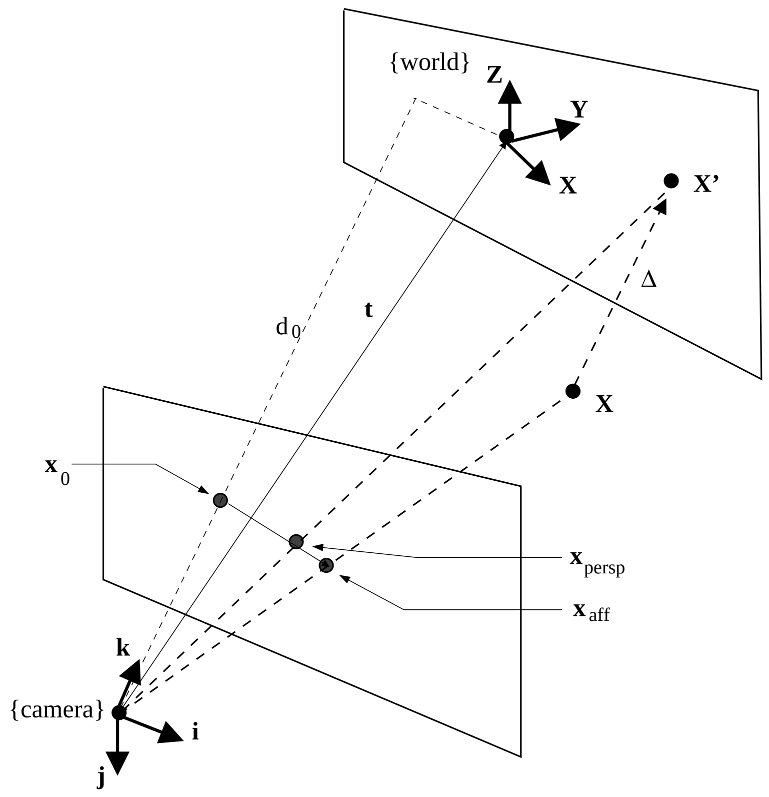


Figure 6.1: Affine camera approximation

- The depth relief (Δ) variation over the object surface is small compared to the average distance to the object (d_0).
- The distances between the object points and the principal ray are small.

A hierarchy of affine cameras

Different affine models can be derived starting with a very simple model and refining it to better approximate the imaging process. This is illustrated in Figure 6.2.

Orthographic camera. If the direction of projection is along Z axis, the camera matrix has the following form:

$$P_{orth} = \begin{bmatrix} \mathbf{i} & t_x \\ \mathbf{j} & t_y \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{6.3}$$

Weak perspective camera. A better approximation of the perspective camera can be obtained by first projecting orthographically on an approximate object plane and then projectively on the image plane. It is also called scaled orthographic

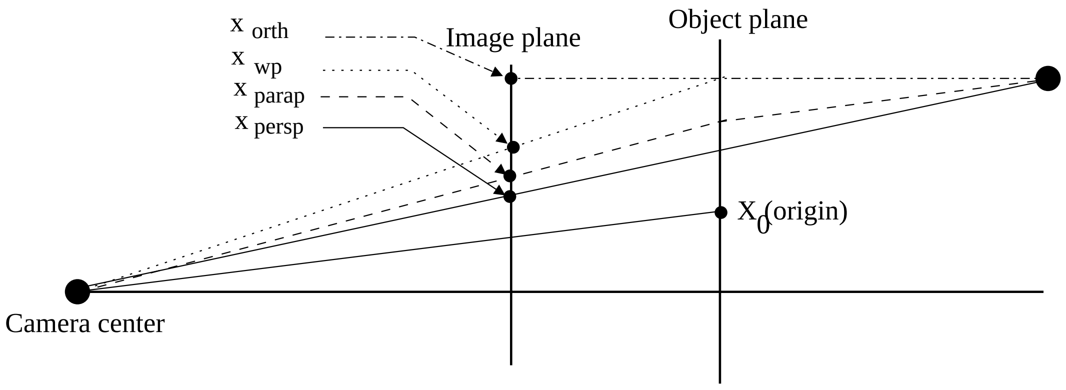


Figure 6.2: Different types of affine projection.

projection and the camera matrix has the form:

$$P_{wp} = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{i} & t_x \\ \mathbf{j} & t_y \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{6.4}$$

Paraperspective camera. The weak perspective camera is a good approximation of the perspective distortion only if the object is close to the principal axis. The paraperspective model compensates this problem by projecting the object point on the model plane with a direction parallel to the ray that projects the center of object coordinate system. It is not an affine camera but a first order approximation of the perspective camera. (The other models can be seen as a zero-order approximation.) The projection equations can be written as:

$$x_p = x_0 + \frac{\mathbf{i} - x_0\mathbf{k}}{t_z}\mathbf{X} \tag{6.5}$$

$$y_p = y_0 + \frac{\mathbf{j} - y_0\mathbf{k}}{t_z}\mathbf{X} \tag{6.6}$$

where (x_0, y_0) represents the image projection of the center of the object coordinate system.

6.1.2 Computation with Camera Models

In practical application we are interested in using camera models computationally to relate the three entities of 3D structure, 2D image projection and projection matrix. Three basic computational problems are:

1. *Resection:* Given a 3D structure and its image projection compute the camera pose (extrinsic parameters).

2. *Intersection:* Given two or more image projections and corresponding cameras compute a 3D structure giving rise to these images.

3. *Structure and Motion* or *Factoring:* Given only the image projections in two or more images find both the camera pose and 3D structure.

Resection

In Resection we are given 3D points $\mathbf{X}_1 \dots \mathbf{X}_n$ and their image projections $\mathbf{x}_1 \dots \mathbf{x}_n$. Under an linear affine camera we seek to determine the eight parameters in an affine projection. Rewriting equation 6.1 in affine (non-homogeneous) coordinates the constraint imposed by each image-3D point is given by

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} \mathbf{X}_i + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{6.7}$$

Each 2D-3D point correspondence hence imposes two constraints on the eight dimensional affine camera space. We can solve for the camera parameters by extending the above equation for $N \geq 4$ points and grouping the camera elements into a matrix,

$$[\mathbf{x}_1, \dots, \mathbf{x}_n] = \begin{bmatrix} \mathbf{p}_1 | t_x \\ \mathbf{p}_2 | t_y \end{bmatrix} \begin{bmatrix} \mathbf{X}_1, \dots, \mathbf{X}_n \\ 1, \dots, 1 \end{bmatrix} \tag{6.8}$$

and then transposing the expression and solving for $P = \begin{bmatrix} \mathbf{p}_1 | t_x \\ \mathbf{p}_2 | t_y \end{bmatrix}$ in

$$\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \mathbf{X}_1, 1 \\ \vdots \\ \mathbf{X}_n, 1 \end{bmatrix} P^T \tag{6.9}$$

Since we use non-homogeneous coordinates we directly minimize geometric error, hence the solution is the ML estimate for measurements perturbed by Gaussian noise.

In the case of a perspective camera model we need to estimate the full 3×4 projective camera matrix in the projection equation, $\mathbf{x}_i \sim P\mathbf{X}_i$ (equation 4.4), where here coordinates are given in homogeneous form, $\mathbf{x} = [u, v, w]^T$ and $\mathbf{X} = [X, Y, Z, t]$. The projective equivalence "$\sim$" is up to a scale, hence we cannot directly write a linear equation system as in equation 6.8. Instead, the standard solution, Direct Linear Transform (DLT) is based on expressing the equivalence as a vector product $0 = \mathbf{x}_i \times P\mathbf{X}_i$. Let $P^T = [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3]$ and rewrite the vector cross product on a matrix form we get the following three constraint equations from each point correspondence

$$0 = \begin{bmatrix} \mathbf{0}^T & -w_i\mathbf{X}_i^T & v_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & \mathbf{0}^T & u_i\mathbf{X}_i^T \\ -v_i\mathbf{X}_i^T & u_i\mathbf{X}_i^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} \tag{6.10}$$

However, only two of the three equations are linearly independent, hence we need six (or more) 2D-3D point correspondences to solve for the projection matrix P.

One common way of solving the above is to pick the first two of the three homogeneous equations for each of the six points and solve for a non-trivial null vector of the $2n \times 12$ matrix. Another alternative is to solve the overdetermined $3n \times 12$ system imposing an additional constraint $\|P\| = 1$. (Without the additional constraint, noise in the measurements will likely increase the rank and give a solution P=0.)

Intersection

In Intersection we are given two camera matrices P_1 and P_2 and the corresponding image point projections $\mathbf{x}_1, \mathbf{x}_2$, and seek the 3D point giving rise to these image projections.

In the affine case we can directly rearrange equation 6.7 as

$$\begin{bmatrix} u_1 - t_{1,x} \\ v_1 - t_{1,y} \\ u_2 - t_{2,x} \\ v_2 - t_{2,y} \end{bmatrix} = \begin{bmatrix} \mathbf{i}_1 \\ \mathbf{j}_1 \\ \mathbf{i}_2 \\ \mathbf{j}_2 \end{bmatrix} \mathbf{X} \tag{6.11}$$

and solve the overdetermined system for the 3D world point $\mathbf{X} = [X, Y, Z]$. Again, since the above equations are in pixel space we minimize geometric error, and we don't normally expect problems. (Note however that the system may be ill conditioned due to the two cameras being nearly equal, i.e. $\mathbf{i}_1$ is nearly co-linear with $\mathbf{i}_2$ and $\mathbf{j}_1$ near $\mathbf{j}_2$)

In the projective case we seek to find a homogeneous 3D point $\mathbf{X} = [X, Y, Z, t]$ that simultaneously satisfies the two equivalences $\mathbf{x}_1 \sim P_1\mathbf{X}$ and $\mathbf{x}_2 \sim P_2\mathbf{X}$. Since the equivalence is up to a scale we can write $\lambda_1\mathbf{x}_1 = P_1\mathbf{X}$ and $\lambda_2\mathbf{x}_2 = P_2\mathbf{X}$ and rewrite into one matrix equation as

$$0 = \begin{bmatrix} P_1 & \mathbf{x}_1 & \mathbf{0}^T \\ P_2 & \mathbf{0}^T & \mathbf{x}_2 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} \tag{6.12}$$

And solve the homogeneous system for a consistent 3D point $\mathbf{X}$ and scale factors λ_1, λ_2.

Structure and Motion

In the Structure and Motion (sometimes called Structure From Motion, SFM), using only the n projected points $\mathbf{x}_{i,j},\ i \in 1 \dots n,\ j \in 1 \dots m$ in m images we seek to find both camera poses $P_1, \dots, P_m$ and a consistent structure $S = [\mathbf{X}_1, \dots, \mathbf{X}_n]$. This is illustrated in Fig. 6.3.

Fast and robust linear methods based on direct SVD factorization of the image point measurements have been developed for a variety of linear cameras, e.g. orthographic [155], weak perspective [165], and para-perspective [123]. Section 6.2

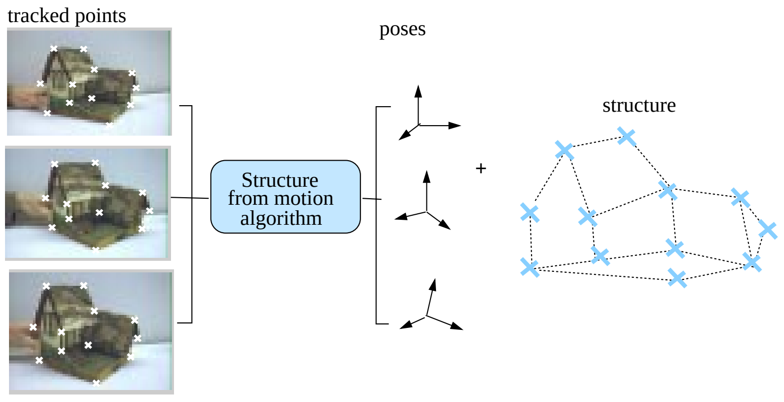


Figure 6.3: A general structure from motion algorithm extracts the structure and camera poses from a set of tracked points.

describes in detail the factorization algorithm used for obtaining an affine structure for the proposed image-based model. For non-linear perspective projection the SFM problem is more difficult. Close form solutions exist for 2,3 or 4 views configurations but there no direct linear factorization methods. However, iterative factorization methods that start with an existing structure have been developed [61, 145]. Section 6.3 presents the algorithm for obtaining the projective model developed for our image-based model and how the metric, Euclidean structure required by a robotics system can be recovered.

For a dynamic scene or a nonrigid object the problem becomes much more complicated. A solution is to decompose the scene into rigid, static components and estimate each structure separately. Bregler [13] proposed a generalization of the Tomasi-Kanade factorization using linear combinations of a set of estimated basis shapes. In our current work we made the assumption that the scene is rigid.

6.2 Affine Metric Structure

For the case when points are tracked in many images there is a linear formulation of the reconstruction problem, called factorization. The original algorithm requires that the image coordinates for all the points are available, so there are no occlusions. More recent extensions deals with missing data. Assuming an affine camera, nonisotropic zero-mean Gaussian noise, the factorization achieves ML affine reconstruction. For the present work we used an extension of the Tomasi-Kanade factorization algorithm [155] for weak perspective camera projection model similar to Weinshall and Kanade [165]. First, the algorithm recovers affine structure from a sequence of uncalibrated images. Then, a relation between the affine structure and camera coordinates is established. This is used to transform the estimated scene structure to an orthogonal coordinate frame. Since we use only image information the metric unit of measure is pixel coordinates. A more detailed mathematical formulation of the problem follows next.

6.2.1 Recovering the Affine Structure

A general affine camera has the form

$$P_\infty = [M|\mathbf{t}]$$

where M is a 2×3 matrix and $\mathbf{t}$ is a 2D vector (note that we dropped the last row $(0, 0, 0, 1)$ from the projection matrix). The projection equation using homogeneous coordinates can be written as:

$$\begin{pmatrix} x \\ y \end{pmatrix} = M \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \mathbf{t}$$

With n points tracked in m images, we form the $2m \times n$ measurement matrix W and we can write the projection equations in a compact was as:

$$W = \begin{bmatrix} \mathbf{x}_1^1 & \cdots & \mathbf{x}_n^1 \\ \vdots & & \vdots \\ \mathbf{x}_1^m & \cdots & \mathbf{x}_n^m \end{bmatrix} = \begin{bmatrix} M^1 \\ \vdots \\ M^m \end{bmatrix} [X_1 \ \cdots \ X_n] + \begin{bmatrix} \mathbf{t}^1 \\ \vdots \\ \mathbf{t}^m \end{bmatrix} = RX + \mathbf{t}\mathbf{1} \tag{6.13}$$

If the image points are registered with respect to their centroid in the image plane and the center of the world coordinate frame is the centroid of the shape points, the projection equation becomes:

$$\hat{W} = RX \ \text{ where } \ \hat{W} = W - \mathbf{t}\mathbf{1} \tag{6.14}$$

Rank theorem

Following [155], in the absence of noise rank$(\hat{W}) = 3$. Under most viewing conditions with a real camera the effective rank is 3[1]. Assuming $2m > n$, $\hat{W}$ can be decomposed $\hat{W} = O_1 \Sigma O_2$, where O_1 is an orthonormal $2m \times n$ matrix, Σ is an $n \times n$ diagonal matrix and O_2 is an $n \times n$ orthonormal matrix (SVD).
Defining

$$\begin{aligned} \hat{R} &= O_1' \\ \hat{X} &= \Sigma' O_2' \end{aligned} \tag{6.15}$$

we can write

$$\hat{W} = \hat{R}\hat{X} \tag{6.16}$$

O_1' is formed from the first three columns of O_1, Σ' is the first 3×3 matrix of Σ and O_2' contains the first three rows of O_2 (assuming the singular values are ordered in decreasing order).

The factorization calculates the affine shape of the structure $\hat{X}$ and the affine projection matrices $\hat{R}$.

[1]By effective rank 3 we mean rank near 3 i.e. the first 3 singular values are large and the $4 - n$ ones are small

Missing data

The factorization algorithm assumes that the feature points are tracked in all views. This is not the case in practice, while capturing different sides of the object when features can easily be occluded. The measurement matrix W will be incomplete. Different algorithm that deal with missing and noisy data have been developed [124]. For this work we developed an algorithm that starts with a factorization with a complete set of points that estimates an initial structure and incrementally adds new sets of tracked features that have common points with the current structure through intersection (see Section 6.1.2). The camera projection for the new frames is estimated based on common points, as described in Section 6.1.2 and then used to estimate the affine structure of the new points. A global minimization called bundle adjustment is performed at the end. The algorithm can be summarized as follows:

- Select a complete submatrix W_1 and estimate the substructure X_1 and motion $R1$ for the selected frames using the original algorithm.
- Repeat till there are no frames left:
  Add frames: estimate the motion for all the frames that have at least 4 tracked points common with the substructure X_1.
  Add points: compute the structure for all the remaining points that are tracked in at least 2 frames with known motion.
- Optimize structure and motion using bundle adjustment.

6.2.2 Recovering the Metric Structure

Assuming a weak perspective camera model, the metric properties of the structure [165] can be recovered. The weak perspective camera is a special case of affine camera where

$$M = \begin{bmatrix} s\mathbf{i} \\ s\mathbf{j} \end{bmatrix} \tag{6.17}$$

The matrices $\hat{R}$ and $\hat{X}$ resulted from the affine factorization, are a linear transformation of the metric scaled rotation matrix R and the metric shape matrix X. There is in general an affine ambiguity in the reconstruction. More specifically there exist a 3×3 matrix Q such that:

$$\begin{aligned} R &= \hat{R}Q \\ X &= Q^{-1}\hat{X} \end{aligned} \tag{6.18}$$

Normally, to align $\hat{X}$ with an exocentric metric frame the world coordinates, at least four scene points are needed. Assuming no scene information is provided,

$\hat{X}$ can be aligned with the pixel coordinate system of the camera row and column. This relates Q to the the components of the scaled rotation R:

$$\begin{aligned} &\hat{\mathbf{i}}_t^T Q Q^T \hat{\mathbf{i}}_t = \hat{\mathbf{j}}_t^T Q Q^T \hat{\mathbf{j}}_t \qquad (\, = s^2) \\ &\hat{\mathbf{i}}_t^T Q Q^T \hat{\mathbf{j}}_t = 0 \end{aligned} \tag{6.19}$$

where $\hat{R} = [\hat{\mathbf{i}}_1 \cdots \hat{\mathbf{i}}_m \hat{\mathbf{j}}_1 \cdots \hat{\mathbf{j}}_m]^T$ The first constraint assures that the corresponding rows $s_t\mathbf{i}_t^T$, $s_t\mathbf{j}_t^T$ of the scaled rotation R in equation 6.17 are unit vectors scaled by the factor s_t and the second equation constrain them to orthogonal vectors. This generalizes [155] from an orthographic to a weak perspective case. The resulting transformation is up to a scale and a rotation of the world coordinate system. To eliminate the ambiguity, the axis of the reference coordinate system is aligned with the first frame. We also estimate only eight parameters in Q (fixing a scale).

6.3 Euclidean Structure for Perspective Projection

The problem of uncalibrated SFM for a projective camera has been the focus of many research studies in the last decade [47, 59]. Close-form solution has been developed for the case of 2,3 or 4 views. The 2 view geometry formulation, was presented in Section 4.1.4. The epipolar geometry between the views is algebraically captured by the fundamental matrix F. There is a family of camera matrices, and projective structures that are consistent with this reconstruction as, in general, a camera matrix depends on the internal parameters and choice of world coordinate frame (external parameters), but F does not depend on the choice of the world frame and is unchanged by a projective transformation of 3D space. So two projective matrices P, P' uniquely determine F but the converse is not true. Given this projective ambiguity, we can formulate a class of projection matrices given a fundamental matrix (canonical cameras):

$$P = [I|0] \tag{6.20}$$

$$P' = [[\mathbf{e}']_\times F + \mathbf{e}'\mathbf{v}^T | \lambda \mathbf{e}'] \tag{6.21}$$

Similar constraints can be defined for 3 and 4 images. The epipolar constraint can be formulated for three images, divided in groups of 2. But there is a stronger constraint that involves all three: the projection of a point in the third image can be computed from corresponding projections in the other two images. A similar constraint hold for lines or combinations of points/lines. This are called trilinear constraints and can be expressed using the trifocal tensor [157].

Quadrifocal constraints are formulated for 4 images (using the quadrifocal tensor) [159]. An important result is that there are no additional constraints between more than 4 images. All the constraints can be expressed using F, the trilinear tensor and the quadrifocal tensor. An unifying theory uses multiview tensors [62, 58].

Here is a summary of different constraints and the minimal configurations for computing structure from motion:

2 images *epipolar geometry*, F

- linear unique solution from 8 points
- nonlinear solution from 7 points (3 solutions)

3 images *trifocal tensor*

- linear solution from 7 points
- nonlinear solution from 6 points

4 images *quadrifocal tensor*

- linear solution from 6 points

For generating the projective geometry for the proposed image-based model, we used the method developed by Urban et al [162] that estimates the trilinear tensors for triplets of views and then recovers epipoles from adjoining tensors. The projection matrices are computed at once using the recovered epipoles. This algorithm is applied for a subset of points and views. More points/camera projections are added through intersection/resection (Sections 6.1.2, 6.1.2).

Having the projective depth recovered by this algorithm, the projective structure is enhanced through factorization.

