

Augmenting Reality, Naturally

by

Iryna Gordon

B.Sc., University of Manitoba, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

November 2004

© Iryna Gordon, 2004

Abstract

Augmented Reality (AR) is a promising new technology, aimed at enhancing the user's visual perception of the physical world with computer-generated virtual imagery. Virtual objects, such as rendered 3D models, 2D textures, highlights and labels, must appear correctly projected onto live video captured by a mobile camera. To achieve such a synthesis in a realistic manner, it is necessary to recognize what is viewed by the camera, and to accurately localize the camera and the virtual objects in the real world.

In this thesis I address the challenge of automated and robust video augmentation in a variety of natural and unprepared settings. I have implemented a system which computes the camera pose by matching natural image features from a current video frame to a previously constructed 3D model of an operating environment. The system provides built-in tools for the construction of the scene model from reference images, the autocalibration of the camera and the interactive insertion of a virtual object into the modelled scene. Invariant natural features replace special markers for achieving successful recognition in images, and enable stable camera tracking in occluded and dynamic settings. Model recognition from arbitrary viewpoints removes the need to manually initialize the tracker. Experimental results demonstrate geometrically consistent augmentation for a wide variety of environments and unconstrained camera motion.

Contents

Abstract	ii
Contents	iii
List of Tables	v
List of Figures	vi
Acknowledgements	viii
Dedication	ix
1 Introduction	1
1.1 Emerging AR Applications	2
1.2 Background	3
1.2.1 Basics	3
1.2.2 Challenges	4
1.2.3 New Directions	6
1.3 Thesis Overview	7
2 Related Research	10
2.1 Towards Markerless AR	10
2.1.1 Invariant Features for Recognition	11
2.2 Describing the World	13

2.2.1	3D Modelling from Images	15
3	Modelling Reality	17
3.1	Image Features	18
3.1.1	Feature Extraction	19
3.1.2	Feature Matching	19
3.1.3	Verifying Geometric Consistency	21
3.2	Scene Structure and Camera Parameters	24
3.2.1	Incremental Model Construction	26
3.3	Inserting Virtual Content	29
4	Camera Tracking	32
4.1	From Features to Camera	33
4.2	Reducing Jitter	34
4.3	Speeding Up Model Recognition	35
5	Experiments and Results	37
5.1	On Modelling	39
5.1.1	Challenges in Shape Reconstruction	43
5.2	On Tracking	46
5.2.1	Computation Times	47
5.2.2	Registration Accuracy	47
6	Concluding Remarks	51
6.1	Future Work	52
	Bibliography	54

List of Tables

5.1	<i>Experimental data</i>	38
5.2	<i>Performance of the modelling algorithm</i>	42
5.3	<i>Computation times</i>	47

List of Figures

1.1	<i>Aligning real and virtual imagery</i>	5
1.2	<i>Augmentation examples</i>	9
2.1	<i>SIFT features in an image</i>	13
3.1	<i>Spanning tree example</i>	22
3.2	<i>Model and cameras</i>	27
3.3	<i>Triangulating the virtual frame origin</i>	30
3.4	<i>Inserting the virtual object</i>	31
4.1	<i>Speeding up recognition</i>	36
5.1	<i>Tabletop model</i>	40
5.2	<i>Boxes model</i>	40
5.3	<i>Mug model</i>	40
5.4	<i>Sneaker model</i>	41
5.5	<i>Book model</i>	41
5.6	<i>Library model</i>	41
5.7	<i>Shape ambiguity</i>	44
5.8	<i>Inaccurate camera pose</i>	45
5.9	<i>Tracking the coffee mug</i>	46
5.10	<i>Tracking the Velveteen Rabbit</i>	46
5.11	<i>Registration accuracy experiment</i>	48

5.12 <i>Aligning squares: stationary camera results</i>	50
5.13 <i>Aligning squares: moving camera results</i>	50

Acknowledgements

My deepest gratitude goes to my supervisor, Dr. David Lowe, for his invaluable guidance, insight and support in this research, and throughout my Masters studies. Without his contribution this work would not have been completed. My sincere thanks to Dr. Jim Little for agreeing to be the second reader of this thesis.

I would like to thank the Robuddies group and the fellow students at the Laboratory for Computational Intelligence, for their helpful comments and stimulating discussions, and for making these last few years of study most enjoyable.

Special thanks to my husband David for his endless support, understanding and encouragement in every step of this endeavour.

This project was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Institute for Robotics and Intelligent Systems (IRIS).

IRYNA GORDON

*The University of British Columbia
November 2004*

*to my husband,
for putting up with everything this work entailed*

Chapter 1

Introduction

Unlike Virtual Reality, which completely immerses the user into a world generated by a computer, Augmented Reality integrates virtual elements into the user's real surroundings. Additional visual information is introduced while preserving natural interaction with the physical world. This characteristic makes AR an exciting technology with a vast potential in human-computer interfaces. A comprehensive introduction to AR as a field, as well as a discussion of important challenges, recent research trends and emerging applications, can be found in [3, 4].

Pioneer works related to AR date back to the introduction of head-mounted 3D displays (HMDs) in the 1960s [46]. Technological advances of the recent years have fuelled the growth of AR into a promising interdisciplinary field. It has attracted interest of researchers in computer vision, graphics, user interfaces and engineering. New applications have emerged in the areas of medicine, industry, tourism, entertainment and education, to name a few. Imagine an architect, observing through a pair of special glasses a rendered model of his design, displayed over an empty construction lot; a tourist, while exploring an archeological site, seeing a virtual replica of an ancient temple where it once stood; a child, turning a page of a book and seeing characters come to life in an animated virtual scene; a civil engineer with an X-ray vision of a building's electrical wiring; a technician, repair-

ing equipment with instructions projected directly onto the relevant locations... the possibilities are endless.

1.1 Emerging AR Applications

Archeoguide [51] is an AR-based computer guide for cultural heritage sites. Historical site visitors are equipped with a see-through HMD, an earphone and a mobile tracking system. Supplemental audio and visual information is meant to provide a deeper insight into the relevant aspects of a historical location.

The Medarpa project [43] involves developing an AR-supported medical workstation for various surgical procedures. With an aid of a semi-transparent display located above a patient, a treating physician can view computer-generated medical images, projected directly onto the patient's body.

A successful application of AR to industry is demonstrated in [1]. 2D technical drawings, such as floor maps, pipe layouts and wiring plans, are used to register 3D models of new installations in factory images.

Magic Book [8] is an innovative application, which combines Augmented and Virtual Realities in a collaborative environment. By looking at a book page through a special HMD, a reader can view animated virtual scenes, projected onto the page. With a flip of a switch, the reader is transported into a fully immersive virtual environment, becoming part of the story. Several readers can gather around the book and share their virtual experiences by exchanging their perspectives of the story scene.

Augmented Chemistry [16] is an interactive educational workbench, which enables a student to construct 3D virtual molecules. Using a special gripper tool, the student can pick up virtual "atoms" from a booklet that contains elements of the periodic table, and bind them to the molecular model.

In [18] AR technology is applied to enrich a visitor's experience of a museum exhibit. A small video camera is attached with a flexible wire to a corner of an

exhibit display. By pointing the camera at different parts of the display, visitors can view relevant textual and graphic information on a nearby PC screen.

1.2 Background

The nature of a particular application dictates the amount of realism to be achieved by the augmentation. Virtual content may range from simple highlights and textual annotations to complex 3D graphics with lighting effects, animation and proper resolution of occlusions. Regardless of the sophistication level of its graphics, the first and foremost demand from any AR application is a timely and accurate geometric *registration* of a virtual projection in a real-world image. A virtual object must appear seamlessly integrated into the user’s environment as the camera moves around. A jittery, floating or flickering virtual content is likely to cause immediate irritation, at the very least.

1.2.1 Basics

In a *video see-through* AR, precise registration is a consequence of an exact alignment of two video streams: the live video of the real world and the rendering of a virtual object by a graphics engine¹. We can think of it as an alignment of two cameras: the real one, through which the user sees the world, and the virtual one, which projects the graphics component (Figure 1.1). The following geometric transformations must be established:

- object-to-scene: 3D coordinates of a virtual object, expressed in a coordinate frame of a real-world scene;
- scene-to-camera: the pose (orientation and translation) of the camera in the real world; and

¹*Optical see-through* technology projects virtual content directly onto a semi-transparent surface in front of the user’s eyes. This thesis assumes video-based synthesis, although the same principles apply to both video and optical AR systems (an additional requirement for the latter is the calibration of the optical combiner with respect to the user’s eyes).

- camera-to-image: the calibration parameters of the camera, defining its projection of a 3D object onto a 2D image plane.

Before projecting a virtual object onto an image surface, the 3D pose of the object must be expressed in the camera frame:

$$R_{co} = R_{cs}R_{so} \quad (1.1)$$

$$\mathbf{t}_{co} = R_{cs}\mathbf{t}_{so} + \mathbf{t}_{cs} \quad (1.2)$$

where R_{co} and \mathbf{t}_{co} compose the object-to-camera transformation; R_{so} and \mathbf{t}_{so} denote the object-to-scene transformation, which is usually determined in advance by the user; lastly, R_{cs} and \mathbf{t}_{cs} form the scene-to-camera transformation, which must be accurately computed for each video frame.

1.2.2 Challenges

While the problem of camera pose estimation, or camera tracking, is not unique to AR, in the context of AR research it is undoubtedly one of the most important and challenging tasks. This is because tracking must be performed online and in real time. Also, the camera is often handheld or headmounted, and therefore its motion can be quite unpredictable and unstable. Most importantly, the human eye is capable of detecting even the slightest image discrepancies, which places a demand on camera tracking results to be as accurate as possible.

Among existing camera tracking technologies, such as those based on magnetic or infrared sensors, vision-based tracking is quickly gaining popularity, in proportion to the increasing accuracy and speed of computer vision algorithms, as well as the emergence of fast dedicated hardware. Vision-based trackers are cheap, accurate and long-range. While computational cost is becoming less of a concern, the problems of robustness and flexibility remain to be solved.