6.3.1 Projective Factorization

A similar idea to the affine factorization (Section 6.2.1) can be adopted for projective cameras. Several algorithms exist in the literature [145, 61]. The projection equations $\lambda_j^i \mathbf{x}_j^i = P^i \mathbf{X}_j$ for n points and m images can be written as (in homogeneous coordinates):

$$W = \begin{bmatrix} \lambda_1^1 \mathbf{x}_1^1 & \cdots & \lambda_n^1 \mathbf{x}_n^1 \\ \vdots & & \vdots \\ \lambda_1^m \mathbf{x}_1^m & \cdots & \lambda_n^m \mathbf{x}_n^m \end{bmatrix} = \begin{bmatrix} P^1 \\ \vdots \\ P^m \end{bmatrix} [X_1 \ \cdots \ X_n] + \begin{bmatrix} \mathbf{t}^1 \\ \vdots \\ \mathbf{t}^m \end{bmatrix} \tag{6.22}$$

λ_j^i are called the projective depth and are in general unknown. Assuming λ_j^i are known (from the initial projective reconstruction), the camera matrices and 3D projective points can be computed using a factorization algorithm as in 6.2.1. The measurement matrix has rank four in this case.

If the projective depths are unknown an iterative approach can be adopted. Initially they are all set to 1 and the structure is estimated using factorization. The depths are re-estimated by reprojecting the structure and the procedure is repeated. This method works well for simple, mainly planar surfaces, but in general there is no guarantee that the procedure will converge to a global minimum.

6.3.2 Bundle Adjustment

Consider that the projective 3D structure $\hat{\mathbf{X}}_j$ and the camera matrices $\hat{P}_i$ had been estimated from a set of image features $\mathbf{x}_j^i$. The bundle adjustment refines this estimates by minimizing the geometric error

$$\min \sum_{i,j} d(\hat{P}^i \hat{\mathbf{X}}_j, \mathbf{x}_j^i)^2 \tag{6.23}$$

The name *bundle adjustment* means readjusting bundle of rays between the camera center and the set of 3D points to fit the measured data. The solution looks for the Maximum Likelihood (ML) estimate assuming that the measurement noise is Gaussian. This step is very important in any projective reconstruction and is tolerant to missing data.

6.3.3 Projective Ambiguity

Given an uncalibrated sequence of images with corresponding points identified it is possible to reconstruct the structure only up to a projective transformation.

But, there exist a homography H (or a family of homographies) such that the transformed matrices P^iH represent true projection matrices and an be decomposed as: $P^iH = KR^i[I|\mathbf{t}^i]$, where $K, R^i, \mathbf{t}^i$ represent the internal and external parameters of the camera as described in subsection 4.1.1. The projective structure is upgraded to metric by $H^{-1}X_j$.

The general approach for a metric reconstruction is:

- Obtain a projective reconstruction P^i, X_j
- Determine the rectifying homography H from autocalibration constraints, and transform the reconstruction to metric $P^iH, H^{-1}X_j$

6.3.4 Self Calibration

The theoretical formulation of self-calibration (auto-calibration) for constant camera parameters was first introduced by Faugeras [45]. The goal is to recover the projective ambiguity by imposing only constraints on the internal camera parameters. To restrict the projective ambiguity (15 DOF) to a metric one (7 DOF - 3 for rotation, 3 for translation and 1 for scale), at least 8 constraints are needed [127].

An internal camera parameter known in m views, gives n constraints; if the parameter is fixed across m views gives $m-1$ constraints. Therefore the absence of skew was shown to be enough for self-calibration with more than 8 views ([126, 63]. If, in addition, the aspect ration is known ($a_u = a_v$), 4 views are sufficient. In the case where the principal point is known two views impose enough constraints for self-calibration.

Present techniques can be dived in two main categories:

(a) a stratified approach that first determines the affine structure and then the metric structure using the absolute conic;

(b) a direct approach that recovers the metric structure using the dual absolute quadric.

We developed a self-calibration algorithm that belongs to the second category and uses the image of the image of the absolute dual quadric (IDAC) assuming known internal camera parameters. A very important concept in self-calibration is the Absolute Conic (AC) and its image projection (IAC). Since it is invariant to Euclidean transformations its relative position to a moving camera is constant. Therefore, if the camera internal parameters are not changing its image is also constant. It can be seen as a calibration object that is present in all scenes. Once the AC is recovered, it can be used to upgrade the reconstruction to metric. A practical way of representing the AC is its dual - Dual Absolute Quadric (DAC) [160] that expresses both the AC and its supporting plane (plane at infinity) in one entity.

The DAC, Ω^* is related by the image of the absolute conic (IAC) ω_t^* in view t by:

$$\omega_t^* \sim P_t \Omega^* P_t^T \tag{6.24}$$

where P_t denotes the projection matrix for view t. For a calibrated Euclidean camera, $P_t = K[R_t|\mathbf{t}_t]$ the DAC has the canonical configuration $\Omega^* = diag(1,1,1,0)$ so by substituting in equation 6.24 we obtain:

$$\omega_t^* = KK^T \tag{6.25}$$

In the case of a projective reconstruction, the DAC will not be in its standard position, so the self-calibration algorithm recovers the projective transformation H that will transform the DAC in its standard position:

$$KK^T = P_t(H\Omega^* H^T)P_t^T \tag{6.26}$$

Assuming the camera has been calibrated, the projection matrices can be normalized, $P_t = K^{-1}P_t$ and camera matrix is a 3×3 unity matrix, $K = diag(1,1,1)$. The model transformed and aligned with the first camera so, $P_0 = [I|\mathbf{0}]$. The rectifying homography has the form

$$H = \begin{bmatrix} K & 0 \\ \mathbf{v}' & a \end{bmatrix} \tag{6.27}$$

where $\mathbf{v} = -\mathbf{p}^T K$, $\mathbf{p}$ is the normal of the plane at infinity $\pi_\infty = (\mathbf{p}^T, 1)^T$ and a is a scalar that can be set to 1. $\mathbf{v}$ is recovered from Equation 6.26. In a more general formulation, K is assumed unknown and the DAC is recovered from 6.26 by imposing internal constraints on K.

Note that, self-calibration is not a general solution to a metric reconstruction. Some critical motions [146] can generate degenerate solutions (e.g. planar motion and constant internal parameters) and the constraints on the internal parameters has to be carefully chosen depending on each real camera. Some external constraints on the scene (if available) like knowledge about parallel lines, angles can improve the robustness. Metric bundle adjustment is performed as a final step.

6.4 Dynamic Textures

In traditional graphics, images are generated by texture mapping a 3D scene model. Using computer vision structure from motion techniques a geometric model is reconstructed from corresponding image features. Typically relatively few points can be extracted reliably, and the scene structure is thus sparse. This leads to significant parallax errors when a planar model facet represents a non-planar scene surface. Additionally, misalignments in the computed corresponding image points lead to incorrect texture coordinates and image plane errors in texture mapping.

We propose a texture-based correction [22, 25], which relaxes the accuracy requirement on the structure estimation. This section describes how to make the texture correctly represent the variation over different viewpoints of a potentially complex underlying geometry. Using the tools of 3D warping, *relief textures* provides an explicit geometric solution to adding 3D structure to a planar texture [117]. However, relief textures require a detailed a-priori depth map of the texture element. This is normally not available in image-based modeling if only uncalibrated camera video is used. An alternative way to represent the image motion caused by depth parallax is by modulating a spatial image basis. Previously, this technique has been used in image (camera) plane encoding of deterministic scene motion [73] and capturing certain quasi-periodic motions [39].

In the previous Chapter, I described how uncalibrated video can be used to obtain geometric information from a scene. A structure-from-motion (SFM) method starts with a set of m images $I_1 \ldots I_m$ from different views of a scene. Through visual tracking or correspondence detection the image projection $x_1 \ldots x_n$ of n physical scene points are identified in every image. From this, a structure from motion computes a structure, represented by a set of n scene points $X_1 \ldots X_n$, and m view projections $P_1 \ldots P_m$ such that (reprojection property):

$$x_{ti} = P_t X_i \quad i \in 1 \ldots n, t \in 1 \ldots m \tag{6.28}$$

Independent of the geometric details and interpretation and central to image based modeling and rendering is that this structure can be reprojected into a new virtual camera and thus novel views can be rendered. Practically, the structure is divided into K planar facets (triangles or quadrilaterals are used in the experiments) with the points x_{ti} as node points. For texture mapping, each one of the model facets are related by a planar projective homography to a texture image. See Figure 6.6.

6.4.1 Texture Basis

In conventional texture mapping, one or more of the real images are used as a source to extract texture patches from, and then warped onto the re-projected structure in the new view.

Instead of using an image as a source texture, we study how to relate and unify all the input sample images into a texture basis. Let x_{Ti} be a set of texture coordinates in one-to-one correspondence to each model point X_i and thus also for each view t with the image points x_{ti} above. A texture warp function W translates the model vertex to texture correspondences into a pixel-based re-arrangement (or warp) between the texture space I_W to screen image coordinates I.

$$T(\mathbf{x}) = I(W(\mathbf{x}; \mu)) \tag{6.29}$$

where μ are the warp parameters and $\mathbf{x}$ the image coordinates.

Common such warp functions are affine, bi-linear and projective warps. The warp function W acts by translating, rotating and stretching the parameter space of the image, and hence for discrete images a re-sampling and filtering step is needed between the image and texture spaces. Details of these practicalities can be found in [108].

Now if for each sample view t, we warp the real image I_t from image to texture coordinates into a texture image T_t, we would find that in general $T_t \neq T_k,\ t \neq k$. Typically, the closer view t is to k, the smaller is the difference between T_t and T_k. This is the rationale for view-dependent texturing, where a new view is textured from one to three (by blending) closest sample images [35].

Here we will develop a more principled approach, where we seek a texture basis B such that for each sample view:

$$\mathbf{T}_t = B\mathbf{y}_t,\ t \in 1 \dots m. \tag{6.30}$$

We denote by $\mathbf{T}$ is a $q \times q$ texture image flattened into a $q^2 \times 1$ column vector. B is a $q^2 \times r$ matrix, where normally $r \ll m$, and $\mathbf{y}$ is a modulation vector. The texture basis B needs to capture geometric and photometric texture variation over the sample sequence, and correctly interpolate new in-between views. We first derive a first order geometric model, then add the photometric variation. For clarity we develop these applied to one texture warp (as in Figure 7.16), while in practical applications a scene will be composed by texturing several model facets (as in Figures 6.6 and 7.13).

6.4.2 Geometric Texture Variation

The starting point for developing a spatial texture basis representing small geometric variations is the well known optic flow constraint, which for small image plane translations relates texture intensity change $\Delta T = T_t - T_k$ to spatial derivatives $\frac{\partial}{\partial u}T$, $\frac{\partial}{\partial v}T$ with respect to texture coordinates $\mathbf{x} = [u, v]^T$ under an image constancy assumption [56].

$$\Delta T = \frac{\partial T}{\partial u}\Delta u + \frac{\partial T}{\partial v}\Delta v \tag{6.31}$$

Note that given one reference texture T_0 we can now build a basis for small image plane translations $B = [T_0, \frac{\partial T}{\partial u}, \frac{\partial T}{\partial v}]$ and from this generate any slightly translated texture $T(\Delta u, \Delta v) = B[1, \Delta u, \Delta v]^T = B\mathbf{y}$.

In a real situation, the patch is deforming in a more complex way than only translating. This deformation is captured by the warp parameters. Given a warp function $\mathbf{x}' = W(\mathbf{x}; \mu)$ we study the residual image variability introduced by the imperfect stabilization achieved by a perturbed warp $W(\mathbf{x}; \hat{\mu})$, $\Delta T = T(W(\mathbf{x}; \hat{\mu}), t) - T(W(\mathbf{x}; \mu))$. Let $\hat{\mu} = \mu + \Delta\mu$ and rewrite as an approximate image variability to the first order (dropping t):

$$\begin{aligned} \Delta T = T(W(\mathbf{x}; \mu + \Delta\mu)) - T_W &= T(W(\mathbf{x}; \mu)) + \nabla T \tfrac{\partial W}{\partial \mu}\Delta\mu - T_W \\ &= \nabla T \tfrac{\partial W}{\partial \mu}\Delta\mu \end{aligned} \tag{6.32}$$

The above equation expresses an optic flow type constraint in an abstract formulation without committing to a particular form or parameterization of $W(\mu)$. In practice, the function W is usually discretized using e.g. triangular or quadrilateral mesh elements. Next we give examples of how to concretely express image variability from these discrete representations.

Particularly for image based modeling and rendering we warp real source images into new views given an estimated scene structure. Errors between the estimated and true scene geometry cause these warps to generate imperfect renderings. We divide these up into two categories, *image plane* and *out of plane* errors. The planar errors cause the texture to be sourced with an incorrect warp.[2]. The out of plane errors arise when piecewise planar facets in the model are not true planes in the scene, and when re-warped into new views under a false planarity assumption will not correctly represent parallax.

Planar texture variability

First we will consider geometric errors in the texture image plane. In most both IBR (as well as conventional rendering) textures are warped onto the rendered view from a source texture T by means of a affine warp or projective homography.

[2]Errors in tracking and point correspondences when computing the SFM, as well as projection errors due to differences between the camera model and real camera both cause model points to be reprojected incorrectly in the sample images

Affine variation Under a weak perspective (or orthographic) camera geometry, plane-to-plane transforms are expressed using an affine transform of the form:

$$\begin{bmatrix} u_w \\ v_w \end{bmatrix} = \mathcal{W}_a(\mathbf{p}, \mathbf{a}) = \begin{bmatrix} a_3 & a_4 \\ a_5 & a_6 \end{bmatrix} \mathbf{p} + \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{6.33}$$

This is also the standard image-to-image warp supported in OpenGL. Now we can rewrite the image variability equation 6.32 resulting from variations in the six affine warp parameters as:

$$\Delta T_a = \sum_{i=1}^{6} \frac{\partial}{\partial a_i} T_w \Delta a_i = \left[\frac{\partial T}{\partial u}, \frac{\partial T}{\partial v}\right] \begin{bmatrix} \frac{\partial u}{\partial a_1} \cdots \frac{\partial u}{\partial a_6} \\ \frac{\partial v}{\partial a_1} \cdots \frac{\partial v}{\partial a_6} \end{bmatrix} \Delta[a_1 \ldots a_6]^T \tag{6.34}$$

Let $\{T\}_{discr} = \mathbf{T}$ be a discretized texture image flattened along the column into a vector. Rewriting the inner derivatives to get an explicit expression of the six parameter variability in terms of spatial image derivatives we get:

$$\begin{aligned} \Delta\mathbf{T}_a(u,v) = \left[\frac{\partial \mathbf{T}}{\partial u}, \frac{\partial \mathbf{T}}{\partial v}\right] \begin{bmatrix} 1 & 0 & u & 0 & v & 0 \\ 0 & 1 & 0 & u & 0 & v \end{bmatrix} [y_1, \ldots, y_6]^T = \\ [\mathbf{B}_1 \ldots \mathbf{B}_6][y_1, \ldots, y_6]^T = B_a \mathbf{y}_a \end{aligned} \tag{6.35}$$

where $[\mathbf{B}_1 \ldots \mathbf{B}_6]$ can be interpreted as a texture variability basis for the affine transform.
Projective variation Under a perspective camera the plane-to-plane warp is expressed by a projective collineation or homography,

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \mathcal{W}_h(\mathbf{x}_h, \mathbf{h}) = \frac{1}{1 + h_7 u + h_8 v} \begin{bmatrix} h_1 u & h_3 v & h_5 \\ h_2 u & h_4 v & h_6 \end{bmatrix} \tag{6.36}$$

Rewrite equation 6.32 with the partial derivatives of W_h for the parameters $h_1 \ldots h_8$ into a Jacobian matrix. Let $c_1 = 1 + h_7 u + h_8 v$, $c_2 = h_1 u + h_3 v + h_5$, and $c_3 = h_2 u + h_4 v + h_6$. The resulting texture image variability due to variations the estimated homography is (to the first order) spanned by the following spatial basis:

$$\begin{aligned} \Delta\mathbf{T}_h(u,v) = \frac{1}{c_1} \left[\frac{\partial \mathbf{T}}{\partial u}, \frac{\partial \mathbf{T}}{\partial v}\right] \begin{bmatrix} u & 0 & v & 0 & 1 & 0 & -\frac{uc_2}{c_1} & -\frac{vc_2}{c_1} \\ 0 & u & 0 & v & 0 & 1 & -\frac{uc_3}{c_1} & -\frac{vc_3}{c_1} \end{bmatrix} \begin{bmatrix} \Delta h_1 \\ \vdots \\ \Delta h_8 \end{bmatrix} = \\ [\mathbf{B}_1 \ldots \mathbf{B}_8][y_1, \ldots, y_8]^T = B_h \mathbf{y}_h \end{aligned} \tag{6.37}$$

Similar expressions can be derived for other warps. E.g. in real time visual tracking a four parameter variability from modeling image u, v translations, image plane rotations and scale has shown to be suitable [56].

Non-planar parallax variation

While in image-based modeling a scene is represented as piecewise planar model facets, the real world scene is seldom perfectly planar. In rendering this gives rise to parallax errors. Figure 6.4 illustrates how the texture plane image T changes for different scene camera centers C. Given a depth map $d(u, v)$ representing the offset between the scene and texture plane, relief texturing [117] can be used to compute the rearrangement (pre-warp) of the texture plane before the final homography renders the new view. In image-based methods, an accurate depth map is seldom available. However we can still develop the analytic form of the texture intensity variation as above. Let $r = [\alpha, \beta]$ be the angle for view P_t between the ray from the camera center C_t to each scene point. The pre-warp rearrangement needed on the texture plane to correctly render this scene using a standard homography warp is then:

$$\begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \mathcal{W}_p(\mathbf{x}, \mathbf{d}) = d(u, v) \begin{bmatrix} \tan\alpha \\ \tan\beta \end{bmatrix} \tag{6.38}$$

As before, taking the derivatives of the warp function with respect to a camera angle change and inserting into equation 6.32 we get:

$$\Delta \mathbf{T}_p(u, v) = d(u, v) \left[\frac{\partial \mathbf{T}}{\partial u}, \frac{\partial \mathbf{T}}{\partial v} \right] \begin{bmatrix} \frac{1}{\cos^2\alpha} & 0 \\ 0 & \frac{1}{\cos^2\beta} \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta\beta \end{bmatrix} = B_p \mathbf{y}_p \tag{6.39}$$

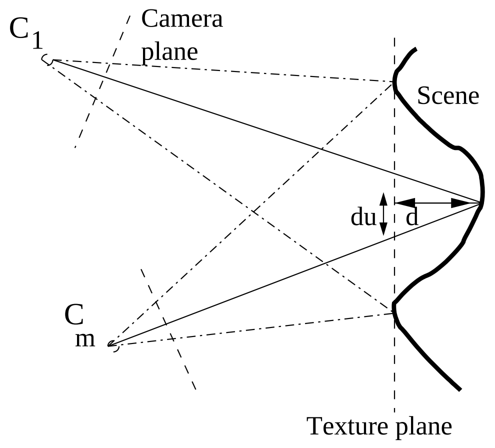


Figure 6.4: Texture parallax between two views

Non-rigidity

We consider only non-rigidities where the shape change is a function of some measurable quantity $\mathbf{x} \in \Re^n$. In this paper we choose $\mathbf{x}$ from pose and articulation parameters. Let $\mathbf{g}(\mathbf{x})$ $(= [u, v]^T)$ represent the image plane projection of the non-rigid warp. We can then write the resulting first order image variation as:

$$\begin{gathered}\Delta\mathbf{T}_n = \left\{\sum_{i=1}^{n} \frac{\partial \mathbf{T}}{\partial x_i}\Delta x_i\right\}_{\mathrm{discr}} = \\ \left\{\left[\frac{\partial \mathbf{T}}{\partial u}, \frac{\partial \mathbf{T}}{\partial v}\right]\left[\frac{\partial}{\partial x_1}g(x)\Delta x_1, \ldots, \frac{\partial}{\partial x_n}g(x)\Delta x_n\right]\right\}_{\mathrm{discr}} = \\ [\mathbf{B}_1 \ldots \mathbf{B}_n][y_1, \ldots, y_n]^T = B_n\mathbf{y}_n\end{gathered} \tag{6.40}$$

6.4.3 Photometric Variation

In image-based rendering real images are re-warped into new views, hence the composite of both reflectance and lighting is used. If the light conditions are same for all sample images, there is no additional intensity variability introduced. However, commonly the light will vary at least somewhat. In the past decade several published both empirical studies and theoretical motivations have shown that a low dimensional intensity subspace of dimension 5-9 is sufficient for representing the light variation of most natural scenes [88]. Hence we introduce nine additional freedoms in our variability model to allow for lighting. (Complex scenes may require more, simple (convex lambertian) less).

$$\Delta\mathbf{T}_l = [\mathbf{B}_1 \ldots \mathbf{B}_9][y_1 \ldots y_9]^T = B_l\mathbf{y}_l \tag{6.41}$$

6.4.4 Estimating Composite Variability

In textures sampled from a real scene using an estimated geometric structure we expect that the observed texture variability is the composition of the above derived planar, parallax and light variation, as well as unmodeled effects and noise $\Delta\mathbf{T}_e$. Hence, total residual texture variability can be written as:

$$\Delta\mathbf{T} = \Delta\mathbf{T}_{h/a} + \Delta\mathbf{T}_p + \Delta\mathbf{T}_n + \Delta\mathbf{T}_n + \Delta\mathbf{T}_e \tag{6.42}$$

Using the basis derived above we can write the texture for any sample view t, and find a corresponding texture modulation vector $\mathbf{y}_t$:

$$\mathbf{T}_k = [\mathbf{T}_0, B_{h/a}, B_p, B_n, B_l][1, y_1, \ldots, y_k] = B\mathbf{y}_t \tag{6.43}$$

Textures for new views are synthesized by interpolating the modulation vectors from the nearest sample views into a new $\mathbf{y}$, and computing the new texture $T_{new} = B\mathbf{y}$

Since this basis was derived as a first order representation it is valid for (reasonably) small changes only. In practical image-based modeling the geometric point misalignments and parallax errors are typically within a few pixels, which is small enough.