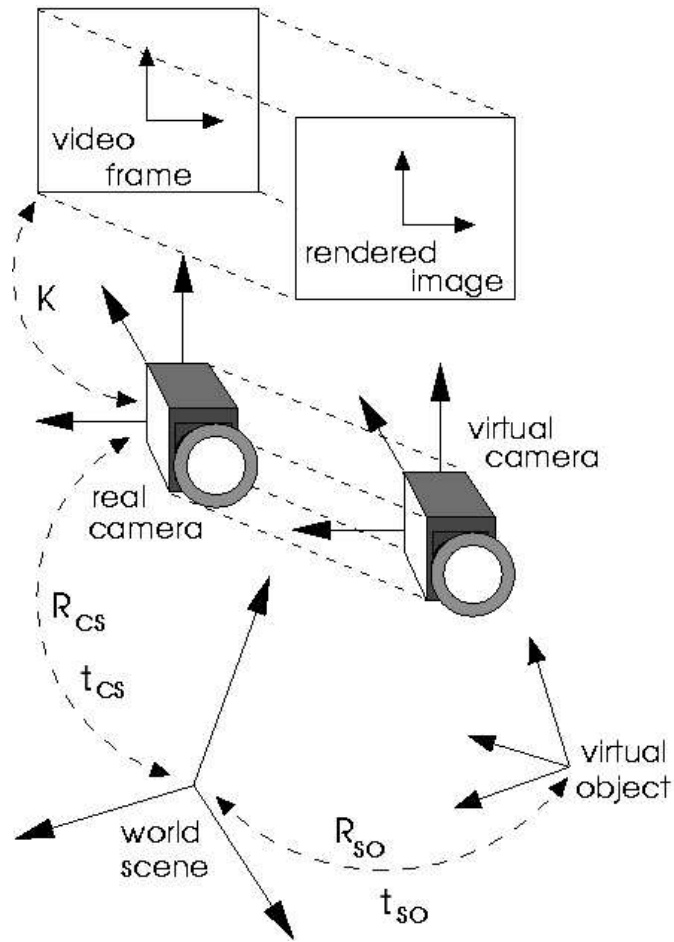


Figure 1.1: Parameters of the real and virtual cameras must match in order to align real and virtual imagery. K is a matrix containing camera calibration parameters; R and \mathbf{t} are a rotation matrix and a translation vector, respectively, which define a 3D Euclidean transformation.

1.2.3 New Directions

Until recently, scene recognition and camera tracking via vision sensing in AR has commonly relied on identifying and tracking fiducial markers [23, 1, 22, 45]. These artificial visual “beacons” take form of distinct and easily recognizable geometric patterns, which must be strategically positioned in the environment prior to the system operation. However, the required modification of the user’s surroundings can be cumbersome, aesthetically questionable, and in some cases entirely infeasible (for example, in outdoor applications). To make matters worse, the fiducials must remain fully visible to be successfully detected. Thus, the use of markers imposes limitations on the range of camera travel and makes an AR system vulnerable to occlusions.

As a result, one of the new research trends in AR is camera tracking using *natural features*, which are already present in a viewed scene. Computer vision algorithms are used to extract local image features and find their reliable correspondences across images. Natural feature tracking enables a wide range of AR applications in unprepared settings, as well as provides a greater degree of robustness to occlusions and changes in the environment. The success of a marker-free approach largely depends on both the quantity and distinctiveness of natural features that can be detected.

Earlier AR technologies required the camera to be carefully precalibrated before use [50]. Manual calibration procedures are typically quite tedious and time-consuming. Besides, they require specialized devices, such as objects or patterns with a well established geometric configuration. Reliable *autocalibration* techniques [21, 38] are a welcome advance in computer vision research, as they can significantly reduce setup requirements for AR systems. Calibration-free approaches to camera tracking in affine and projective frameworks have also been considered [25, 45].

1.3 Thesis Overview

My work is motivated by the rising need for versatile, low-maintenance AR systems, capable of functioning in unprepared settings and generally requiring less effort to initialize and operate. This thesis aims at developing an approach to AR which performs automatic tracking (re)initialization, does not require manual camera pre-calibration procedures or prior knowledge of scene geometry, and can operate in a vast variety of environments, yielding consistent and accurate augmentation results.

As an alternative to special markers, I use stable natural features as visual data which drives the system's algorithms. Generated from an image via the Scale Invariant Feature Transform (SIFT) algorithm [29, 30], these features act as descriptors of local image patches. They are invariant to image scaling and rotation, and partially invariant to changes in illumination and viewpoint. The distinctiveness of SIFT features, as well as their abundant presence over a large range of image scales, make them suitable for recognition in cluttered and dynamic settings. Feature matching is performed efficiently and reliably via an approximate tree search algorithm.

The system operates in two stages. During the first, offline stage, SIFT features are extracted from reference images of an environment to be augmented (this can be an individual object or a general scene). The images are assumed to have been taken by an uncalibrated camera from unknown viewpoints. Multi-view feature correspondences are then found, to be used for building a Euclidean model of the image contents. At the same time, calibration parameters and camera poses corresponding to the reference viewpoints are computed. These *structure-from-motion* computations are performed with a straightforward and powerful technique. It is based on non-linear least squares, and can construct models of arbitrary geometric complexity starting from a very simple initialization.

Once the real-world model has been obtained, the position, orientation and size of a virtual object must be specified in the coordinate frame of the model. To

this end I have developed an interactive application, which allows the user to place a virtual object into a scene by determining its appearance in the reference images.

The second stage of the system involves model recognition and camera tracking for live video augmentation (Figure 1.2 shows a few examples of augmented video frames). Features detected in a current video frame are matched to those of the world model, and the matches are used to compute the pose of the camera relative to the model. Virtual jitter, resulting from computational inaccuracies, is reduced by regularizing the solution, using the camera pose computed for the previous frame. The influence of the previous solution on the current one is weighted in such a way as not to impose constraints on the overall camera motion. The camera tracker can perform online scene recognition and recovery from failure, with the tracked scene going in and out of view. Automatic recognition is an essential feature for many applications, particularly for automated mobile systems.

The remainder of this paper is organized as follows. An overview of related research is provided in Chapter 2. Chapter 3 discusses the approach to scene modelling and virtual object insertion. The details of camera pose tracking are presented in Chapter 4. Chapter 5 discusses experimental results and overall system performance. Final remarks and conclusions are found in Chapter 6.

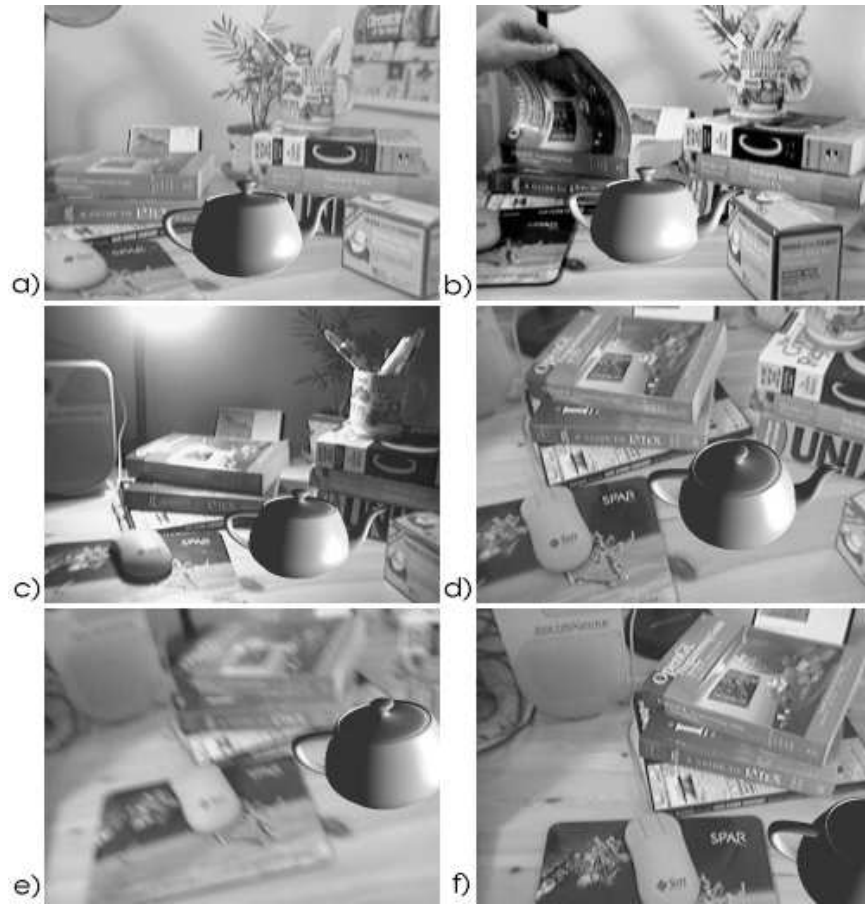


Figure 1.2: *System in action: a) a virtual teapot in the modelled desk scene; b) a non-rigid change in the scene; c) a change in lighting; d) a change in viewpoint; e) motion blur; f) partial view of the scene.*

Chapter 2

Related Research

The multitude of current research directions in AR reflects the interdisciplinary nature of the field. New developments include enabling hardware technologies for wearable AR systems, user acceptance studies, photorealistic rendering, collaborative applications and many others [4]. This chapter narrows its survey to visual environment sensing, which is the focus of the thesis. In particular, under discussion are natural feature-based approaches and methods for acquisition of an environment representation, which underlay camera tracking in AR.

2.1 Towards Markerless AR

In most markerless AR systems, natural features are used for establishing correspondences between consecutive frames in a video sequence, i.e., for narrow baseline matching. Such correspondences, often referred to as optic flow, are used for determining the frame-to-frame motion of the camera. Some of the most common choices of features for narrow baseline matching are the Harris corner detector [20], applied in [10, 11], and the Kanade-Lucas-Tomasi (KLT) feature tracker [31], used in [52, 17, 41]. These features can be efficiently extracted and have been used quite effectively for the tasks of motion tracking and structure-from-motion. However, they do not provide a good basis for wide baseline matching, which is required for

automated tracker initialization and demands higher viewpoint and scale invariance from image features.

A recent approach to markerless AR [13] proposes tracking of parallelogram-shaped and elliptical image regions, extracted in an affinely invariant way. These local features can also be used for scene recognition, although not in real time. Compelling results are presented, however the tracker lacks generality, as it relies on the presence of planar structures in the viewed scene.

In [27] viewpoint invariance is achieved by applying eigen image methodology to describe a collection of local image patches. These patches are extracted from reference images and capture the appearance of a small region in several distinct views. The AR system expects the user to supply a Computer-Aided Design (CAD) model of a real object to be augmented, and requires manual matching of CAD model points to their 2D projections in reference images.

The system described in [24] uses image edges as natural features. Contours of a supplied CAD model are matched to detected image edges. The visual tracking system relies on rate gyroscopes to handle rapid movements of a headmounted camera. Automatic localization and recovery from tracking failure are not supported.