In IBR typically, neither the dense depth needed to analytically compute B_p, nor light and reflectance models needed for B_l are available. Instead the only available source of information are the sample images $I_1 \dots I_m$ from different views of the scene, and from these, the computed corresponding textures $\mathbf{T}_1 \dots \mathbf{T}_m$.

However, from the above derivation we expect that the effective rank of the sample texture set is the same as of the texture basis B, i.e. $\text{rank}[\mathbf{T}_1, \dots, \mathbf{T}_m] \approx k$. Hence, from $m \gg k$ (typically 100-200) sample images we can estimate the best fit (under some criterion) rank 20 subspace using e.g. PCA, SVD, or ICA.

Briefly PCA can be performed as follows. Form a measurement matrix $A = [\mathbf{T}_1, \dots, \mathbf{T}_m]$. The principle components are the eigenvectors of the covariance matrix $C = AA^T$. A dimensionality reduction is achieved by keeping only the first k of the eigenvectors. For practical reasons, usually $k \ll m \ll l$, where l is the number of pixels in the texture patch, and the covariance matrix C will be rank deficient. We can then save computational effort by instead computing $L = A^T A$ and eigenvector factorization $L = VDV^T$, where V is an ortho-normal and D a diagonal matrix. From the k first eigenvectors $\hat{V} = [\mathbf{v}_1 \dots \mathbf{v}_k]$ of L we form a k-dimensional eigenspace $\hat{B}$ of C by $\hat{B} = A\hat{V}$. Using the estimated $\hat{B}$ we can now write a least squares optimal estimate of any intensity variation in the patch as

$$\Delta \mathbf{T} = \hat{B}\hat{\mathbf{y}}, \tag{6.44}$$

the same format as equation 6.42, but without using any a-priori information to model B. While $\hat{\mathbf{y}}$ captures the same variation as $\mathbf{y}$, it is not parameterized in the same coordinates. For every training texture $\mathbf{T}_t$ we have from the orthogonality of $\hat{V}$ that the corresponding texture mixing coefficients are the columns of $[\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_m] = \hat{V}^T$. From the geometric structure we also have the corresponding $\mathbf{p}_t$.

This yields an estimated texture basis $\hat{B}$ and corresponding space of modulation vectors $\hat{\mathbf{y}}_1, \dots \hat{\mathbf{y}}_m$ in one-to-one correspondence with the m sample views. From the derivation of the basis vectors in B we know this variation will be present and dominating in the sampled real images. Hence, the analytical B and the estimate $\hat{B}$ span the same space and just as before, new view dependent textures can now be modulated from the estimated basis by interpolating the $\hat{\mathbf{y}}$ corresponding to the closest sample views and modulating a new texture $\mathbf{T} = \hat{B}\hat{\mathbf{y}}$. Practically to estimate the texture mixing coefficients for intermediate poses, we first apply n-dimensional Delaunay triangulation over the sampled poses $\mathbf{p}_1, \dots, \mathbf{p}_m$. Then given a new pose $\mathbf{p}$ we determine which simplex the new pose is contained in, and estimate the new texture mixing coefficients $\hat{\mathbf{y}}$ by linearly interpolating the mixing coefficients of the corner points of the containing simplex.

6.4.5 Interpretation of the Variability Basis

The geometric model captures gross image variation caused by large movements. The remaining variation in the rectified patches is mainly due to:

1. Tracking errors as well as errors due to geometric approximations (e.g. weak perspective camera) cause the texture to be sourced from slightly inconsistent locations in the training images. These errors can be modeled as a small deviation $\Delta[h_1, \ldots, h_8]^T$ in the homography parameters from the true homography, and causes image differences according to equation 6.37. The camera approximations, as well as many tracking errors are persistent, and a function of object pose. Hence they will be captured by $\hat{B}$ and indexed in pose $\mathbf{x}$ by $\hat{\mathbf{y}}$.

2. Depth variation is captured by equation 6.39. Note that projected depth variation along the camera optic axis changes as a function of object pose.

3. Assuming fixed light sources and a moving camera or object, the light variation is a function of relative camera-object pose as well.

From the form of Equations 6.37 and 6.39 we expect that pose variations in the image sequence will result in a texture variability described by combinations of spatial image derivatives. In Figure 6.5 we compare numerically calculated spatial image derivatives to the estimated variability basis $\hat{B}$.

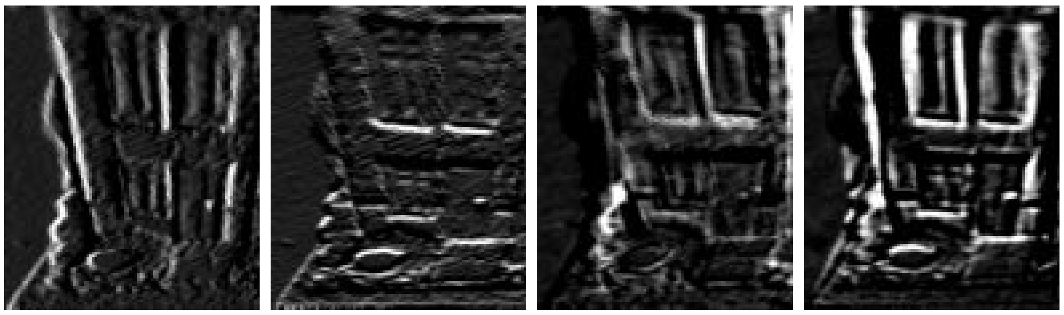


Figure 6.5: Comparison between spatial derivatives $\frac{\partial T_w}{\partial x}$ and $\frac{\partial T_w}{\partial y}$ (left two texture patches) and two vectors of the estimated variability basis $[\mathbf{B}_1, \mathbf{B}_2]$ (right) for house pictures.

6.5 Combining Geometric Model and Dynamic Textures for Image-Based Rendering

In the proposed system, the approximate texture image stabilization achieved using a coarse model reduces the difficulty of applying IBR techniques. The residual image (texture) variability can then be coded as a linear combination of a set of spatial filters (Figure 6.6). The main steps in obtaining and reprojecting the model are enumerated below. Next chapter describes the implementation details for each step.

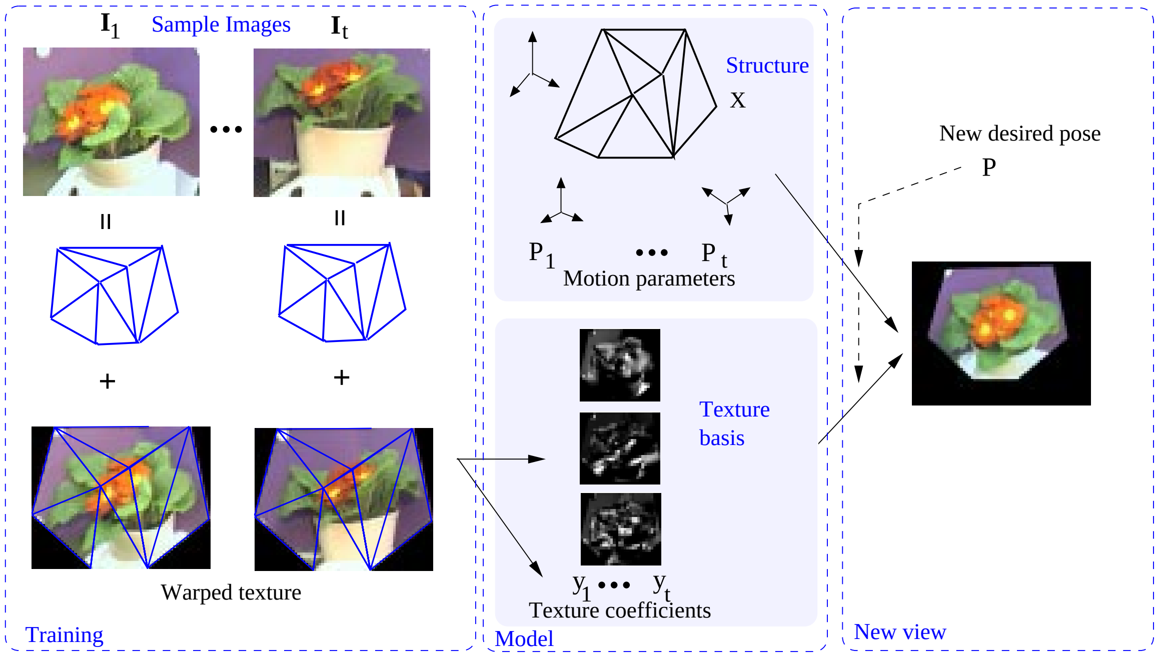


Figure 6.6: A sequence of training images $I_1 \cdots I_t$ is decomposed into geometric shape information and dynamic texture for a set of quadrilateral patches. The scene structure X and views P_t are determined from the projection of the structure using a structure-from-motion algorithm. The view-dependent texture is decomposed into its projection $\mathbf{y}$ on an estimated basis B. For a given desired position, a novel image is generated by warping new texture synthesized from the basis B on the projected structure. On the web site is a compiled demo rendering this flower and some captured movies

Training data

We capture a training sequence of images I_t where feature point locations $x_t = [\mathbf{u}_t; \mathbf{v}_t]$ were tracked using visual tracking as described in [57] and Section 7.1.

Geometric model

A coarse geometric structure of the scene X and a set of motion parameters $\mathbf{p}_t = (R_t, \mathbf{t}_t)$ that uniquely characterize each frame is estimated from the tracked points using structure from motion. We have shown (Sections 6.2, 6.3) that metric structures can be reconstructed from a sequence of uncalibrated images with a set of tracked feature points under different camera models (affine or projective). R represents a 3D rotation and, depending of the choice of geometric representation $\mathbf{t}$ is a 2D translation (affine metric structure) or a 3D translation (Euclidean structure). The reprojection of the structure given a set of motion parameters $\mathbf{p}$ is obtained by

$$x = P(\mathbf{p})X \tag{6.45}$$

where $P(\mathbf{p})$ is the projection matrix formed using the motion params $\mathbf{p}$.

Dynamic texture

The projection of the estimated structure into the sample images, x_t, is divided into Q triangular regions I_{kt} that are then warped to a standard shape T_{kt} to generate a texture T_t.

$$I_t = \sum_{k=1}^{Q} I_{kt} \tag{6.46}$$

$$T_{kt} = I_{kt}(\mathcal{W}(x; \mu(x_t))) \tag{6.47}$$

$$T_t = \sum_{k=1}^{Q} T_{kt} \tag{6.48}$$

Notation $\mu(x_t)$ indicate that the warp parameters are computed from the position of the tracked points x_t, that are mapped to their standard pose x_T. The standard shape x_T is chosen to be the average positions of the tracked points scaled to fit in a square region as shown in Figure 6.6. Practically, using HW accelerated OpenGL (see Section 7.2) each frame I_t is loaded into texture memory and warped to a standard shape texture T_t based on tracked positions (equations 6.46, 6.47, 6.48). Using the algorithm described in section 6.4.4 we compute a set of basis images B that capture the image variability caused by geometric approximations and illumination changes and the set of corresponding blending coefficients $\mathbf{y}_t$.

$$T_t = B\mathbf{y}_t + \bar{T} \tag{6.49}$$

New view generation

To generate a new view from the desired pose **p** we compute the reprojection $x = [\mathbf{u}; \mathbf{v}]$ of the geometric model as in equation 6.45. The texture blending coefficients **y** are estimated by interpolating the the nearest neighbors coefficients from the coefficients and poses of the original training sequence. The new texture in the standard shape is computed using equation 6.49, and finally the texture is warped to the projected structure (inverse of equations 6.47 and 6.46). The texture computation and rendering are performed in hardware as described in Section 7.2.3.

Chapter 7

Dynamic Texture Model Implementation and Applications

This chapter presents the practical implementation and performance of the dynamic texture model as well as its applications in tracking and predictive display. A 3D model-based tracking algorithm uses the estimated geometric model to track the camera motion. The algorithm was integrated in the model acquisition stage but also used independently for tracking the position of a mobile robot. The dynamic texture model is more complex than the panoramic model and its different components were integrated into a rendering system with real-time performance. Section 7.2 presents the implementation details for each part of the system. Most of the initial experiments were done with small objects, mainly due to the tracking limitations. For robotics applications bigger scenes need to be captured and rendered. Later experiments demonstrate the applicability of the model to a bigger space (room and research lab). Section 7.3 presents a qualitative and quantitative evaluation of different models created with our system. The model can be reanimated and rendered from any desired position, that is made use of in a predictive display application described in Section 7.4.

7.1 3D SSD Tracking

In visual tracking pose parameters of a moving camera or object are extracted from a video sequence. One way of classifying tracking methods is into feature based, segmentation based and registration based.

In feature based tracking a feature detector is used to locate the image projection of either special markers or natural image features. Then a 3D pose computation can be done by relating 2D image feature positions with their 3D model. Many approaches use image contours (edges or curves) that are matched with an a-priori given CAD model of the object [95, 99, 41]. Most systems compute pose parameters by linearizing with respect to object motion. A characteristic of these algorithms is that the feature detection is relatively decoupled from the pose computation, but sometimes past pose is used to limit search ranges, and the global model can be used to exclude feature mismatches [95, 1].

In segmentation based tracking some pixel or area based property (e.g. color, texture) is used to binarize an image. Then the centroid and possibly higher moments of connected regions are computed. While the centroid and moments are sufficient to measure 2D image motion, it is typically not used for precise 3D tracking alone, but can be used to initialize more precise tracking modalities [158].

In registration based tracking the pose computation is based on directly aligning a reference intensity patch with the current image to match each pixel intensity as closely as possible. Often a sum-of-squared differences (e.g. L_2 norm) error is minimized, giving the technique its popular name SSD tracking. This technique can also be used in image alignment to e.g. create mosaics [147]. Early approaches used brute force search by correlating a reference image patch with the current image. While this worked reasonably well for 2D translational models, it would be impractical for planar affine and projective (homography) image transforms. Instead, modern methods are based on numerical optimization, where a search direction is obtained from image derivatives. The first such method required spatial image derivatives to be recomputed for each frame when "forward" warping the reference patch to fit the current image [97], while most recently, efficient "inverse" algorithms have been developed, which allow the real time tracking for the above mentioned 6D affine [56] and 8D projective warp [3]. A related approach [77, 53], where instead of using spatial image derivatives, a linear basis of test image movements is used to explain the current frame, has proved equally efficient as the inverse methods during the tracking, but suffers from much longer initialization times to compute the basis, and a heuristic choice of the particular test movements.

We extend the registration based technique by involving a full 3D scene model, estimated from the same uncalibrated video, and used directly in the computation of the motion update between frames. Hence, we are able to utilize general image patches in a general 3D scene, and directly track the rotational and translational camera-scene motion from image differences. Some advantages of using a global 3D model and surface patches are that only surfaces with salient intensity variations need to be processed, while the 3D model connect these together in a physically correct way. We show experimentally that this approach yields more stable and robust tracking than previous approaches, where each surface patch motion is computed individually.

The rest of this section is organized as follows: We start with a presentation of the general tracking algorithm in Section 7.1.1, and then the details for useful combinations of motions (3D models and 2D planar image warps) in Section 7.1.2. The qualitative and quantitative evaluation of the algorithm is presented in Section 7.1.3.

7.1.1 General Tracking Problem formulation

We consider the problem of determining the motion of a rigid structure through a sequence of images using image registration. A sparse 3D structure for the model described by 3D points $\mathbf{Y}_i$, $i = 1, N$ is calculated in a training stage using uncalibrated structure from motion techniques (see Section 6.3). The structure points define Q image regions that are tracked in the sequence. Each region $\mathcal{R}_k$ is determined by a number of control points $\mathbf{Y}_{kj}$ that define its geometry. For example, a planar surface region can be specified by 4 corner points. The model points are projected onto the image plane using a projective transformation. First we develop the general theory without committing to a particular projection model and denote the general 3×4 projection matrix for image I_t by P_t. Hence, the model points are projected in image I_t using:

$$\mathbf{y}_{ti} = P_t \mathbf{Y}_i, \quad i = 1, N \tag{7.1}$$

Let $\mathbf{x}_k = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_{K_k}\}$ denote all the (interior) image pixels that define the projection of region $\mathcal{R}_k$ in image I. We refer to $I_0 = T$ as the *reference image* and to the union of the projections of the model regions in T, $\cup_k T(\mathbf{x}_k)$ as the *reference template*. The goal of the tracking algorithm is to find the (camera) motion P_t that best aligns the image template with the current image I_t. A more precise formulation follows next. Refer to Figure 7.1 for an illustration of the tracking approach.

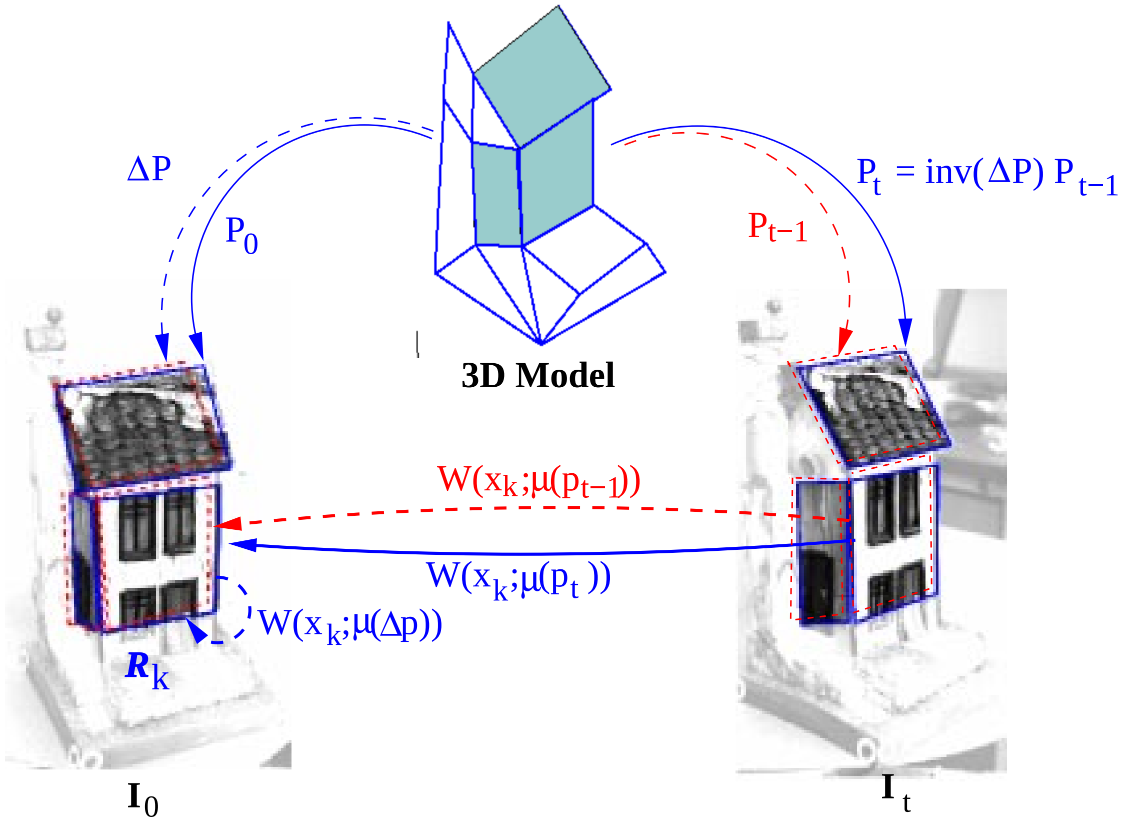


Figure 7.1: Overview of the 2D-3D tracking system. In standard SSD tracking 2D surface patches are related through a warp W between frames. In our system a 3D model is estimated (from video alone), and a global 3D pose change ΔP is computed, and used to enforce a consistent update of all the surface warps.

Assume that the image motion in image t for each individual model region k can be perfectly modeled by a parametric motion model $W(\mathbf{x}_k; \mu(P_t, \mathbf{Y}_k))$ where μ are 2D motion parameters that are determined by the projection of the region control points $\mathbf{y}_{tkj} = P_t \mathbf{Y}_{kj}$. As an example for a planar region the corresponding 4 control points in the template image and target image t define a homography (2D projective transformation) that will correctly model all the interior region points from the template image to the target image t. Note that the 3D model motion is global but each individual region has a different 2D motion warp W_k. For simplicity, the 2D warp is denoted by $W(\mathbf{x}_k; \mu(\mathbf{p}_t))$ where $\mathbf{p}_t$ are the 3D motion parameters that define the camera projection matrix P_t.

Under image constancy assumption [65] (e.g. no illumination change) the tracking problem can be formulated as finding $\mathbf{p}_t$ such as:

$$\cup_k T(\mathbf{x}_k) = \cup_k I_t(W(\mathbf{x}_k; \mu(\mathbf{p}_t))) \tag{7.2}$$

$\mathbf{p}_t = \mathbf{p}_{t-1} \circ \Delta\mathbf{p}$ can be obtained by minimizing the following objective function with respect to $\Delta\mathbf{p}$:

$$\sum_k \sum_x [T(\mathbf{x}_k) - I_t(W(\mathbf{x}_k; \mu(\mathbf{p}_{t-1} \circ \Delta\mathbf{p})))]^2 \tag{7.3}$$

For efficiency, we solve the problem by an inverse compositional algorithm [3] that switches the role of the target and template image. The goal is to find $\Delta\mathbf{p}$ that minimizes:

$$\sum_k \sum_x [T(W(\mathbf{x}_k; \mu(\Delta\mathbf{p}))) - I_t(W(\mathbf{x}_k; \mu(\mathbf{p}_{t-1})))]^2 \tag{7.4}$$

where in this case the 3D motion parameters are updated as:

$$P_t = \text{inv}(\Delta P) \circ P_{t-1} \tag{7.5}$$

The notation $\text{inv}(\Delta P)$ means inverting the 3D motion parameters in a geometrically valid way that will result in inverting the 3D motion, i.e. in the case when $\Delta P = K[R|\mathbf{t}]$ is a calibrated projection matrix, the inverse motion is given by $\text{inv}(\Delta P) = K[R'| - R'\mathbf{t}]$ (see Section 7.1.2). As a consequence, if the 2D warp W is invertible, the individual warp update is (see Figure 7.1):

$$W(\mathbf{x}_k; \mu(\mathbf{p}_t)) = W(\mathbf{x}_k; \mu(\Delta\mathbf{p}))^{-1} \circ W(\mathbf{x}_k; \mu(\mathbf{p}_{t-1})) \tag{7.6}$$

Performing a Taylor extension of equation 7.4 gives:

$$\sum_k \sum_x [T(W(\mathbf{x}_k; \mu(\mathbf{0}))) + \nabla T \frac{\partial W}{\partial \mu} \frac{\partial \mu}{\partial \mathbf{p}} \Delta\mathbf{p} - I_t(W(\mathbf{x}_k; \mu(\mathbf{p}_t)))] \tag{7.7}$$

Assuming that the 3D motion for the template image is zero (which can be easily achieved by rotating the model in order to be aligned with the first frame at the beginning of tracking), $T = T(W(\mathbf{x}_k; \mu(\mathbf{0})))$. Denoting $M = \sum_k \sum_x \nabla T \frac{\partial W}{\partial \mu} \frac{\partial \mu}{\partial \mathbf{p}}$, equation 7.7 can be rewritten as:

$$M\Delta\mathbf{p} = \mathbf{e}_t \tag{7.8}$$

where $\mathbf{e}_t$ represents the image difference between the template regions and warped image regions. Therefore the 3D motion parameters can be computed as the solution of a least square problem:

$$\Delta\mathbf{p} = (M^T M)^{-1} M^T \mathbf{e}_t \tag{7.9}$$

The stepest descent images $M = \sum_k \sum_x \nabla T \frac{\partial W}{\partial \mu} \frac{\partial \mu}{\partial \mathbf{p}}$ are evaluated at $\mathbf{p} = \mathbf{0}$ and they are constant across iterations and can be precomputed, resulting in an efficient tracking algorithm that can be implemented in real time (see Section 7.2.1).

Computing stepest descent images

We compute the stepest descent images from spatial derivatives of template intensities and inner derivatives of the warp. As mentioned before, the 2D motion parameters μ for a region k are functions of the 3D parameters $\mathbf{p}$, the 3D control points $\mathbf{Y}_j$ and the position of the control points in the template image $\mathbf{y}_{0j}$. The projection of the points in the current image $\mathbf{y}_j = P\mathbf{Y}_j$ are mapped to the template image control points through the 2D warp $W(\mu(\mathbf{p}))$ using:

$$\mathbf{y}_{0j} = W(\mu(\mathbf{p}))(P\mathbf{Y}_j), \quad j = 1, N \tag{7.10}$$

The warp W is a composed function, and its derivatives can be calculated as:

$$\frac{\partial W}{\partial \mathbf{p}} = \frac{\partial W}{\partial \mu}\frac{\partial \mu}{\partial \mathbf{p}}$$

The warp derivatives with respect to the 2D motion parameters are directly computed from the chosen warp expression (see Section 7.1.2 for some examples). The explicit dependency between the 2D parameters μ and the 3D motion parameters $\mathbf{p}$ is in general difficult to obtain, but equation 7.10 represents their implicit dependency, so $\frac{\partial \mu}{\partial \mathbf{p}}$ are computed using implicit function derivatives. Assume that equation 7.10 can be written in the form (see Section 7.1.2 for some examples):

$$A(\mathbf{p})\mu(\mathbf{p}) = B(\mathbf{p}) \tag{7.11}$$

Taking the derivatives with respect to each component p of $\mathbf{p}$:

$$\frac{\partial A}{\partial p}\mu + A\frac{\partial \mu}{\partial p} = \frac{\partial B}{\partial p} \tag{7.12}$$

For a given $\mathbf{p}$ value μ can be linearly computed from equation 7.11 and then $\frac{\partial \mu}{\partial p}$ is computed from equation 7.12.

7.1.2 Practically Useful Motion Models

Different levels of 3D reconstruction - projective, affine, metric Euclidean - can be obtained from an uncalibrated video sequence [59]. A projective reconstruction gives more degrees of freedom (15 DOF) so it might fit the data better under different noise conditions. On the other hand, fitting a metric structure will result in a stronger constraint, and fewer parameters can represent the model motion (6DOF). For our tracking algorithm we will investigate two levels of geometric models reconstructed under perspective camera assumption - projective and metric. The 3D motion of the model results in 2D motion of the regions $\mathcal{R}_k$ on the image plane.

As mentioned before, the 2D motion is determined through the regions' control points. Different motion approximations are common for the 2D-2D image warps.

Warps with few parameters (e.g 2D translation) are in general stable for small regions or simple motion. To better capture the deformation of a region, more parameters should be considered but in general tracking with these warps need large surface area or stabilization from a 3D model. A natural parametrization, which also correctly captures motion of planar regions, would be a homography warp for a perspective camera model (projective or Euclidean) and an affine warp for a linear camera model (orthographic, weak perspective, para-perspective). The next subsections give concrete examples of how the tracking algorithm can be applied to three types of useful combinations of motions: an Euclidean model with small translational patches, or larger homography patches, and a projective model with small translational patches.

Euclidean model with translational warps

A perspective calibrated camera has the following form in Euclidean geometry:

$$P = K[R|\mathbf{t}] \tag{7.13}$$

where the internal parameters are:

$$K = \begin{bmatrix} a_u & s & u_c \\ 0 & a_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \tag{7.14}$$

$R = R_x(\alpha_x)R_y(\alpha_y)R_z(\alpha_z)$ represents the rotation matrix and $\mathbf{t} = [t_x, t_y, t_z]^T$ is the translation vector. So the 3D motion parameters are $\mathbf{p} = [\alpha_x, \alpha_y, \alpha_z, t_x, t_y, t_z]$. A translational warp is controlled by one model points for each region and has the form:

$$W(\mathbf{x}_k; \mu) = \mathbf{x}_k + \mu \tag{7.15}$$

where $\mu = [\mu_x, \mu_y]^T$ is the 2D image translation vector and is computed from the motion of the control point $\mathbf{Y}_k$ using:

$$\mu(\mathbf{p}) = \mathbf{y}_{0k} - K[R|\mathbf{t}]\mathbf{Y}_k \tag{7.16}$$

The inner derivatives $\frac{\partial W}{\partial \mu}$ and $\frac{\partial \mu}{\partial \mathbf{p}}$ can be directly computed from equation 7.15,7.16 without needing the implicit function formulation.

Euclidean model with homography warps

The image motion of a planar patch can be modeled projectively using a homography warp that is determined by at least 4 control points $\mathbf{Y}_{kj}$. Denote the projection of the control points in the current image by $\mathbf{y}_{tj}$. Note that k is dropped as all the calculations are done for one region. With the Euclidean camera model,

$\mathbf{y}_j = K[R|\mathbf{t}]Y_j$. A homography can be represented using 8 independent parameters $\mu = [\mu_1\mu_2\mu_3\mu_4\mu_5\mu_6\mu_7\mu_8]^T$:

$$W(\mathbf{x}_k;\mu) = \begin{bmatrix} \mu_1 & \mu_2 & \mu_3 \\ \mu_4 & \mu_5 & \mu_6 \\ \mu_7 & \mu_8 & 1 \end{bmatrix} \mathbf{x} = H\mathbf{x} \tag{7.17}$$

The explicitly dependency of the 2D warp parameters as function of 3D motion parameters is difficult to obtain analytically in this case, but we can apply the method described in Section 7.1.1 to compute the inner derivatives $\frac{\partial \mu}{\partial \mathbf{p}}$ using the implicit dependency from equation 7.10:

$$\mathbf{y}_{0j} = H\mathbf{y}_j \quad j = 1, N \quad (N \geq 4) \tag{7.18}$$

which can be put in the form of equation 7.11 $A(\mathbf{p})\mu = B(\mathbf{p})$ with

$$A(\mathbf{p}) = \begin{bmatrix} y_1^1 & y_1^2 & 1 & 0 & 0 & 0 & -y_1^1 y_{01}^1 - y_1^2 y_{01}^1 \\ 0 & 0 & 0 & y_1^1 & y_1^2 & 1 & -y_1^1 y_{01}^2 - y_1^2 y_{01}^2 \\ \vdots & & & & & & \\ y_N^1 & y_N^2 & 1 & 0 & 0 & 0 & -y_N^1 y_{0N}^1 - y_N^2 y_{0N}^1 \\ 0 & 0 & 0 & y_N^1 & y_N^2 & 1 & -y_N^1 y_{0N}^2 - y_N^2 y_{0N}^2 \end{bmatrix} \tag{7.19}$$

$$B(\mathbf{p}) = [y_{01}^1, y_{01}^2, \ldots, y_{0N}^1, y_{0N}^2]^T \tag{7.20}$$

where $[y_j^1, y_j^2, 1]^T$ are the normalized homogeneous coordinates for $\mathbf{y}_j$.

Projective model with translational warp

This last example is very similar to the first one except that the 3D motion is represented by a projective 3×4 camera matrix P with 11 independent parameters $\mathbf{p} = [p_1 p_2 \ldots p_{11}]^T$. The 2D warp parameters μ are related to $\mathbf{p}$,:

$$\mu(\mathbf{p}) = \mathbf{y}_{0k} - P\mathbf{Y}_k \tag{7.21}$$

The translational warp is given by equation 7.15.

This model presents difficulties in calculating a unique and numerically stable inverse of the 3D motion, as required in equation7.5. To avoid this problem, while we still compute a global motion update $\Delta\mathbf{p}$ instead we update each warp independently as in equation 7.6. This solution is closer to the original SSD tracking algorithm [3, 56] and, as demonstrated by the experimental results, performs worse than our new algorithm described in Section 7.1.1, but still better than the simple unconstrained SSD tacker.

The 3D model-based tracking is incorporated into a complete tracking system, described in Section 7.2.1 that first acquires the geometric model of the scene using simple 2D SSD translational tracking and then continues tracking the model with our new algorithm.

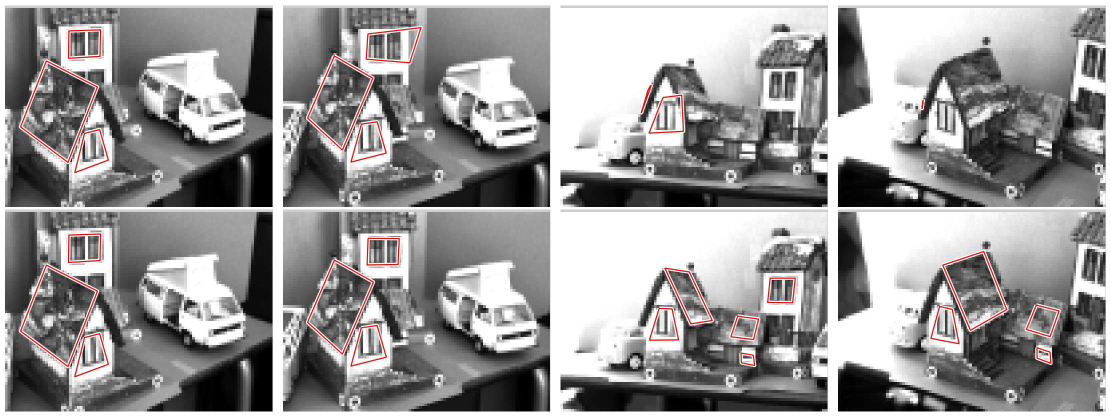


Figure 7.2: **Top** Tracking individual patches using a homography. Not all regions can be tracked thought the whole sequence and occlusion is not handled. **Bottom** Through the 3D model each region motion is rigidly related to the model, and tracking succeeds through the whole sequence. The model also allows detection and removal of occluded regions and introduction of new regions.

7.1.3 Experimental Results

Two important properties of tracking methods are convergence and accuracy. Tracking algorithms based on an optimization and spatio-temporal derivatives (equation 7.7) can fail to converge because the image difference between consecutive frames is too large, and the first order Taylor expansion around $\mathbf{p}_{t-1}$is no longer valid, or some disturbance causes the image constancy assumption to be violated.

In numerical optimization, the pose update $\Delta\mathbf{p}$ is computed by solving an overdetermined equation system, equation 7.8. Each pixel in a tracking patch provides one equation, and each model freedom (DOF) one variable. The condition number of M affects how measurement errors propagate into $\Delta\mathbf{p}$, and ultimately, if the computation converges or not. In general, it is more difficult to track many DOF. In particular, models which cause very apparent image change, such as image plane translations are easy to track, while ones with less apparent image change such as scaling and out-of-plane rotations are more difficult. A general plane-plane transform such as homography contains all of these and tend to have a relatively large condition number. By tracking a 3D model, the convergence is no longer solely dependent on one surface patch alone, and the combination of differently located and oriented patches can give an accurate 3D pose estimate.

In Figure 7.2 planar regions in the image sequence are tracked using an 8DOF homography. When each patch is tracked individually (top images) the first region is lost already after 77 frames and all lost after 390 frames. The condition number for M varies between $5 * 10^5$ and $2 * 10^7$, indicating a numerically ill conditioned situation. When instead the regions are related by the global 3D model, pose is

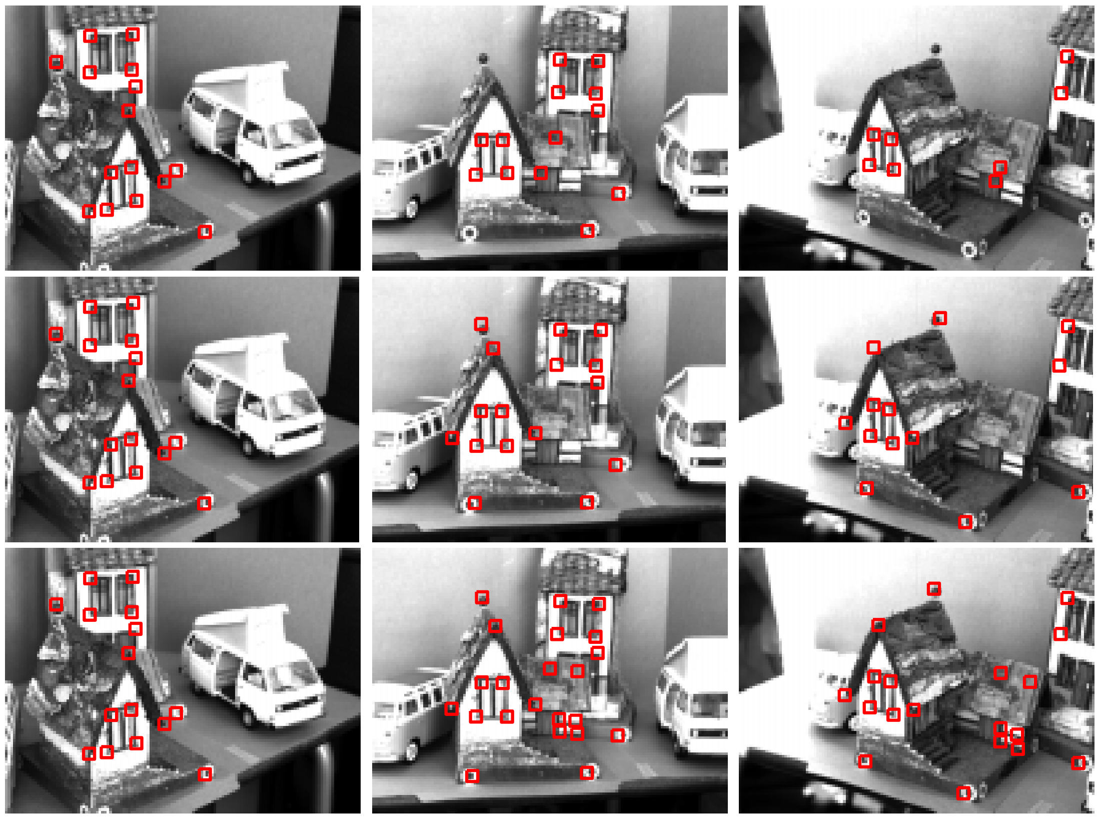


Figure 7.3: **Top** Translation tracking of individual regions. Though the video sequence many patches are lost. **Middle** A projective 3D model is used to relate the regions, and provide more stable tracking through the whole sequence. **Bottom** An Euclidean model relates the region, and also allow the introduction of new regions.

successfully tracked through the whole sequence of 512 frames. Additionally the model allows the detection and removal of the region on the left roof side when it becomes occluded and the introduction of three new regions on the right roof side and the smaller outhouse when they come into view. The condition number of the 6DOF (3 rot, 3 trans) model is 900, which is significantly better than the 8DOF homography.

The next experiment uses a simpler 2DOF translation model to relate regions as described in Sections 7.1.2 and 7.1.2, and either an Euclidean or Projective global 3D model. In Figure 7.3 three cases are compared. In the first, (figure top) no model is is used, and almost half of the region trackers are lost starting already from frame 80. Because only 2D spatial x and y derivatives are used in M the condition number is very low at an average 1.3. In the middle sequence, a projective model is used to relate the regions. This stabilizes the tracking until about frame 400, where one tracker is slightly off target and further about 430 some are lost due to occlusion. The projective model has 11 DOF and the condition number is quite high at $2 * 10^4$. In the final (figure bottom) sequence a Euclidean model relates the trackers, and provides handling of occlusions. The condition number is a reasonable 600, and the whole 512 frame sequence is successfully tracked.

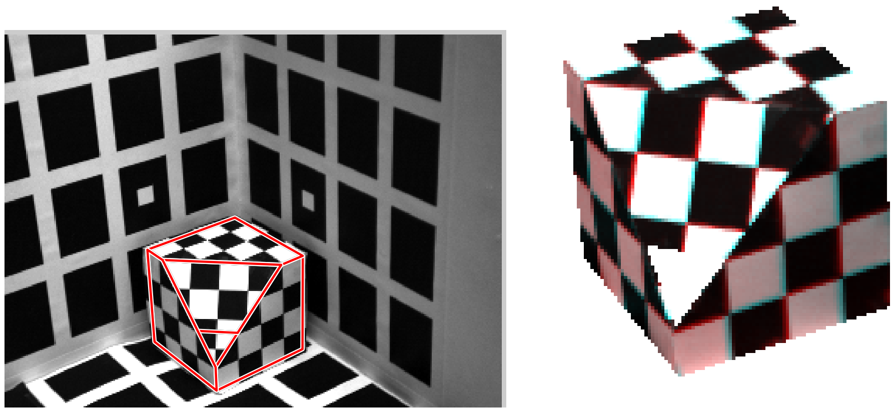


Figure 7.4: Accuracy experiment. **Left** Scene image. Right current image and template superimposed indicate accurate alignment

In the final experiment we evaluate the accuracy by tracking four sides of a cube textured with a calibration pattern, see Figure 7.4. For each frame we superimpose the warped image onto the template, displaying each using separate color channels. Hence any misalignment can be seen as color bands between the white and black squares. We measured the width of these bands and found that for the Euclidean model with homography warps misalignments were less than 1 pixel on average , and worst case over several hundred frames was 2 pixels. The 3D model constrain the relative location of the regions, and inaccuracies in the estimated model cause the (small) misalignments we observed. In the case of no 3D model and tracking with homography warps alone, the regions would eventually lose track (as in the previous house sequence). However, for the frames that did converge alignment was very good, with an error significantly less than a pixel. This is to be expected since the homography parameters allow exactly the freedoms needed to warp planar surfaces into any camera projection.

7.1.4 Discussion on the Tracking Algorithm

We have shown how a 3D scene model, estimated from images alone, can be integrated into SSD region tracking. The method makes tracking of a-priori unknown scenes more stable and handles occlusions by removing and introducing tracking regions as appropriate when new views become available.

In combining different types of 3D global models and 2D region warps we found that

- Tracking planar regions using an 8DOF homography without a 3D model is unstable due to the many DOF estimated, but limited image signature available from geometric change of only one planar patch.

- On the other hand, using the estimated 3D model we constrain multiple individual patches to move in a consistent way and achieve very robust and stable tracking of full 3D pose over long sequences.

- With some loss in generality and magnitude of maximum trackable pose change, the 8DOF homography can be replaced by simple 2DOF translational trackers. Each such individual tracker has to use only a small image region since it doesn't deform projectively, but instead many regions can be used. Using 2DOF regions and either an Euclidean or projective 3D model this gives almost as good tracking as the homography + 3D model, and makes execution somewhat faster.