2.1.1 Invariant Features for Recognition

The problem of feature invariance and distinctiveness is closely related to the task of object recognition in computer vision. To this end, numerous types of stable features have been proposed, some of which will be discussed below.

Features invariant to affine transformations [35] have been used for 3D object modelling and recognition [40], as well as for multi-view matching of unordered image sets [42]. An affinely invariant descriptor for a local image patch is obtained by resampling the patch in an affine reference frame. An affine transformation provides a mapping for an elliptical image region into a unit circle, or similarly, for a parallelogram into a unit square. For image patches on planar surfaces, affinely

invariant descriptors provide full viewpoint invariance under orthographic projection. Affine invariants can also be used for recognizing non-planar shapes, which are approximated by collections of small planar patches [40].

In [33] “maximally stable extremal” (MSE) regions are sought by thresholding an image and keeping track of the connected components as the threshold value changes. An MSE region is found when an area of a component remains unchanged over a large range of thresholds. MSE regions correspond to image areas which have high contrast with respect to their surroundings, and are invariant to affine transformations of image intensities. The above paper proposes an efficient extraction algorithm and a robust similarity measure for establishing tentative matches.

A descriptor for matching object shapes is developed in [7]. A shape is represented by a set of points, sampled from an object contour in an image. For a point on the contour, a shape context descriptor is computed from a coarse histogram of the relative coordinates of the rest of the points. The descriptor is invariant under scaling, image rotation and translation, and is robust to small affine transformations.

Another contour-based approach is discussed in [36]. Curves are extracted from an image, which are then segmented at points of high curvature. The longest curves are chosen as keys and are fit into local context patches. Each context patch is a normalized square image region, containing the key curve and all other curve segments that intersect the region. Good recognition results are demonstrated for complex objects in the presence of clutter and occlusions.

This thesis makes use of scale-invariant SIFT descriptors, which are highly distinctive, relatively efficient to extract and stable in the presence of image noise. In addition, the SIFT algorithm produces a dense population of features for typical textured images (Figure 2.1), which makes it suitable for recognition in cluttered settings, and for optimization algorithms which benefit from large input sets. Previous experiments demonstrate a remarkable success of SIFT features in a variety of object recognition tasks [30].

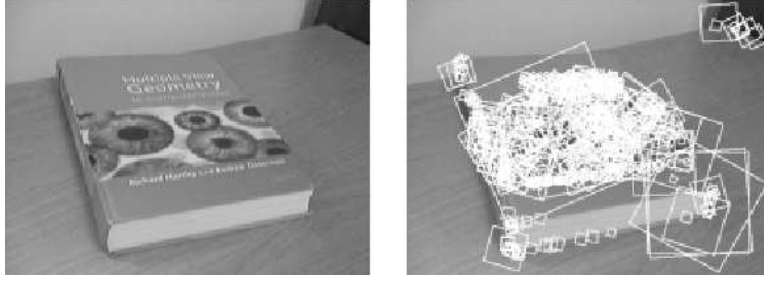


Figure 2.1: A 640×480 image of a famous book (left) and its found SIFT features (right). The algorithm extracted 625 features, shown as overlapping squares with varying scale and orientation.

2.2 Describing the World

3D CAD models have proven to be quite effective as reference tools for object recognition and camera tracking in AR. The drawback is extra effort involved in manually building a CAD model in advance, as it may not be readily available. Moreover, CAD tools are limited to objects that can be easily modelled by hand, i.e. those with well defined geometric shapes. In contrast, in my work I attempt to target arbitrary shapes and textures, often present in natural environments and man-made settings. I also aim to reduce system requirements for additional commercial tools or equipment, limiting them to just an off-the-shelf video camera. Consequently, an alternative reference representation of the real world must be sought.

Various techniques have been suggested in AR for acquiring a reference representation of an operating environment, to aid online computations. In [10] two or more reference views are used to compute the current camera pose from epipolar geometry constraints on natural feature correspondences. Markers must still be used to precalibrate the camera and to determine its pose for the reference images. Another disadvantage of the approach is that the camera pose for the very first video frame must be very close to one of the reference views, due to limited wide baseline matching capabilities.

A learning-based strategy is proposed in [17], where a real environment is

represented by an acquired set of natural features. These features have been detected and localized in 3D during an initial tracking phase. The learning phase relies on recognizing fiducial markers, whose locations in the world have been determined in advance. Markers are first used to compute the pose of the camera, after which the system can compute world coordinates of newly detected natural features. In this manner, an initially limited tracking range is extended during online operation of the system.

The fiducial-based system, presented in [25], uses the factorization method [48] to represent a scene and its virtual content in an affine frame of reference, with an aim to avoid Euclidean camera calibration. This innovative approach achieves comparable results with minimum initialization effort and simple algorithms. However, it does not allow the modelling of perspective projection effects at close camera distances. Non-Euclidean representation also precludes the use of more complex rendering techniques, such as lighting effects.

The authors of [45] extend the method described in [25] to a perspective projection framework. During system setup, a world coordinate frame must be manually inserted into two reference views, by specifying image locations of control points. Five control points establish a projective frame of reference in which the motion of the camera is defined. Line intersections on fiducials are matched across video frames to estimate camera motion.

In [9] a parameterized cuboid structure, which must be present in a viewed scene, serves as a reference object. Once the user has identified in an image six control points, defining the cuboid, the system can estimate both the intrinsic (calibration) and extrinsic (pose) parameters of the camera, by solving a linear system. Experimental results demonstrate successful insertion of virtual objects into both static and dynamic scenes, albeit not without substantial human interaction.

Fully markerless and general techniques are presented in [11] and [41], where virtual object registration is achieved based on structure-from-motion results. Eu-

clidean 3D structure of a viewed scene and camera motion are recovered from an input video sequence. Both of the above methods rely on an image order, dictated by the video sequence, and perform offline batch processing and augmentation of the entire sequence, with no support for online scene recognition or tracking¹. In contrast, this thesis employs automated 3D modelling from unordered images to support live video augmentation.

2.2.1 3D Modelling from Images

The task of structure-from-motion is one of the classic problems in computer vision and as such has been extensively studied. Earlier approaches primarily considered shape recovery from point correspondences in two images [12, 28], while more recent algorithms often deal with larger image sets, which provide wider baseline and richer image data for increased accuracy of results [38, 15, 44, 19].

In [2] an algorithm based on Kalman filtering is applied to recursively recover object shape and camera motion from an image sequence. In [34] a recursive approach is applied to affine reconstruction. A disadvantage of sequential processing of image data lies in its sensitivity to poor initialization, which can significantly affect the accuracy of subsequent computations.

A well-known factorization method, developed in [48], uses singular value decomposition (SVD) to factor a matrix of image measurements into two matrices, which define affine scene geometry and camera poses, respectively. In [25] this measurement matrix is assumed to be complete, i.e., all model points are visible in all images; in [40] missing correspondences are handled by splitting the partially filled measurement matrix into several complete submatrices, applying factorization to each and stitching the results into a single consistent solution.

A widely used approach to the structure-from-motion problem begins with an algebraic initialization of structure and cameras in a projective frame of refer-

¹Offline video augmentation methods, such as those applied in a cinema industry, are commonly referred to as *Match Move*.

ence, using two- or three-view epipolar constraints on image point matches. The constraints are defined by a fundamental matrix or a trifocal tensor, computed for each image pair or triple, respectively. Projective initialization is followed by an upgrade to an affine or Euclidean framework, with camera calibration parameters recovered by autocalibration techniques [21, 37]. Lastly, the solution is refined via an iterative bundle adjustment optimization.

Bundle adjustment refers to the refinement of an image-based reconstruction to produce jointly optimal structure and camera parameter estimates. A solution that is statistically optimal and robust to image noise can be found by iteratively minimizing a cost function that quantifies the model fitting error (the average image reprojection error of 3D model points). A exhaustive survey of the theory of bundle adjustment and applied methods can be found in [49].

My approach to structure-from-motion is based on a simple optimization strategy, suggested in [47]. The projective or affine initialization step is omitted entirely: all of the unknown parameters are estimated directly in a Euclidean framework, by applying bundle adjustment. Experimental results, reported in [47], demonstrate successful convergence of the non-linear optimization in the absence of any *a priori* information about scene structure or camera poses.

Chapter 3

Modelling Reality

The initial stage of the system automatically constructs a 3D representation of an operating environment, computes calibration parameters of the camera and provides means for visually specifying a desired location of a virtual object in its real surroundings. These preliminary computations are performed offline, and are intended to provide supporting data for live video processing.

The modelling algorithm requires as an input a set of reference images of the environment. The images can be acquired by a handheld, uncalibrated camera from spatially separated viewpoints. No assumptions are made about the relative positioning of the cameras; however, certain overlap in the image content is recommended for reliable feature matching (up to about 45° of rotation between adjacent viewpoints has been found acceptable). At least two snapshots are needed to estimate the 3D structure of the viewed content; using more provides a richer visual description of the scene, and thus enables better recognition performance and wider range of camera travel. In my experiments, I have used up to 40 images, covering from 60° to 360° of a surrounding view. The scene is assumed to be rigid and mostly static¹, with no special markers or geometric structures known to be present. The current implementation requires calibration parameters of the camera to remain

¹Individual moving elements (e.g., people in a room) are treated as outliers.

constant during acquisition of reference images and online operation. This requirement can be easily alleviated by adding the necessary unknown parameters to be computed by the corresponding algorithms.

The reference images are used to build a sparse model of the viewed scene. Such a model is a collection of distinctive world points, which appear as SIFT features in the input images, and for which 3D Euclidean coordinates have been computed. Camera poses, corresponding to the reference viewpoints, as well as calibration parameters are computed simultaneously with the scene model. The structure-from-motion computations are followed by the insertion of the virtual object into the modelled environment.

The offline processing is divided into the following steps:

1. Local invariant point features are extracted from the input images.
2. A robust wide baseline matching technique is applied to find two-view feature correspondences, leading to the construction of multi-view matches.
3. A subset of multi-view matches is chosen as an input to an iterative structure-from-motion algorithm, which produces an initial portion of the model.
4. The rest of the matches are used to compute the remaining camera poses and world point coordinates via resectioning and triangulation, respectively.
5. The position, orientation and size of a virtual object, expressed in the coordinate frame of the final model, are defined by the user.

3.1 Image Features

The first objective is to extract local interest points and find their correspondences over multiple images in the input set. A brief overview of the feature extraction algorithm is given below (details can be found in [29, 30]), followed by the discussion of the approach to multi-view feature matching.