Convergence in the last case (translational only warp) over large angular changes in camera viewpoint can be improved by using a few view-dependent templates, each associated with a smaller angular range, and switch these in and out depending on the current angular pose computed from the 3D model. While this introduces a risk for drifts and errors from the templates being slightly offset, in practice we have found it works well using 5-10 different templates over the visible range of a patch.

Visual tracking has many applications in e.g. robotics, HCI, surveillance and model building. Tracking and modeling are interrelated in that (as we have shown) a model improves tracking, and tracking can also be used to obtain the image correspondences needed for a model. In unstructured environments this used to present a chicken-and-egg problem: Without a model it was difficult to track, and without tracking one couldn't obtain a model. Our method integrates both into a system which is started by defining regions to track in only a 2D image. First 2D tracking is used over an initial video segment with moderate pose change to obtain point correspondences and build a 3D model from image data. After the model is built, the system switches to 3D tracking and is now ready to handle large pose changes and provide full 3D pose (rotation, translation) tracking.

A main feature of our method is that 3D pose change ΔP is computed directly from image intensity derivatives w.r.t. P. Note that this guarantees the best 3D pose update available from the linearized model (here using L_2 norm, but other e.g. robust norms are also possible [56]). This is unlike the more common approach of first tracking 2D image correspondences, and then computing a 3D pose from points, where each 2D point location is first committed, based on a locally optimal image fit but without regards to the global 3D constraints.

7.2 IBR System Implementation

This section presents the practical implementation aspects for putting together the rendering system. The IBR system, as described in Section 6.5 consists of three main parts:

1. A real-time tracking program interfaces to a digital camera and uses XVision real time tracking and our 3D model-based tracking algorithm to maintain point correspondences in the video sequence. It also grabs sample frames of intensity images.

2. A structure and texture editor is used to view and verify the geometric model, as well as provide an interface to control the texture selection and generation.

3. A real-time renderer takes a processed models and renders it under varying virtual camera pose controlled by mouse input.

7.2.1 Video Capture and Tracking

The model training data consists of an image sequence and tracked feature locations. We use XVision [56, 57] to set up a video pipeline and implement real time tracking. The tracking program can connect to either consumer IEEE 1394 standard web cams in yuv422 mode or to higher quality machine vision cameras, (we use Basler A301fc) in raw Bayer pattern mode. To initiate tracking, the user selects a number (about a dozen or more) high-contrast regions through a user interface as illustrated in Figure 7.5. The regions are grabbed, and tracked by estimating affine image variability from the input video images. Most scenes are difficult to tile fully with trackable regions that are required to be close to planar, while it is possible to successfully track small subparts. For the dynamic texture model, larger quadrilaterals are formed over the whole scene. From these quadrilaterals, the textures are sampled at rates from 2 to 5 Hz, while the trackers run in the background at about 30Hz (for up to 20 trackers).

For capturing larger objects and scenes, trackers can easily become occluded. The 3D model based tracking system presented in Section 7.1 was integrated with the model acquisition to control the trackers' appearance/disappearance. The system initializes the model from 2D image tracking over a limited motion in an initial video segment and then switches to track and refine the model using 3D model based tracking. The main steps in the implemented method are:

1. Several salient surface patches are selected in a non-planar configuration from a scene image and tracked for about 100 frames using standard (planar) XVision SSD trackers.

2. From the tracked points a 3D model is computed using structure from motion and the stratified uncalibrated approach (as in Section 6.3).

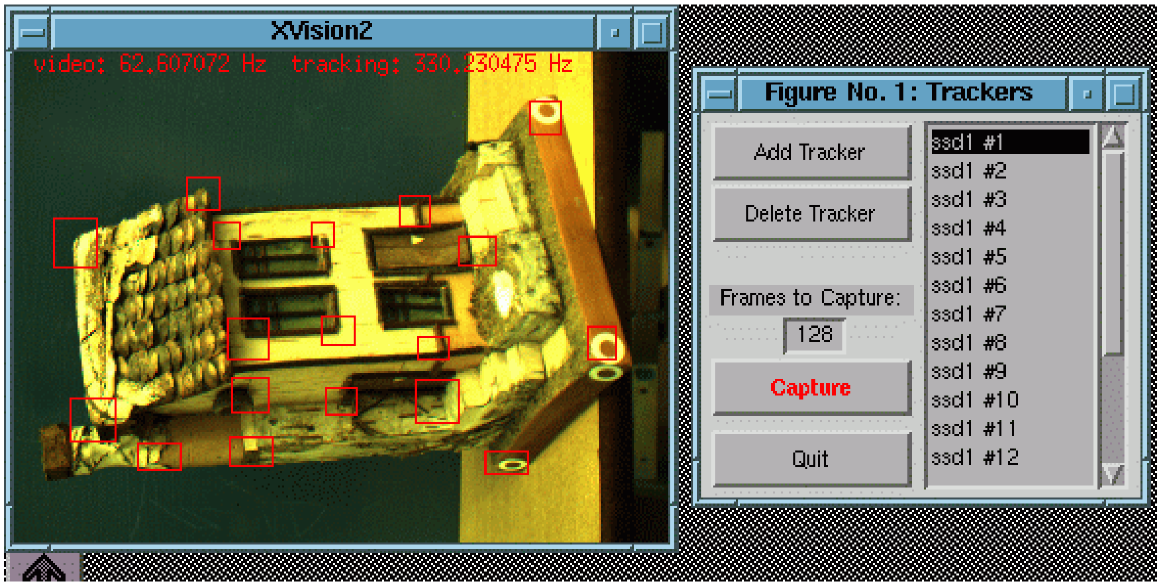


Figure 7.5: Video capture and tracking user interface

3. The 3D model is related to the start frame of 3D tracking using the 2D tracked points and camera matrix computed using resection (non-linear for accuracy) from 2D-3D correspondences. Then the model based tracking algorithm is initialized by computing the stepest descent images at that position. The tracking is now continued with the 2D surface patches integrated in the 3D model that enforces a globally consistent motion for all surface patches.

4. New patches visible only in new views are added by first tracking their image projection using 2D tracking then computing their 3D coordinates through intersection in $n \geq 2$ views then incorporate them in the 3D model. In the current implementation the user specifies (clicks on) the image control points that will characterize the new surfaces but in the future we plan to automatically select salient regions.

7.2.2 Structure and Texture Editor

We designed an user interface that assists the geometric model generation. The editor has three modes - structure editing, texture coordinates editing and a blue-screening.

The first mode, presented in Figure 7.6 assists model building. The editor is designed for both projective or affine reconstruction. The accuracy of the captured geometry can be evaluated by comparing the tracked points (blue circles and numbers) to the reprojected points (green circles), while stepping through the captured images by clicking or dragging the image slider at the bottom. If they differ significantly (as marked in magenta in the editor), the likely cause is that

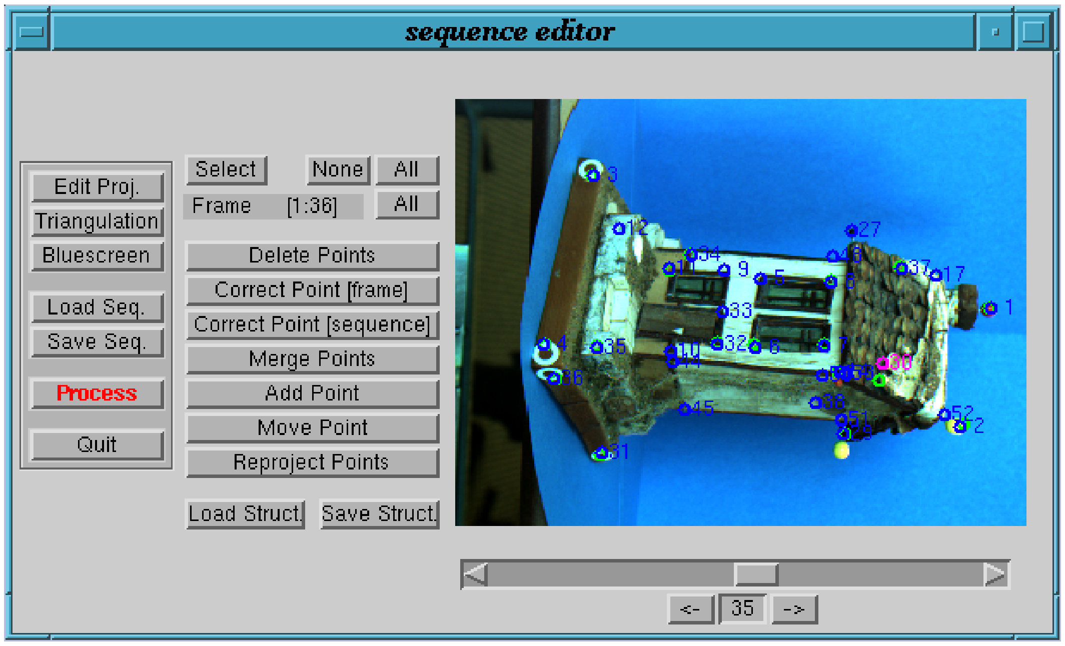


Figure 7.6: Structure editor mode

the real time tracking lost the point for some frames. The editor allows *correcting* or *deleting* the erroneous points in one or a range of frames. After getting a robust structure new points can be *added* through intersection by clicking in a number of frames (5-10) equally spread through a selected set of frames. We implemented an additional mode for adding points that introduces a new points by *moving* an existing one along a line defined by that point and an additional selected point. This might be useful when the user wants to introduce a point that does not have a distinctive signature in the image. If a point was not tracked in a number of frames but is still visible, its image position can be added by *reprojection*. In some situations, a point lost track but it was reintroduced by the user in the subsequent frames. The tracking program will consider it a new point. Using the editor the point can be *merged* with the original one and considered the same model point.

In the second mode, the triangulation used to extract and represent the dynamic texture can be modified, Figure 7.7. A captured structure is initially triangulated using standard Delauney triangulation. This triangulation often does not put triangles to best correspond to the real scene surfaces. In the triangulation editor unfortunate triangles can be *deleted* and new triangles can be *added* by clicking on three points. Additionally, the resolution or relative area of texture given to a particular triangle can be modified by clicking and dragging the triangles in the "texture editor" window. The "opengl" window shows the actual texture representation for the image frame selected.

In the third mode, a blue screen color can be selected by clicking in the image. Pixels with this (or close) color will be made transparent. This is useful when

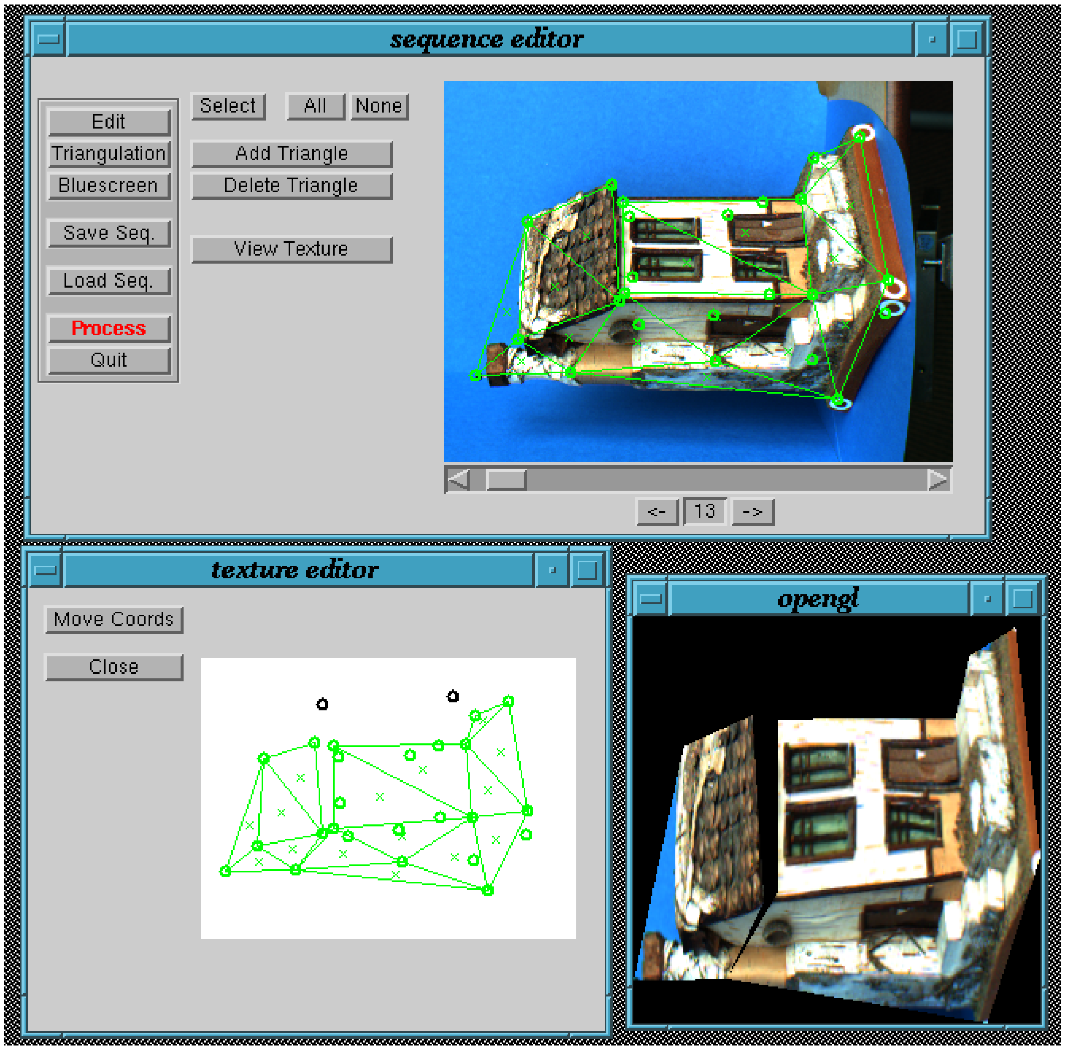


Figure 7.7: The triangulation and texture editor

rendering objects without the original background. A useful application [24] , not presented in this thesis is an augmented reality system when captured objects are inserted into real scenes.

7.2.3 Real-time Texture Blending

To render the *dynamic texture* we use the texture blending features available on most consumer 3D graphics cards. These graphics accelerators can blend textures very efficiently, however they are very restrictive in terms of the types textures that can be used, making it somewhat complicated to hardware accelerate this type of rendering.

Unsigned Basis

The rendering hardware used is designed for textures containing positive values only, while the spatial basis, equation 6.44 is a signed quantity. We rewrite this as a combination of two textures with only positive components:

$$I_w(t) = B^+\mathbf{y}(t) - B^-\mathbf{y}(t) + \bar{T}$$

Where B^+ contains only the positive elements from B (and 0 in the place of negative elements) and B^- contains the absolute values of all negative elements from B. When blending, some textures will be added, and others subtracted.

Quantization

Graphics cards generally require textures to be represented as bytes in the range 0-255, and after each blending operation (addition or subtraction of a basis texture) values are clipped to this range. This can be problematic since neither our basis textures nor the intermediate values when combining basis textures are will have values within the required range. The only guarantee is that the final result will be within that range. We scale the basis textures and coefficients to fit within this range as follows:

$$\begin{aligned}\tilde{B}^+ &= 255B^+\zeta^{-1}\\ \tilde{B}^- &= 255B^-\zeta^{-1}\\ \tilde{\mathbf{y}} &= 255^{-1}\zeta\mathbf{y}\end{aligned}$$

Where ζ is a diagonal matrix of the maximum absolute values from the columns of B. ($\zeta = \mathrm{d}iag(\max|B|)$)

Now $\hat{B}^+$ and $\hat{B}^-$ are both in the range 0-255, and can be used as textures in hardware. The problem regarding overflow in intermediate blending stages cannot be completely solved, but by drawing the mean image first, and then alternately adding and subtracting scaled eigenvectors, overflow is avoided in most cases.

Rendering of each frame is performed as in the following pseudo-code.

```
// draw the mean
BindTexture($\bar{I}$);
DrawTriangles();

// add basis textures
for(each $i$)
{
  SetBlendCoefficient($|\tilde{\mathbf{y}}_i(t)|$);

  BindTexture($\tilde{B}_i^+$);
  if($\tilde{\mathbf{y}}_i(t) > 0$) SetBlendEquation(ADD);
  else SetBlendEquation(SUBTRACT);
  DrawTriangles();

  BindTexture($\tilde{B}_i^-$);
  if($\tilde{\mathbf{y}}_i(t) > 0$) SetBlendEquation(SUBTRACT);
  else SetBlendEquation(ADD);
  DrawTriangles();
}
```

Using the HW accelerated implementation we can obtain a frame rate of about 50 Hz using a 2 year old GeForce 3 graphics card, and about 25-30 Hz running on a basic GeForce 2-to-go in a 1GHz laptop.

The real time renderer reads several files from the current directory, and starts a glut window, where the scene or object viewpoint can be interactively varied using the mouse.

7.3 Model Evaluation and Examples

This section presents a qualitative as well as quantitative evaluation of the geometric model with dynamic texture. First, the two geometric models presented in Section 6.2 (affine metric model) and Section 6.3 (Euclidean model for a perspective projection) are compared. Next we present some examples of renderings for different objects as well as part of a room, followed by a quantitative evaluation of the dynamic texture performance.

7.3.1 Geometric Model Accuracy

For evaluating the model reconstruction accuracy for affine and projective case, we built a synthetic room structure with 3 walls and 240 feature points and generated about 200 images from different camera positions shown with red for position and green line for viewing direction in Figure 7.8. The images are cropped to an image plane of 240×320 pixels.

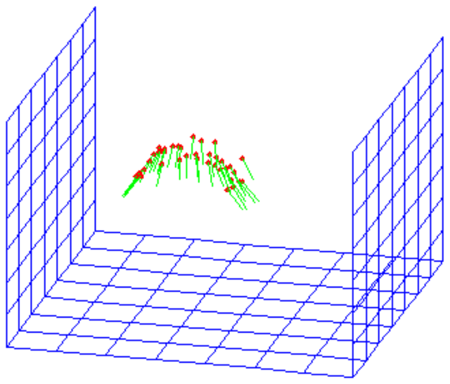


Figure 7.8: Synthetic room data and camera positions. Red dots indicate the camera positions and green lines indicate viewing direction

In the first experiment we vary the focal length. Figure 7.9 shows the reconstructed geometry for the affine case (top) and projective case (bottom) with focal length respectively of $50, 100$ and 200 pixels. The numerical results for reprojection error and range of motion recovered are summarized in Table 7.1. The affine SFM algorithm recovers only an image plane translation that cannot be compared with the original 3D translation. As expected, the affine structure accuracy improves (Figure 7.9 top right) when increasing the focal length, but it is still far from the original structure. The reprojection error is reasonable (4 pixels for $f = 200$) that indicates that the model fits the image data but by not being constrained to an Euclidean structure it is not straight. Note that the range of the recovered horizontal rotation is very small compared to the original motion. By contrast, the projective model fits the data almost perfect and the recovered structure and motion is also close to the original ones, and is not dependent on the focal length. From a rendering point of view, only the reprojection points meters, and if the structure is visualized through the same type of camera geometry in a motion range similar to the recovered one, it will "look good". Therefore, if the affine structure is loaded into a traditional graphics rendering program that assumes that the structure is Euclidean and it projects it with a perspective camera it will appear deformed. The same arguments also holds for a robotic application. In most situations, the robot has to be controlled in a metric space, so the projective structure would be more appropriate.

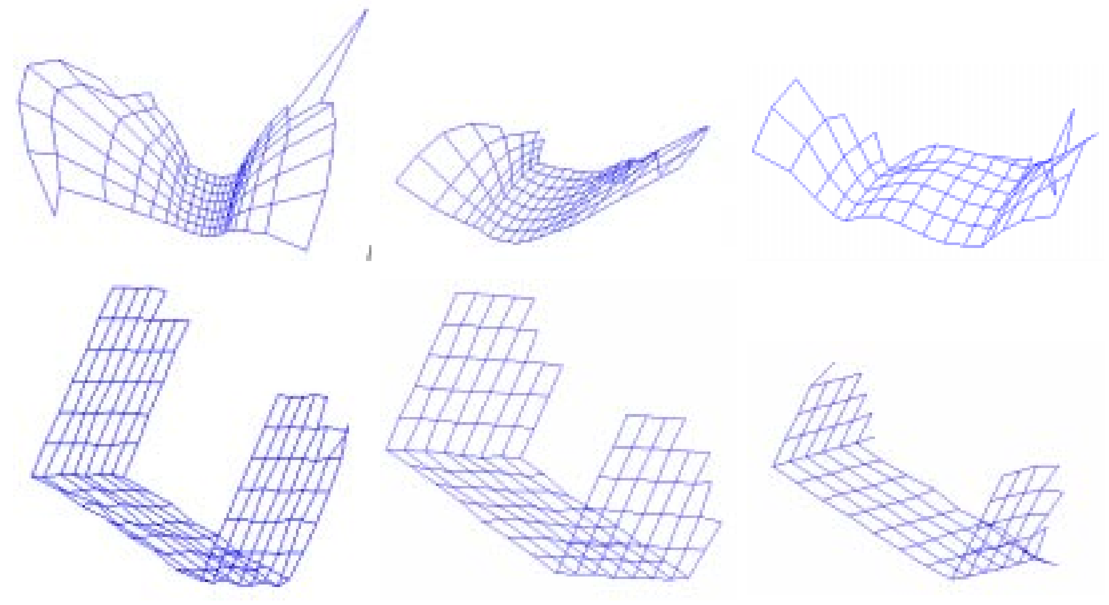


Figure 7.9: Reconstructed affine (top) and projective upgraded to Euclidean (bottom) structure under varying focal length. f = 50 (left), f = 100 (middle), f = 200 (right)

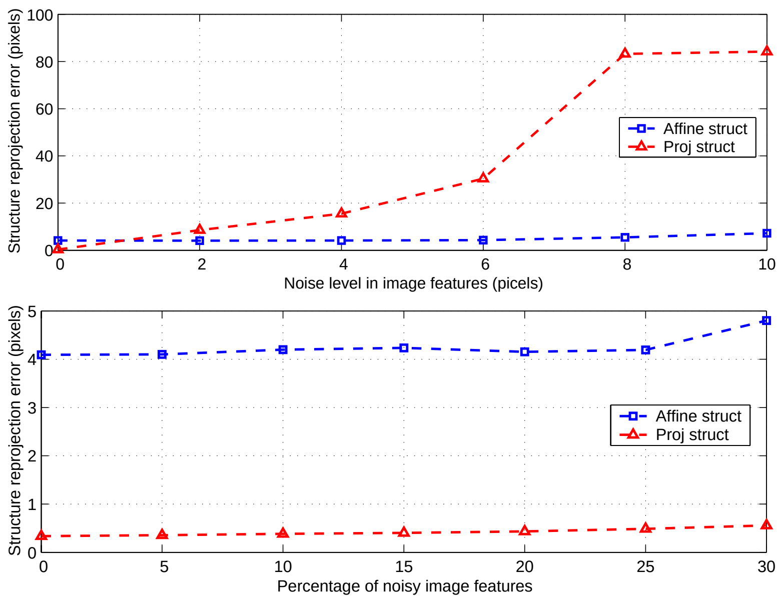


Figure 7.10: Accuracy of reconstructed affine and projective Euclidean structure as dependent of noise level with 10% outliers (top) and number of outliers with a noise level of 2 pixels (bottom).

Model	f (pix)	Reproj. err.	Range rot. (deg)			Range tr.		
			x	y	z	x	y	z
original			60	20	0	297	293	293
afine	50	8.46	1.46	0.38	0.2			
	100	7.87	12.94	9.37	0.4			
	150	7.83	7.91	12.46	0.5			
	200	4.12	3.20	22.94	0.2			
proj.	50	0.35	57.23	19.44	0.24	308.47	277.30	295.03
	100	0.34	57.33	17.65	0.63	301.18	290.91	289.57
	150	0.37	59.00	19.43	0.35	299.84	295.97	288.59
	200	0.34	58.74	19.29	0.8	306.53	279.91	296.84

Table 7.1: Pixel reprojection error (pixels) and recovered motion range (degrees for rotation and original structure units for translation). The affine SFM algorithm recovers only an image plane translation that cannot be compared with the original 3D translation.

In the second experiment we studied the influence of image feature noise on the reconstructed structure accuracy. There are two aspects that were considered: the number of outliers (noisy features) and the level of noise. Figure 7.10 shows the results. It was found that the projective upgraded to Euclidean structure is more sensitive to noise level than the affine structure (figure top). This is probably due to the number of constraints imposed to the Euclidean structure and therefore the difficulty to fit a noisy data. For this, in case when it is known that the data has outliers, a robust estimator (RANSAC) is recommended. When increasing the number of outliers the accuracy of both structures decreased.

In a real situation, when the structure is reconstructed from camera images, there are more errors that can decrease structure accuracy (calibration errors, quantization errors, tracking errors) and the reconstructed model is only the best fit to the given data. Figure 7.11 shows an example of the reconstructed structures for a dinning room. The top pictures show two examples from the training images overlapped with the triangulated structure. On the middle row is shown the reconstructed affine structure and on the last row the reconstructed projective structure. Next section shows how the dynamic texture is correcting geometric mistakes and compensating for the sparsity of the model (see Figure 7.15).

7.3.2 Dynamic Texture Rendering Examples

We have tested our method both qualitatively and quantitatively by capturing various scenes and objects and then reanimating new scenes and motions using dynamic texture rendering.

Many man-made environments are almost piece-wise planar. However, instead of making a detailed model of every surface, it is more convenient to model only

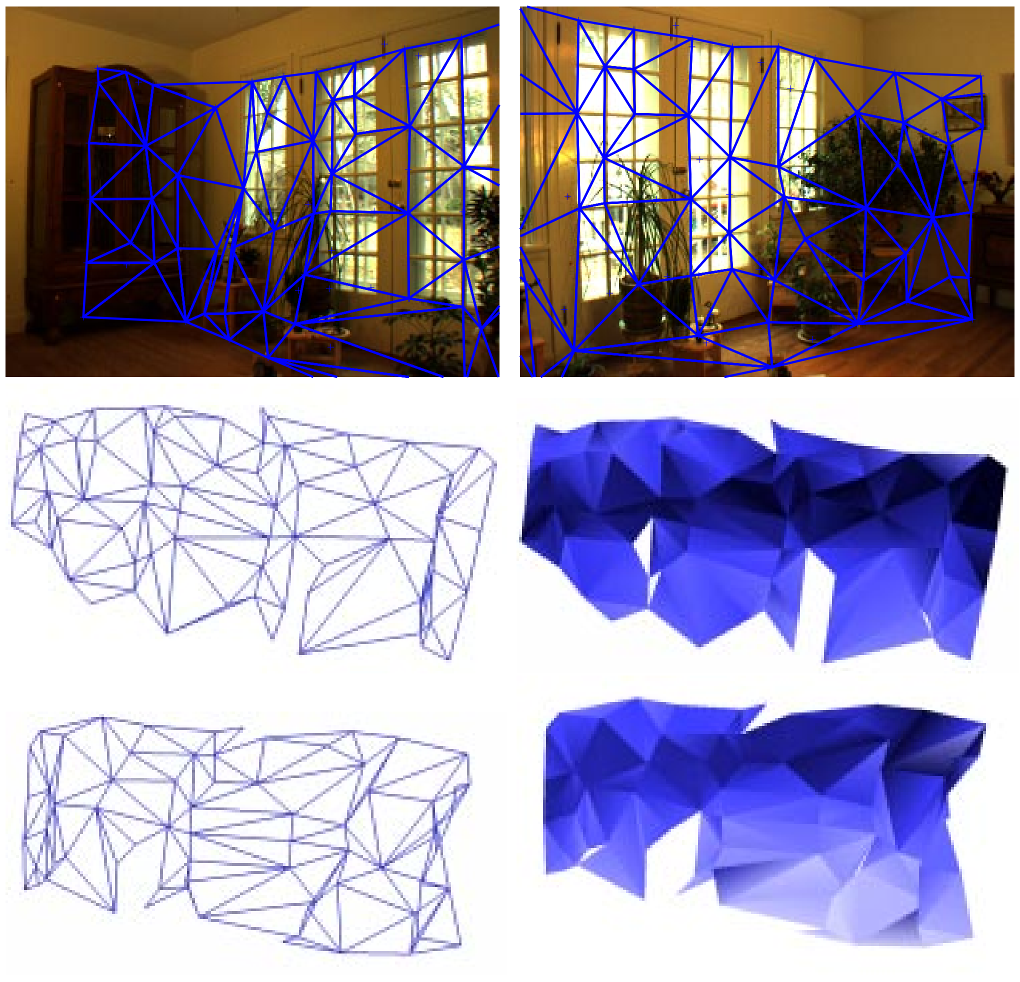


Figure 7.11: Illustration of recovered structure from real data. (top) two of the original images overlapped with a Delaunay triangulation of the projected structure. (middle) recovered affine structure (same triangulation); (bottom) recovered projective Euclidean structure.

the large geometric structure, e.g. the walls and roofs of houses, and avoid the complexity of the details, e.g. windows, doors, entry ways, eaves and other trim. But, when using automatic image point tracking and triangulation the resulting triangles sometimes do not correspond to planar house walls (Figure 7.12 left). Using standard texture mapping on this geometry we get significant geometric distortions (Figure 7.12 right). In Figure 7.13 these errors have been compensated for by modulating the texture basis to correct for the parallax between the underlying real surfaces and texture triangles.

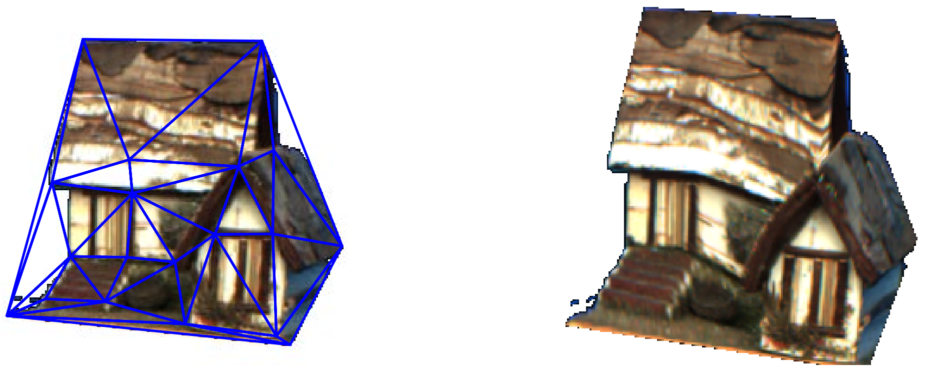


Figure 7.12: Left: Delauney triangulation of a captured house model. Note that triangles don't correspond well to physical planes. Right: Static texturing of a captured house produces significant errors. Especially note the deformations where the big and small house join due to several triangles spanning points on both houses.

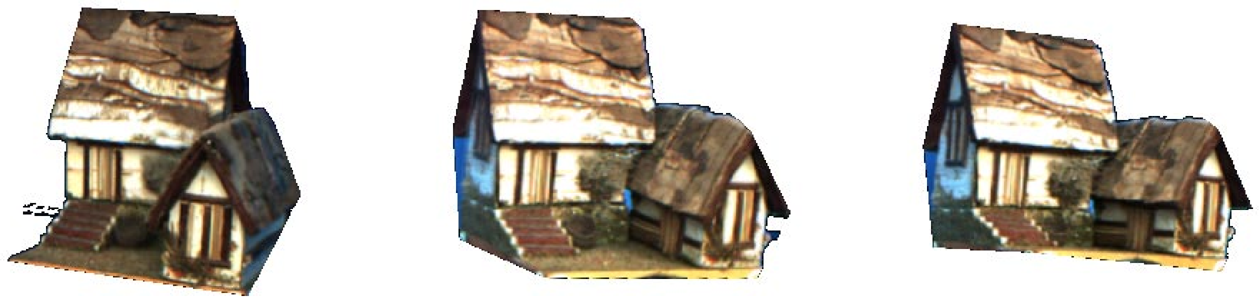


Figure 7.13: Rendered novel views of a house by modulating a texture onto a coarse captured geometric model. Note the absence of geometric distortions compared to the previous figure.

Unlike man-made scenes, most natural environments cannot easily be decomposed into planar regions. To put our method to test, we captured a flower using a very simple geometry of only four quadrilaterals. This causes a significant residual variability in the texture images. A training sequence of 512 sample images from motions with angular variation of $\mathbf{r} = [40, 40, 10]$ degrees around the camera u- v- and z-axis respectively. A texture basis of size 100 was estimated, and used to render the example sequences seen in Figure 7.14.

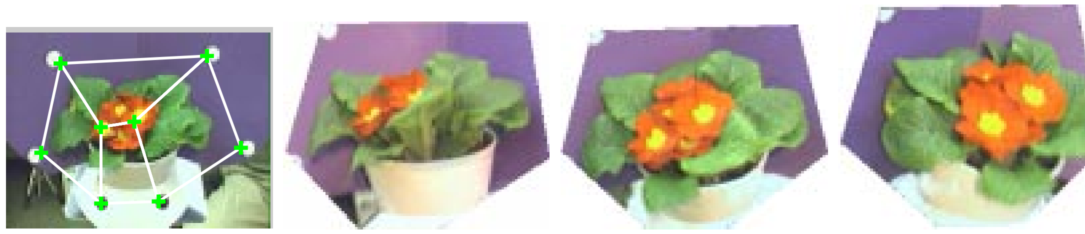


Figure 7.14: Flower sequence: (left most) One of the original images and the outline of the quadrilateral patches

A more complex model that we capture is the dinning room model presented in Figure 7.11. The affine model (figure top) has big distortions that cannot be totally compensated by the dynamic texture especially when the position is far from the original samples (figure top left). The projective model (figure bottom) captures the true geometry of the room better and, despite the sparsity of the geometric model, it generates good renderings for almost all viewing angles. The geometry of the scene was not concurrently visible but built by incrementally adding more points/views through camera intersection/resection to an initial model. In processing the texture for the model, it was divided in three regions to avoid compressing null texture when not necessary.

Quantitative comparison

In order to quantitatively analyze how modulating a texture basis performs compared to standard view dependent texturing from a close real image, we produced three image sequences. The first image sequence are 80 real scene images of a wreath viewed under different camera poses from straight on to approximately 50 degrees off axis. The second image sequence is a synthesized rendering of those same poses from the texture basis (Figure 7.16). The third is the same rendering using standard view dependent textures from 30 sample textures quite close (at most a few degrees from) the rendered pose. The average image intensity error per pixel between rendered and real images was calculated for sequence two and three. It was found that for most views modulating a basis texture we can achieve about half the image error compared to standard view dependent texturing. This error is also very stable over all views, giving real time rendering a smooth natural appearance. The view dependent rendering from sample images did better only when a rendered frame is very close to a sample image, and otherwise gave a

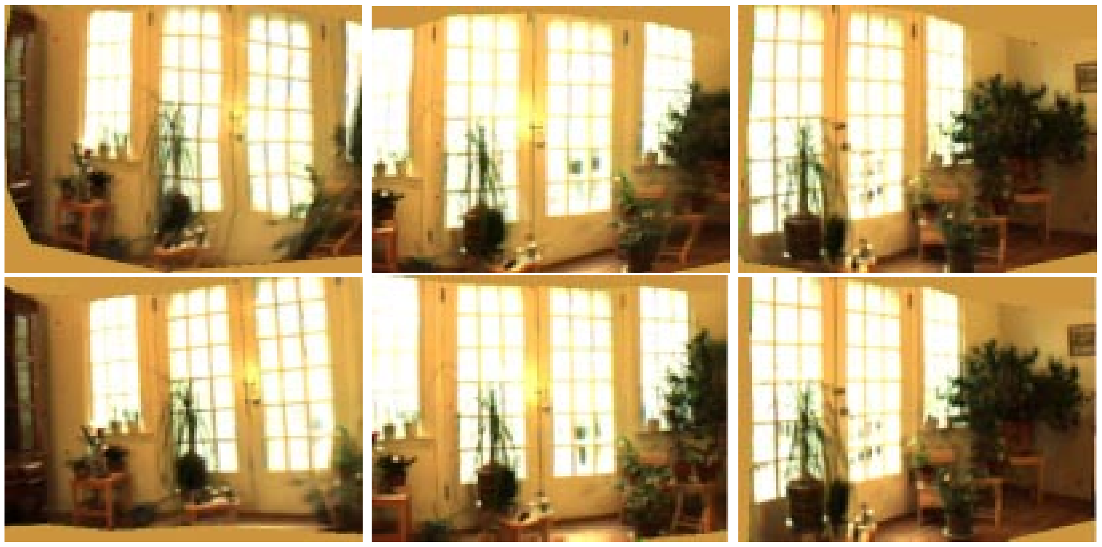


Figure 7.15: Examples of renderings for dinning room model from Figure 7.11. (top) example renderings of the affine model (bottom) example renderings of the projective model

jumpy appearance where the error would go up and down depending on the angular distance to a sample view. The error graph for 14 of the 80 views is shown in 7.17.

In animation there are global errors through the whole movie that are not visible in one frame but only in the motion impression from the succession of the frames. One important dynamic measurement is motion smoothness. When using static texture we source the texture from a subset of the original images ($k+1$ if k is the number of texture basis) so there is significant jumping when changing the texture source image. We tracked a point through a generated sequence for the pattern in the two cases and measure the smoothness of motion. Table 7.2 shows the average pixel jitter.

	Vertical jitter	Horizontal jitter
Static texture	1.15	0.98
Dynamic texture	0.52	0.71

Table 7.2: Average pixel jitter

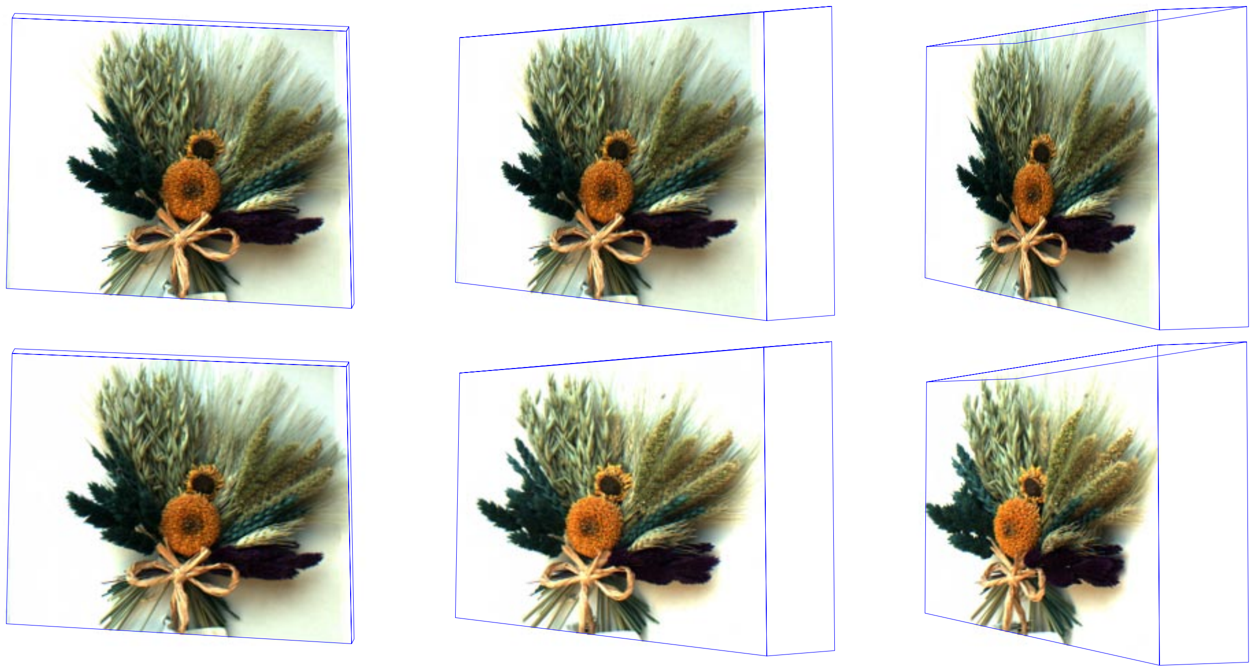


Figure 7.16: Texturing a rotating quadrilateral with a wreath. Top: by warping a flat texture image. Bottom: by modulating the texture basis B and generating a continuously varying texture which is then warped onto the same quad. Demo on web site

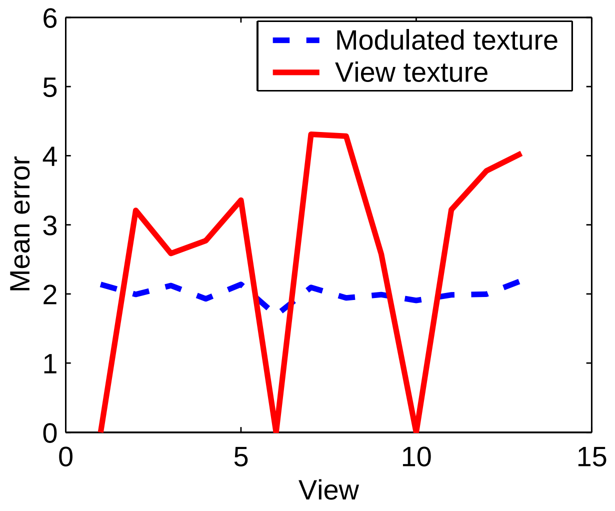


Figure 7.17: Pixel intensity error when texturing from a close sample view (red) and by modulating the texture basis. For most views the texture basis gives a lower error. Only when the rendered view has the same pose as the one of the three source texture images (hence the IBR is a unity transform) is the standard view based texturing better

7.4 Tracking and Predictive Display for a Indoor Robot Environment

In order to demonstrate the usability of uncalibrated model capture, tracking and rendering, the model based tracking algorithm presented in Section 7.1 was incorporated into a predictive display system designed for generating synthetic images of the current robot view for a remote operator (Figure 7.18). The system is similar to the one presented in Section 5.4 and can be summarized as follows:

```
(1) initialize robot pose
for each time step
      **robot site:**
         (2) track robot pose $\mathbf{p} = (\alpha_x, \alpha_y, \alpha_z, t_x, t_y, t_z)$
         (3) send position to operator site
      **user remote site:**
         (4) add current operator motion command
         (5) project geometric model in new location
                 $\mathbf{x} = K[R(\alpha_x, \alpha_y, \alpha_z)|\mathbf{t}(t_x, t_y, t_z)]\mathbf{X}$
         (6) compute the dynamic texture $T$ for the new location
         (7) warp texture onto the projected structure and display to operator
         (8) send motion command to robot site
end for
```

The dynamic model (geometry and dynamic textures) is stored at both robot and operator site. At the beginning (**step 1**), the operator selects the model's point/patch locations in the initial view in order to calculate the starting position with respect to the model position. In subsequent frames, the position is automatically tracked using the 3D SSD tacking algorithm (**step 2**). As the robot is controlled in an Euclidean space, we used the Euclidean 3D model with a translational or homography warp (Sections 7.1.2). The current robot location is transmitted to the operator site (**step 3**), where the current operator motion command is added (**step 4**) and a synthesized view is computed using the rendering algorithm described in Section 6.5 (**step 5-7**). The new desired position is transmitted back to the robot site and converted into motion commands that will move the robot toward the desired view (**step 8**).