3.1.1 Feature Extraction

Candidate feature locations are first identified in spatial and scale domains at extrema of a difference-of-Gaussian (DOG) function. An initial image is repeatedly convolved with Gaussians at different scales. Afterwards Gaussian images, adjacent in the scale pyramid, are subtracted to produce DOG images. Each point in the latter is compared to its neighbours in both image location and scale, in order to find the DOG peaks. At each peak, a detailed model is fit for location, edge response and peak magnitude, rejecting unstable candidates which are sensitive to noise.

Besides image location and scale at which it was found, each stable feature is assigned an image orientation and a descriptor vector, reflecting local image properties. Both orientation and descriptor vector are computed from gradient magnitudes and directions, sampled within a circular Gaussian-weighted window around the feature. To achieve scale invariance, the scale of the feature is used to select a smoothed image from which the samples are drawn. Feature orientation is assigned based on the dominant direction of the gradient samples, corresponding to a peak in a 36-bin histogram².

The length of the descriptor vector varies depending on the number of orientation histograms used to accumulate the samples. Best results are achieved with 128 dimensions, corresponding to a 4×4 sample region with 8 orientations. The descriptors are represented in an orientation-invariant manner by rotating gradient orientations relative to the assigned feature orientation. Lastly, the descriptors are normalized to reduce the effects of illumination changes.

3.1.2 Feature Matching

Once the features have been extracted from the input images, the next step is to establish their reliable correspondences across the image set. For each feature in

²Multiple peaks with similar magnitudes result in multiple features at the same location, but with different orientations.

a reference image, its putative matches are found in each of the remaining images. Next, the tentative set of matches is improved by removing outliers and introducing additional matches.

The best candidate match for a SIFT feature is its nearest neighbour, defined as a feature with a minimum Euclidean distance between descriptor vectors. A simple and effective method can be applied to test the reliability of a best match. It involves comparing Euclidean distances of a nearest neighbour and a second nearest neighbour, both found in the same image. If the ratio of the nearest to the second nearest distance is too high, the nearest neighbour match is discarded as an outlier. The threshold value of 0.8 for the distance ratio has been found to work well in practice, discarding many true outliers and relatively few inliers. In a sense, the distance ratio tests the ambiguity of a best match: in a high-dimensional feature space incorrect matches are more likely to have competing candidates with similar distances, than correct ones.

Large numbers of features found in images, as well as the high dimensionality of their descriptors, make exhaustive search for nearest neighbours extremely inefficient. In order to improve efficiency without sacrificing the quality of matches, I employ the approximate Best-Bin-First (BBF) algorithm, based on a k-d tree search [5]. BBF finds true nearest neighbours with a high probability and enables feature matching to run in real time.

A k-d tree of image features is constructed as follows. Starting with a complete set of features, the set is split in two on the dimension in which the feature descriptors exhibit the highest variance. The mean value of the descriptors in that dimension is chosen as the division point. An internal node is created to store the dimension index and the division value. This process is repeated recursively on the divided subsets, with tree leaves eventually created to contain a single feature. Each dimension is chosen for division at most once, which works well for high-dimensional spaces and simplifies computation of search boundaries during tree traversal.

The BBF search is performed on the k-d tree to find nearest and second nearest neighbours for each feature, in all images other than the one from which the query feature originated. Each feature in the tree stores an index of an image where it was found. For each query the search algorithm maintains two parallel arrays of nearest and second nearest neighbours, indexed by an image number.

First, the tree is traversed down to the leaf, and the Euclidean distance is computed between the leaf and the query feature. The search then backtracks, examining branches that have not yet been explored, in the order of their increasing distance to the query. This order is maintained with a heap-based priority queue. Each time a branch decision is made at an internal node, an entry is added to the queue. The entry contains a child node not being examined next and a distance between the query and the current search boundary, the latter defined by the path traversed so far.

The search terminates after a predetermined number (a few hundred) of leaf checks have been made. Although this termination strategy does not guarantee that matches are found in all images for each feature, it has worked quite effectively for the experimental data sets, producing hundreds of putative two-view matches for each image pair.

3.1.3 Verifying Geometric Consistency

A set of putative correspondences is gathered from all two-view matches that satisfy the distance ratio check and are one-to-one consistent (two features in a match are each other’s nearest neighbours). These matches undergo further outlier removal by enforcing the epipolar constraint. For each image pair the constraint is defined as

$$\mathbf{x}_i^T F_{ij} \mathbf{x}_j = 0 \tag{3.1}$$

where $\mathbf{x}_i = [u_i \ v_i \ 1]^T$ and $\mathbf{x}_j = [u_j \ v_j \ 1]^T$ are homogeneous coordinates of the matched feature points in images I_i and I_j , respectively, and F_{ij} is the fundamental matrix.

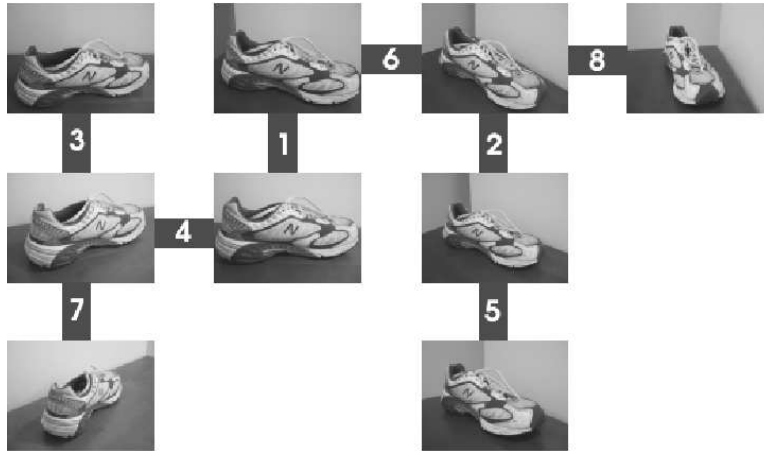


Figure 3.1: A spanning tree, constructed on the set of 9 sneaker images. Edge numbers represent the order in which image pairs were processed. Note the viewpoint proximity of the adjacent images.