7.4.1 Experimental Results

To evaluate the tracking with predictive display system, we captured a model of a research lab. We used the model based tracking algorithm from Section 7.1 to recover camera location along two motion trajectories.

The first trajectory was a straight line in the horizontal plane of about 1m. Figure 7.19 (left) illustrates the recovered trajectory. For measuring the accuracy of the tracking algorithm we calibrated the 3D room model assuming some given

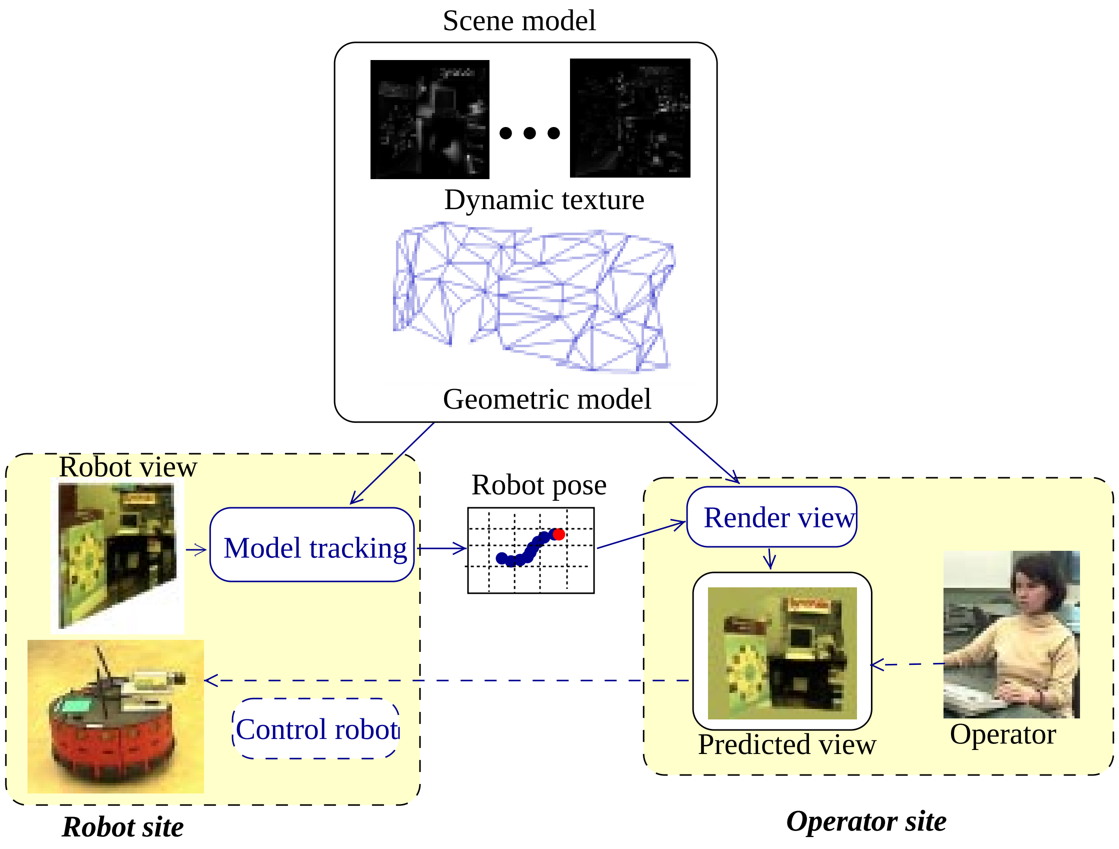
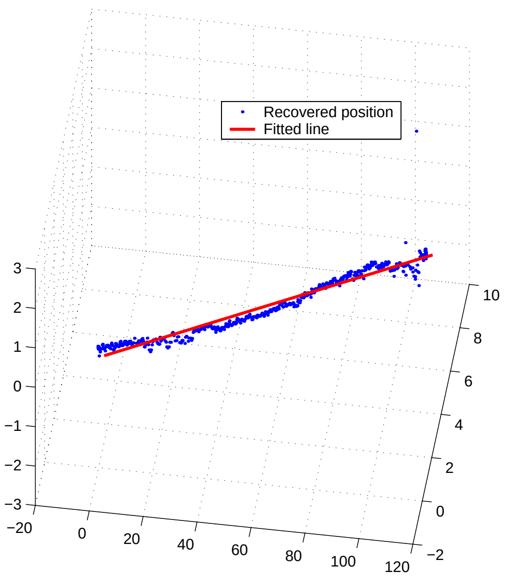


Figure 7.18: Overview of the tracking with predictive display system.

real dimensions (e.g. the size of the monitor) so we could get the translation in actual metric measurements (m). We found that the trajectory had about 0.95 cm mean deviation from a line and 5.1 cm mean deviation from the horizontal plane. The recovered line length was about 1.08 m, that result in an error of 0.08 m with respect to the measured ground truth. There was no camera rotation along the first trajectory, that corresponded to the measured rotation (error was less than 1 degree on average).

We tracked the second trajectory along two perpendicular lines in the horizontal plane. In this experiment, the physical motion was not particularly smooth and the recorded data somehow jumpy. We measured the angle between the two lines fitted to the recovered positions (see Figure 7.19) as 82°. Hence it had an error of about 8° with respect to the ground truth.

The experiments shown that the accuracy of the measurements connected to properties that are not directly related to calibrated properties of the structure (e.g. deviation from lines, planes, angles) is higher that the accuracy in measured distances. This is probably due to the difficulty in calibrating a projective structure. Overall the results were much better than the ones from the previous model (panorama with range data) from Section 5.3. For example in the case of the panoramic model the deviation from line was about 2 cm.



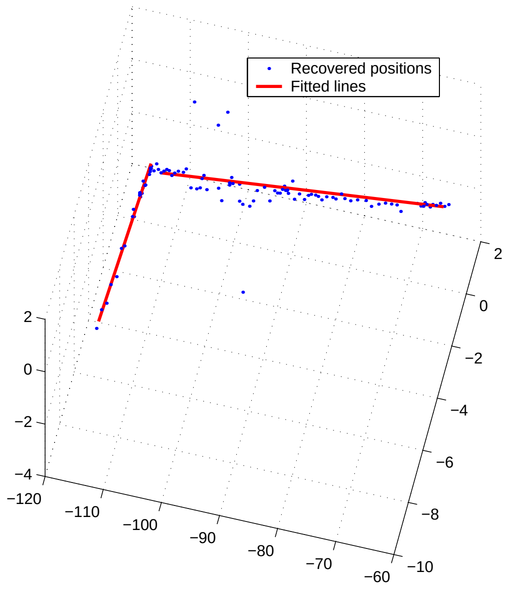


Figure 7.19: Recovered positions for the straight line trajectory (left) and the 2 perpendicular lines trajectory (left). The red line are the fitted 3D lines to each line segment.

For each recovered position we generated the view predicted from the model. Figure 7.20 (top row) shows examples of rendered views along the two trajectories using the dynamic texture model. Comparing them with the real views (bottom row), we notice that the dynamic texture model produces good quality renderings that could replace the actual images. The limited field of view is due to the viewing frustum defined in the original training sequence that was uses for building the model.

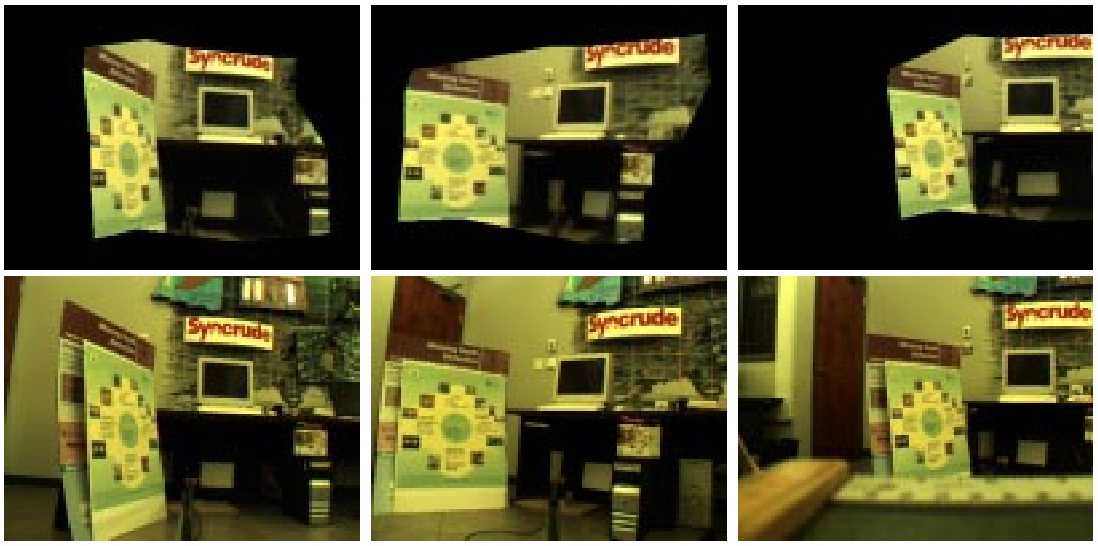


Figure 7.20: Examples of predictive views (top row) and the corresponding actual images (bottom row)

Chapter 8

Conclusions and Future Work

Capturing and modeling scenes is a challenging problem in the fields of computer vision, graphics, and robotics. The traditional geometry based approach reconstructs a detailed 3D model of the space. This has proven to be difficult and in many cases requires a lot of manual work. Image-Based Modeling and Rendering (IBMR), is a relatively new field at the intersection between computer graphics and computer vision. It investigates ways of capturing models directly from images. This new methods offer practical alternatives to traditional geometry-based modeling techniques. This thesis investigates the applicability of image-based models in mobile robotics more precisely in mapping for robot navigation. Mapping involves acquiring a model of the navigation space that is used for e.g. tracking robot's position or controlling the robot. We studied the applicability of image-based modeling in this new field and the level of calibration that the model should have in order to fulfill the accuracy needs for robotics applications. Used as a navigation map, the image-based model does not only need the capability of generating new renderings but also a way to relate the robot current view with the model. To solve this problem we combine a sparse geometric model with image information to generate hybrid geometric and image-based models.

We developed two types of models - a *calibrated* panoramic mosaic registered with depth data and an *uncalibrated* sparse geometric model that uses a special view dependent texture - *dynamic texture*.

8.1 Image-Based Models

The first image-based model is formed by a *calibrated* panoramic mosaic augmented with depth information. The mosaic was built by rotating a camera around the optical center, and stitching together images at every $5 - 10°$. The depth was acquired using either a trinocular stereo vision system or a laser range-finder, and registered with the intensity data. For the stereo vision based system, depth is generated from intensity images so no extra registration is needed. This is not the case with the laser range-finder where the intensity and range data are acquired using separate sensors. This leads to the first important contribution of this thesis:

- A new image-based algorithm for registering range and intensity information from separate sensors.

Traditionally, data registration involves recovering the rigid transformation between the sensors. This is in general a nonlinear problem and requires a good initial estimate to converge. We investigated alternative methods where instead a 2D-2D affine warp is used to align the sensors. Being only an approximation of the physical displacement this approach will only give an approximate alignment, but for applications like robot navigation has proven sufficient. Besides, it is faster and guaranteed to converge, that makes it suitable for real-time applications. We performed a comparative study of registration algorithms [30] and found that the image-based type alignments perform well when there is little variation in the 3D data or when the sensors are close (which is the case of our configuration for generating the panorama with depth - Section 4.4). The exact geometric alignment is more suitable for application that require high precision - such as medical applications. When a camera has significant lens distortion and thus cannot be modeled as a projective camera, the image-based approach can still be applied to locally recover the transformation for portions of the data set.

The aligned depth and intensity data is then segmented into planar patches that can be reprojected in new positions. Due to limitations in extracting depth data from stereo (small baseline - 10 cm - relative to a room size of about 7-10 m), we could not use it directly for segmenting planar patches. We designed:

- A planar patch segmentation algorithm that integrates both depth and intensity data provided by the trinocular vision system.

This algorithm offers a good solution for segmenting indoor environments where planes (e.g. doors, walls, cabinets) have either regular or no texture, but they are visually distinctive in the intensity image. For a more complex environment robust estimators applied to depth data (e.g. RANSAC type algorithms) could offer further improvements.

The second model is composed of a sparse geometry calculated from *uncalibrated* images using structure from motion techniques. We experimented two types of geometries structure - an affine and a projective model. Both were upgraded to a metric model applicable in robotics applications. We found that the affine model works well for capturing small objects but failed to capture a more complex room (3 walls) - Section 7.3. This is due to the weak perspective camera assumption that is a good approximation when the depth variation in the image points is small compared to distance from the camera to the scene. The projective model fits better data in most situations but it is more computationally demanding to estimate and sensitive to outliers. Forcing the models to a metric space introduces additional constraints that decrease the accuracy of reconstruction, but in many cases a metric structure can be valuable. In mobile robotics is almost always required (except in visual servoing approaches) as robot motion is commonly specified in metric coordinates. One of the major drawbacks of our system for capturing the geometric structure was the sparsity of available points. We used real time visual tracking for keeping correspondences and typically relatively few points could be robustly tracked over a whole sequence. We are planning to develop an automatic way of getting more corresponding points that would refine the geometric structure.

The model is then bundle adjusted and reprojected into original images to acquire surface appearance. Surface appearance is represented by:

- A new type of view dependent texture - dynamic texture - that represents surface appearance not using a single traditional texture, but rather by modulating a basis which captures the view dependency of the surface.

We mathematically formulated texture variability for different geometric (nonplanarities, tracking inaccuracies) and photometric variations. This formulation was experimentally proven. Some might argue there is a gap between the theory and the actual way of estimating the texture from statistics of the sample images. The theory shows that such a basis exist and depending on different variabilities present in a sequence we can estimate the number of basis images needed, but there is no formal proof that the actual estimated basis is a combination of the theoretically derived one. We plan, as future work to acquire a calibrated sequence (with known depth and reflectance model) that allows computation of both analytical and estimated basis and see their relation.

To render new poses, the correct texture is modulated from the texture basis and then warped back to the projected geometry. The texture generation and blending is performed in real time on hardware graphics accelerators.

8.2 Applications in Mobile Robotics

We demonstrate the use of our two models as navigation maps in mobile robotics with specific applications in localization, tracking and predictive display. The *calibrated* panoramic model registered with depth data was integrated into:

- Localization algorithms that from a single image compute the camera position.

The first algorithm uses two panoramic mosaics with manually selected corresponding vertical line features for absolute robot localization [26]. A second global localization algorithm, with automatic feature detection and matching was designed for the panorama with depth from stereo [28, 21]. We found that for a global localization algorithm, the feature matching problem becomes very difficult and strong characteristics should be associated with each feature to make it globally distinct. In our case, we used planar patches and a matching score that combines relative distance in 3D and image space with difference in average intensity. For an incremental localization or tracking algorithm the search space is limited by knowing the previous position and assuming a smooth motion. For the panorama with depth from laser we implemented an incremental localization algorithm that uses vertical lines as features [31]. The image information contained in the model was used for robust feature matching.

The *uncalibrated* geometric model was incorporated into:

- A region-based tracking algorithm that relates spatial and temporal image derivatives to update the 3D camera position.

The geometry of the scene is both estimated from uncalibrated video and used for tracking. The method makes tracking of a-priori unknown scenes more stable and handles occlusions by removing and introducing tracked regions when new views become available. Tracking planar regions using a 8DOF homography without a 3D model is unstable due to the many DOF estimated, from a limited image signature. The model constrains individual patches to move in a consistent way and achieve stable tracking of full 3D pose over long sequences.

As a last application, we integrated the image-based model's ability to generate novel views from any position with the developed localization and tracking algorithms in:

- A predictive display system where synthesized immediate visual feedback replaces the delayed video from a remote scene.

A main consideration in designing robotic tele-operation systems is the quality of sensory feedback provided to the human operator. For effective tele-operation the operator must get the feeling of being present in the remote site and get immediate visual feedback from his or her motion commands. While in consumer applications of image rendering, the most important criterion may be the subjective pleasantness of the view, for accurate robot control the geometric precision of the rendered viewpoint is more important than minor errors in scene surfaces or textures. The localization and tracking algorithms provide precise position information that is used for generating a synthesized view from the image-based model. By also adding the current operator motion command to the pose estimate, local predictive display is synthesized immediately in response to operator command.

In general, we found that for calibrated models the reconstruction accuracy depends on the calibration accuracy and precision of feature selection. The uncalibrated models give more flexibility and are easier to acquire. A less constrained model like the projective model will better fit the data than an Euclidean model. The choice of model depends on the level of information and accuracy required by the application.

8.3 Future Work

This thesis investigated the applicability of two type of image-based models (calibrated and uncalibrated) in mobile robotics. Improvements and future directions to the work are connected to both the models and applications.

For the geometry of dynamic texture model, we are planning to further improve the tracking by automatically detecting interest features in the image and initialize them as new trackers. This will eliminate the need for the user to manually intervene the tracking process to select new trackers. As mentioned before, we also plan to have an automatic way to refine the estimated structure by adding new feature points (e.g. using a robust correlation based approach) as well as a way to handle occlusions (maybe by using the model estimated till that point). For the dynamic texture, we plan to formulate a dynamic texture to capture the spatio-temporal statistics of non-static scenes that exhibits quasi periodicity (e.g. wind, water, clouds, fluttering leaves, creasing cloth). A major problem with the image-based models that makes them difficult to incorporate in a scene as graphics objects, because they will carry the lighting of the original capture sequence. Another extension would be to develop an algorithm that separates light from texture. The dynamic texture model can have many applications besides a navigation map. We have already investigate the model applicability in augmented reality [24]. We are planning to elaborate this application and have full scenes composed using models captured using our method.

One of the main drawbacks of all the localization algorithms applied to the calibrated models stem from not introducing feature estimation uncertainty. The robustness of these algorithms could be improved by considering a probabilistic approach when estimating the position, or estimate both position and refine the model at every step (SLAM type approach). The image information contained in the model is a powerful tool for solving data association problem that might be considered still unsolved for the current SLAM approaches.

Our models directly relate geometric robot pose and image views, and this also can support control interfaces where the motion goal is specified in image space instead of robot motor space. The predictive display system could incorporate an image-based motion control. One such possible intuitive interaction paradigm is tele-operating the robot by “pointing” in the image space or by “dragging” the model viewpoint to obtain the desired next view, and then have the robot move to this location using visual servo control.

Overall, this work indicates the benefit of combining ideas from different fields. The field we draw upon, computer vision, robotics and graphics have evolved separately for decades, but recently come to a confluence in research such as image-based rendering and visual servoing. the future is bound to offer more exciting results from such cross-disciplinary fertilization.

Bibliography

[1] M. Armstrong and A. Zisserman. Robust object tracking. In *Second Asian Conference on Computer Vision*, pages 58–62, 1995.

[2] N. Ayache and O. D. Faugeras. Maintaining representation of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6):804–819, 1989.

[3] S. Baker and I. Matthews. *Lucas-Kanade 20 Years On: A Unifying Framework.* Technical Report CMU-RITR02-16, 2002.

[4] J. Baldwin, A. Basu, and H. Zhang. Panoramic video with predictive windows for telepresence applications. In *Int. Conf. on Robotics and Automation*, 1999.

[5] R. Barsi, E. Rivlin, and I. Shimshoni. Visual-homing: surfing the epipoles. In *Proc. of 1998 6th Int. Conf. on Compute Vision*, pages 863–869, 1998.

[6] R. Barsi, E. Rivlin, and I. Shimshoni. Image-based robot navigation under the perspective model. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, pages 2578–2583, 1999.

[7] M. Barth, T. Burkert, C. Eberst, N.O. Stöffler, and G. Färber. Photo-realistic scene prediction of partially unknown environments for the compensation of time delays in presence applications. In *Int. Conf. on Robotics and Automation*, 2000.

[8] A. Bartoli and P. Sturm. Constrained structure and motion from n views of a piecewise planer scene. In *International Symposium on Virtual and Augmented Architecture*, pages 195–206, 2001.

[9] T. Beier and S. Neely. Feature-based image methamorphosis. In *Computer Graphics (SIGGRAPH'92)*, pages 35–42, 1992.

[10] A. K. Bejczy, W. S. Kim, and S. C. Venema. The phantom robot: predictive displays for teleoperation with time delay. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 546–551, 1990.

[11] J. Borenstein, H. R. Everett, and L. Feng. *Where am I?"- Systems and methods for mobile robot positioning.* Technical Report, University of Michigan, 1996.

[12] J. Borestein and Y. Koren. Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Journal on Robotics and Automation*, 7(4):535–539, 1991.

[13] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *IEEE Conference Computer Vision and Pattern Recognition (CVPR00)*, 2000.

[14] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Computer Graphics (SIGGRAPH 2001)*, 2001.

[15] J. Canny and B. Donald. Simplified voronoi diagrams. *Autonomous Robot Vehicles*, pages 272–290, 1990.

[16] J. A. Castellanos, J.M.M. Montiel, J. Neira, and J.D. Tardos. The spmap: a probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–952, 1999.

[17] C.-F. Chang, G. Bishop, and A. Lastra. Ldi tree: a hierarchical representation for image-based rendering. In *Computer Graphics (SIGGRAPH'99)*, 1999.

[18] S. Chen. Quicktime VR - an image-based approach to virtual environment navigation. In *Computer Graphics (SIGGRAPH'95)*, pages 29–38, 1995.

[19] S. Chen and L. Williams. View interpolation for image synthesis. In *Computer Graphics (SIGGRAPH'93)*, pages 279–288, 1993.

[20] D. Cobzas and M. Jagersand. A comparison of non-euclidean image-based rendering. In *Proceedings of Graphics Interface*, 2001.

[21] D. Cobzas and M. Jagersand. Cylindrical panoramic image-based model for robot localization. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1924–1930, 2001.

[22] D. Cobzas and M. Jagersand. Tracking and rendering using dynamic textures on geometric structure from motion. In *European Conference on Computer Vision (ECCV 2002)*, 2002.

[23] D. Cobzas and M. Jagersand. A comparison of viewing geometries for augmented reality. In *Proc. of Scandinavian Conference on Image Analysis (SCIA 2003)*, 2003.

[24] D. Cobzas, M. Jagersand, and K. Yerex. Editing real world scenes: Augmented reality with image-based rendering (poster). In *IEEE Virtual Reality*, 2003.

[25] D. Cobzas, K. Yerex, and M. Jagersand. Dynamic textures for image-based rendering of fine-scale 3d structure and animation of non-rigid motion. In *Eurographics*, 2002.

[26] D. Cobzas and H. Zhang. 2d robot localization with image-based panoramic models using vertical line features. In *Proceedings of Vision Interface*, 2000.

[27] D. Cobzas and H. Zhang. Using image-based panoramic models for 2d robot localization. In *Proceedings of Western Computing Graphics Symposium*, pages 1–7, 2000.

[28] D. Cobzas and H. Zhang. Mobile robot localization using planar patches and a stereo panoramic model. In *Proceedings of Vision Interface*, pages 7–9, 2001.

[29] D. Cobzas and H. Zhang. Planar patch extraction with noisy depth data. In *Proceedings of Third International Conference on 3-D Digital Imaging and Modeling*, pages 240–245, 2001.

[30] D. Cobzas, H. Zhang, and M. Jagersand. A comparative analysis of geometric and image-based volumetric and intensity data registration algorithms. In *Proc. of ICRA*, pages 2506–2511, 2002.

[31] D. Cobzas, H. Zhang, and M. Jagersand. Image-based localization with depth-enhanced image map. In *Proc. of ICRA*, 2003.

[32] S. Coorg and S. Teller. Extracting textured vertical facades from controlled close-range imagery. In *Proc. of the IEEE Int. Conf. on Pattern Recognition (CVPR'99)*, pages 625–632, 1999.

[33] I. J. Cox. Blanche: position estimation for an autonomous robot vehicle. In *Proc. of IEEE/RSJ International Workshop on Robots and Systems (IROS'98)*, pages 432–439, 1989.

[34] I. J. Cox and Editors G. T. Wilfong. *Autonomous Robot Vehicles*. Springer-Verlag, 1990.

[35] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from phtographs. In *Computer Graphics (SIGGRAPH'96)*, 1996.

[36] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[37] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

[38] M. Dissanayake, P. Newman, S. Clark, and H. Durrant-Whyte. A solution to the simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[39] G. Doretto and S. Soatto. Editable dynamic textures. In *ACM SIGGRAPH Sketches and Applications*, 2002.

[40] F. Dornaika and R. Chung. Image mosaicing under arbitrary camera motion. In *Asian Conference on Computer Vision*, Taipei, Taiwan, 2000.

[41] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *PAMI*, 24(7):932–946, July 2002.

[42] S. F. El-Hakim, P. Boulanger, F. Blais, and J. A. Beraldin. Sensor-based creation of indoor virtual models. In *Proc. Virtual Systems and Multimedia - VSMM'97*, 1997.

[43] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, 1987.

[44] A. Hoover *et al.* An experimental comparison of range image segmentation algorithm. *IEEE Trans. PAMI*, 18(7):637–689, 1996.

[45] O. Faugeras. Camera self-calibration: theory and experiments. In *ECCV*, pages 321–334, 1992.

[46] O. Faugeras, B. Hotz, H. Mathieu, P. Fua Z. Zhang, E. Theron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy. *Real time correlation-based stereo algorithm, implementations and applications.* INRIA Technical Report No. 2013, 1993.

[47] O. D. Faugeras. *Three Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, Boston, 1993.

[48] O. D. Faugeras. Stratification of 3D vision: Projective, affine, and metric representations. *Journal of the Optical Society of America, A*, 12(7):465–484, 1995.

[49] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, 1997.

[50] J. Forsberg. *Mobile Robot Navigation Using Non-Contact Sensors.* 1998.

[51] G. Giralt, R. Sobek, and R. Chatila. A multi level planning and navigation system for a mobile robot; a first approach to hilare. In *International Joint Conference on Artificial Intelligence*, volume 1, pages 335–337, 1979.

[52] J. Givant and E. Nebot. Optimization of the simultaneous localization and map building for real time implementation. *IEEE Transactions on Robotics and Automation*, 2001.

[53] M. Gleicher. Projective registration with difference decomposition. In *CVPR97*, pages 331–337, 1997.

[54] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 1993.

[55] S. J. Gortler, R. Grzeszczuk, and R. Szeliski. The lumigraph. In *Computer Graphics (SIGGRAPH'96)*, pages 43–54, 1996.

[56] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *PAMI*, 20(10):1025–1039, October 1998.

[57] G.D. Hager and K. Toyama. X vision: A portable substrate for real-time vision applications. *CVIU*, 69(1):23–37, January 1998.

[58] R. Hartley. Multilinear relationships between coordinates of corresponding image points and lines. In *Sophus Lie Symposium,Norway*, 1995.

[59] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[60] R. Held, A. Efstathiou, and M. Greene. Adaptation to displaced and delayed visual feedback from the hand. *J. Exp Psych*, 72:871–891, 1966.

[61] A. Heyden. Projective structure and motion from image sequances using subspace methods. In *SCIA*, pages 963–968, 1997.

[62] A. Heyden. Algebraic varieties in multiview geometry. In *ICCV*, pages 3–19, 1998.

[63] A. Heyden and K. Åström. Eucledian reconstruction from image sequances with varying an unknown focal length and principal point. In *CVRP*, 1997.

[64] J. Hong, X. Tan, B. Pinette, R. Weiss, and E. Rseman. Image-based homing. *IEEE Control Systems*, pages 38–44, 1992.

[65] B.K.P. Horn. *Computer Vision.* MIT Press, Cambridge, Mass., 1986.

[66] S. L. Horowitz and T. Pavlidis. Picture segmentation by a direct split and merge procedure. In *Proc. of 2nd International Conference on Pattern Recognition (ICPR'74)*, pages 424–433, 1974.

[67] D. Huttenlocher, D. Klanderman, and A. Rucklige. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[68] M. Irani, P. Anandan, and S. Hsu. Mosaic based representation of video sequences and their applications. In *Proc. of the Fifth International Conference on Computer Vsion*, pages 605–611, 1995.

[69] H. Ishiguro, T. Miyashita, and S. Tsuji. T-net for navigating a vision-guided robot in a real world. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1068–1073, 1995.

[70] H. Ishiguro and S. Tsuji. Image-based memory of the environment. In *Proc. of IEEE International Workshop on Robots and Systems (IROS'96)*, pages 634–639, 1996.

[71] D. Cobzas M. Jagersand and H. Zhang. A panoramic model for remore robot environment mapping and predictive display -under review -. *Acta Press: International Journal of Robotics and Automation.*

[72] D. Cobzas M. Jagersand and H. Zhang. A panoramic model for robot predictive display. In *Proc. of Vision Interface*, 2003.

[73] M. Jagersand. Image based view synthesis of articulated agents. In *Computer Vision and Pattern Recognition*, 1997.

[74] X. Y. Jiang and H. Bunke. Fast segmentation of range images into planar regions by scan line grouping. *Machine Vision and Applications*, 7(2):115–122, 1994.

[75] M. Jogan and A. Leonardis. Panoramic eigenimages for spatial localization. In *Proc. Computer Analysis of Images and Patterns (CAIP'99)*, 1999.

[76] J.-M. Jolion, P. Merr, and S. Bataouche. Robust clustering with application in computer vision. *IEEE Trans. PAMI*, 13(8):791–802, 1991.

[77] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *PAMI*, 24(7):996–1000, July 2002.

[78] S. B. Kang. *A Survey of Image-Based Rendering Techniques.* Technical Report CRL 97/4, Cambridge Research Laboratory, 1997.

[79] S. B. Kang, A. Johnson, and R. Szeliski. *Extraction of Concise and Realistic 3-D Models from Real Data.* Technical Report CRL 95/7, Cambridge Research Laboratory, 1995.

[80] S.B. Kang and R. Szeliski. 3D scene data recovery using omnidirectional multibaseline stereo. In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR'96)*, pages 364–370, 1996.

[81] R. Koch, M. Pollefeys, B. Heigl, L. Van Gool, and H. Niemann. Calibration of hand-held camera sequences for plenoptic modeling. In *International Conference on Computer Vision (ICCV 99)*, pages 585–591, 1999.

[82] J. Kostkova, J.Cech, and R.Sara. Dense stereomatching algorithm performance for view prediction and structure reconstruction. In *Proc. of the 13th Scandinavian Conference on Image Analysis (SCIA2003)*, pages 101–107, 2003.

[83] B. J. Kuipers and Y.T. Byun. A robotic exploration and mapping startegy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–630, 1991.

[84] C. Kunz, T. Willeke, and I. R. Nourbakhsh. Automatic mapping of dynamic office environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1681–1687, 1997.

[85] R. Kurazume, M. D. Wheeler, and K. Ikeuchi. Mapping textures on 3d geometric model using reflectance image. In *Proc. of the Data Fusion Workshop in IEEE, ICRA)*, 2001.

[86] K. Kutulakos and S. Seitz. A theory of shape by shape carving. *International Journal of Computer Vision*, 38:197–216, 2000.

[87] S. Laveau and O.D. Faugeras. 3-D representation as a collection of images. In *Proc. of the IEEE Int. Conf. on Pattern Recognition (CVPR'97)*, pages 689–691, Jerusalem,Israel, 1994.

[88] K.-C. Lee, J. Ho, and D. Kriegman. Nine points of light: Acquiring subspaces for face recognition under variable lighting. In *Computer Vision and Pattern Recognition*, 2001.

[89] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, 1991.

[90] M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics (SIGGRAPH'96)*, pages 31–42, 1996.

[91] M. Lhuiller and L. Quan. Image interpolation by joint view triangulation. In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR'99)*, pages 139–145, 1999.

[92] S. Li and S. Tsuji. Qualitative representation of scenes along route. *Image and Vision Computing*, 17:685–700, 1999.

[93] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In *EUROGRAPHICS*, 1999.

[94] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using EM to learn 3D models with mobile robots. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[95] D.G. Lowe. Fitting parameterized three-dimensional models to images. *PAMI*, 13(5):441–450, May 1991.

[96] F. Lu and E. Milios. Globally consistant range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[97] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. on Artificial Intelligence*, 1981.

[98] R. A. Manning and C. R. Dyer. Interpolating view and scene motion by dynamic view morphing. In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR'99)*, pages 388–394, 1999.

[99] E. Marchand, P. Bouthemy, and F. Chaumette. A 2d-3d model-based approach to real-time visual tracking. *IVC*, 19(13):941–955, November 2001.

[100] M. J. Mataric. *A Distributed Model for Mobile Robot Environment Learning and Navigation.* Technical Report AITR-1228, MIT, 1990.

[101] L. Mathies and A. Elfes. Integration of sensor and stereo range data using a grid based representation. In *Proc of IEEE Int. Conf. on Robotics and Automation*, pages 727–733, 1988.

[102] Y. Matsumoto, M. Inaba, and H. Inoue. Visual navigation using view-sequenced route representation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 83–88, 1996.

[103] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image based visual hulls. In *Computer Graphics (SIGGRAPH'2000)*, pages 367–374, 2000.

[104] D. K. McAllister, L. Nyland, V. Popescu, A. Lastra, and C. McCue. Real-time rendering of real world environments. In *Proc. of Eurographics Workshop on Rendering*, Spain, June 1999.

[105] L. McMillan. *An Image-Based Approach to Three-Dimensional Computer Graphics.* Ph.D. Dissertation. UNC CS TR97-013, University of North Carolina, 1997.

[106] L. McMillan and G. Bishop. Plenoptic modeling: Am image-based rendering system. In *Computer Graphics (SIGGRAPH'95)*, pages 39–46, 1995.

[107] M. Meng and A. C. Kak. Neuro-nav: A neurol network-based architecture for vision guided mobile robot navigation using non-metrical models of the environment. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 750–757, 1993.

[108] T. Möller and E. Haines. *Real-time Rendering.* A.K. Peterson, 2002.

[109] M. Montemerlo, W. Whittaker, and S. Thurn. Fastslam: A factored solution to the simultaneous localization and map building. submitted for publication. 2002.

[110] H. P. Moravec. *Obstacle Avoidance and Navigation in the Real World by Seeing a Robot Rover.* PhD thesis, Stanford University, 1980.

[111] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. of 1985 IEEE Int. Conf. on Robotics and Automation*, pages 116–121, 1985.

[112] D. Murray and C. Jennings. Stereo vision based mapping and navigation for mobile robots. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, pages 1694–1699, 1997.

[113] E. Natonek. Fast range image segmentation for servicing robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 406–411, 1998.

[114] S. Nayar. Catadioptric okmnidirectional camera. In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR'97)*, pages 482–488, 1997.

[115] M. Oliveira. *Image-Based Modeling and Rendering Techniques: A Survey.* Technical Report , Instituto de Informatica, UFRGS, Porto Alegre, Brasil, 2002.

[116] M. Oliveira and G. Bishop. Image-based objects. In *ACM Symposium on Interactive 3D Graphics*, pages 191–198, 1999.

[117] M. Oliviera, Gary Bishop, and David McAllister. Relief texture mapping. In *Computer Graphics (SIGGRAPH'00)*, 2000.

[118] C. F. Olson and L. H. Matthies. Maximum likelihood rover localization by matching range maps. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 272–277, 1998.

[119] B. Parvin and G. Medioni. Segmentation of range images into planar surfaces by split and merge. In *Proc. of International Conference on Computer Vision and Pattern Recognition (CVPR'86)*, pages 415–417, 1986.

[120] S. Peleg and M. Ben-Ezra. Stereo panorama with a single camera. In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR'99)*, 1999.

[121] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR'97)*, pages 338–343, 1997.

[122] J. S. Perrier, G. Agam, and P. Cohen. Physically valid triangulation of scarcely matched images using texture information: application to view-synthesis. In *Proc. of Vision Interface (VI'2000)*, pages 233–240, 2000.

[123] C. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In *ECCV*, pages 97–108, 1994.

[124] C. J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):206–218, 1997.

[125] M. Pollefeys, L Van Gool, and M. Proesmans. Eucledian 3d recinstruction from image sequances with variable focal length. In *ECCV*, pages 31–42, 1996.

[126] M. Pollefeys and L.Van Gool. Self-calibration from the absolute conic on the plane at infinity. *LNCS 1296*, pages 175–182, 1997.

[127] M. Pollyfeys. *Tutorial on 3D Modeling from Images.* Lecture Nores, Dublin, Ireland (in conjunction with ECCV 2000), 2000.

[128] W. H. Press, B.P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipies in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, England, 1992.

[129] P.W. Rander, P.J. Narayanan, and T. Kanade. Virtualized reality: Constructing time varying virtual worlds from real world events. In *IEEE Visualization*, 1997.

[130] C. Rasmussen. *Visual Servoing and Mobile Robot Navigation.* Technical Report, John Hopkins University, 1995.

[131] Point Grey Research. *http://www.ptgrey.com.*

[132] S. Sarkar. Lola edge detection and linking code. *http://marathon.csee.usf.edu/ sarkar/vision_html/lola_code/.*

[133] F. Schmitt and X. Chen. Fast segmentation of range images into planar regions. In *Proc. of International Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pages 710–711, 1991.

[134] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2051–2058, 2001.

[135] S. M. Seitz and C. R. Dyer. View morphing. In *Computer Graphics (SIGGRAPH'96)*, pages 21–30, 1996.

[136] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR'97)*, pages 1067–1073, 1997.

[137] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *Computer Graphics (SIGGRAPH'98)*, 1998.

[138] T. B. Sheridan. Space teleoperation through time delay: Review and prognisis. *IEEE Tr. Robotics and Automation*, 9, 1993.

[139] J. R. Shewchuk. Triangle-a two-dimensional quality mesh generator and delaunay triangulator. *http://www.cs.cmu.edu/ quake/triangle.html.*

[140] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *Computer Graphics (SIGGRAPH'99)*, pages 299–306, 1999.

[141] H.-Y. Shum and R. Szeliski. *Panoramic image mosaics.* Technical Report MSR-TR-97-23, Microsoft Research, 1997.

[142] R. Sim and G. Dudek. Learning and evaluating visual features for pose estimation. In *ICCV*, pages 1217–1222, 1999.

[143] R. C. Smith and P. Cheeseman. *On the representation and estimation of spatial uncertainty.* Technical Report, TR 4760 & 7239, SRI, 1985.

[144] I. Stamos and P. K. Allen. Integration of range and image sensing for photorealistic 3d modeling. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 1435–1440, 2000.

[145] P. Sturm and B.S. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *ECCV (2)*, pages 709–720, 1996.

[146] P.F. Sturm. Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length. *IVC*, 20(5-6):415–426, March 2002.

[147] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, pages 22–30, March 1996.

[148] R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and environment maps. In *Computer Graphics (SIGGRAPH'97)*, pages 251–258, 1997.

[149] C. Taylor and D. J. Kriegman. Structure and motion from line segments in multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 1995.

[150] S. Thrun. Learning maps for mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[151] S. Thrun. *Robotic Mapping: A Survey.* Technical Report CMU-CS-02-111, Carnegie Mellon University, 2002.

[152] S. Thrun, A. Buecken, W. Burgard, D. Fox, T. Froehlinghaus, D. Henning, T. Hofmann, M. Krell, and T. Schmidt. *AI-based Mobile Robots: Case Studies of Successful Robot Systems*, chapter Map Learning and High-Speed Navigation in RHINO. MIT Press, 1998.

[153] S. Thrun, D. Fox, and W. Burgad. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998.

[154] S. Thurn, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2000.

[155] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9:137–154, 1992.

[156] P. Torr. *Motion segmentation and outliers detection.* PhD thesis, University of Oxford, 1995.

[157] P. Torr and A. Zisserman. Robust parametrization and computation of the trifocal tensor. *Image and Visual Computing*, 15:591–605, 1997.

[158] K. Toyama and G.D. Hager. Incremental focus of attention for robust vision-based tracking. *IJCV*, 35(1):45–63, November 1999.

[159] W. Triggs. The geometry of projective reconstruction i: Matching constraints and the joint image. In *ICCV*, pages 338–343, 1995.

[160] W. Triggs. Auto-calibration and the absolute quadric. In *CVRP*, pages 609–614, 1997.

[161] R. Y. Tsai. A versitile camera clibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Transactions and Robotics and Automation*, 3(4):323–344, 1987.

[162] M.Urban T.Werner, T.Pajdla. Practice of 3d reconstruction from multiple uncalibrated unorganized images. In *Czech Pattern Recognition Workshop*, 2000.

[163] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological maps. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1023–1029, 2000.

[164] S.-C. Wei, Y. Yagi, and M. Yachida. Building local floor map by use of ultrasonic and omni-directional vision sensor. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, pages 2548–2553, 1998.

[165] D. Weinshall and C. Tomasi. Linear and incremental aquisition of invariant shape models from image sequences. In *Proc. of 4th Int. Conf. on Compute Vision*, pages 675–682, 1993.

[166] M. Wilczkowiak, E. Boyer, and P. Sturm. 3d modelling using geometric constraints: A parallelepiped based approach. In *European Conference on Computer Vision (ECCV 2002)*, pages 221–236, 2002.

[167] N. Winters and J. Santo-Victor. Mobile robot navigation using omnidirectional vision. In *Proc. 3rd Irish Machine Vision and Image Processing Conference (IMVIP'99)*, 1999.

[168] D. Wood, D. Azuma, W. Aldinger, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle. Surface light fields for 3d photography. In *Computer Graphics (SIGGRAPH 2000)*, 2000.

[169] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, and D. H. Salesin. Multiperspective panoramas for cell animation. In *Computer Graphics (SIGGRAPH'97)*, 1997.

[170] Y. Yagi, S. Fujimura, and M. Yachida. Route representation for mobile robot navigation by omnidirectional route panorama fourier transform. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1250–1255, 1998.

[171] Y. Yagi and M. Yachida. Environmental map generation and egomotion estimation in a dynamic environment for an omnidirectional image sensor. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, pages 3493–3498, 2000.

[172] B. Yamauchi and P. Langley. Place recognition in dynamic environments. *Journal of Robotic Systems, Special Issue on Mobile Robots*, 14(2):107–120, 1997.

[173] B. Yamauchi, A. Schultz, and W. Adams. Mobile robot exploration and map-building with continuous localization. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, 1998.

[174] Z. Zhang and O. Faugeras. A 3d world model builder with a mobile robot. *International Journal of Robotics Research*, 11(4):269–285, 1992.

[175] J. Y. Zheng and S. Tsuji. Panoramic representation for route recognition by a mobile robot. *International Journal of Computer Vision*, 9(1):55–76, 1992.