The computation of F between each pair of N images has $\binom{N}{2}$ complexity, thus quickly becoming prohibitively expensive with increasing N . Therefore I have decided to apply a more selective approach, based on [42], which is linear in the number of images. A greedy algorithm is used to select a subset of $N - 1$ image pairs for computing F . The algorithm constructs a spanning tree on the image set (see Figure 3.1 for an example). Starting with the two images that have the most putative matches, the algorithm joins the images with an edge, uses RANSAC [14] to compute F consistent with the majority of matches and removes the disagreeing minority as outliers. The image pair with the next highest number of matches is processed next, subject to the constraint that joining these images does not introduce a cycle in the tree. Once the tree is completed, multi-view correspondences can be easily found by traversing the tree and stitching together two-view matches.

The spanning tree method limits expensive epipolar computations to be performed only on more promising candidates. These are typically image pairs from less separated viewpoints, which lead to a larger and cleaner set of candidate matches.

The computations at each tree edge deserve a closer look. First, F is repeat-

edly computed via the normalized 8-point algorithm [21] for a number of random RANSAC samples. Since computing F for each possible sample of 8 matches is computationally infeasible, the number of samples is taken to be sufficiently high to give a success probability in excess of 99%. The success probability Γ is defined as

$$\Gamma = 1 - (1 - (1 - o)^m)^n \quad (3.2)$$

where o is the fraction of outliers, m is the number of matches in each sample and n is the number of samples [37]. For data sets used in this thesis it typically took a few hundred RANSAC samples to reach the desired success probability.

Once Γ has reached its desired value or n exceeds the maximum allowable threshold (2,000 samples), F with the largest number of inliers is selected as the correct solution. A match between two image points is considered an inlier if the sum of perpendicular distances from the points to the corresponding normalized epipolar lines, defined by F , is below a predefined threshold of a few pixels. Matches inconsistent with F are discarded as outliers, after which the algorithm iteratively finds additional inliers and refines the estimate of F .

Additional point matches between images I_i and I_j are found as follows. First, for each unmatched \mathbf{x}_i its nearest neighbour \mathbf{x}_j is found, which has not yet been matched and is consistent with the computed F_{ij} . The epipolar consistency constraint defines a search space for a match in I_j along an epipolar line. The next step is to narrow down this search space by also looking for consistency in the nearby image content. To this end, 3 points are found in I_i among the already matched inliers, which are closest to \mathbf{x}_i in terms of image distance. Let these points be $\mathbf{n}_{i,k}$, matched to $\mathbf{n}_{j,k}$ in I_j , where $k \in [1, 2, 3]$. The verification compares

1. feature scale ratio between \mathbf{x}_i and \mathbf{x}_j to that between $\mathbf{n}_{i,k}$ and $\mathbf{n}_{j,k}$, and
2. scaled image distance between \mathbf{x}_i and $\mathbf{n}_{i,k}$ to that between \mathbf{x}_j and $\mathbf{n}_{j,k}$,

which should be similar for all k , i.e., the ratio of a smaller to a larger value should be above a threshold of 0.8. Essentially, image features nearby \mathbf{x}_i in I_i are expected

to match features nearby \mathbf{x}_j in I_j , having undergone a scale change similar to that between \mathbf{x}_i and \mathbf{x}_j .

3.2 Scene Structure and Camera Parameters

Once the multi-view correspondences have been established between the image points, the next objective is to compute world coordinates of the corresponding 3D points, calibration parameters and camera poses for each reference image.

In homogeneous coordinates, a linear relationship between a 3D point $\mathbf{X}_j = [X_j \ Y_j \ Z_j \ 1]^T$ and its 2D projection $\mathbf{x}_{ij} = [u_{ij} \ v_{ij} \ 1]^T$ in an image I_i is defined as

$$\mathbf{x}_{ij} \sim P_i \mathbf{X}_j \quad (3.3)$$

where \sim denotes equality up to a scale factor, and P_i is a 3×4 projective camera matrix of the form

$$P_i = K[R_i \ \mathbf{t}_i] \quad (3.4)$$

In the above equation, R_i and \mathbf{t}_i are a rotation matrix and a translation vector of the world frame relative to the camera frame for I_i . Matrix K contains calibration parameters:

$$K = \begin{bmatrix} f & 0 & p_u \\ 0 & af & ap_v \\ 0 & 0 & 1 \end{bmatrix} \quad (3.5)$$

where f , a and $[p_u \ p_v]^T$ are a focal length, an aspect ratio and principal point coordinates, respectively.

Using the inhomogeneous representation of the points, a 2D projection $\mathbf{x}_{ij} = [u_{ij} \ v_{ij}]^T$ (in an image coordinate frame) of a 3D point $\mathbf{X}_j = [X_j \ Y_j \ Z_j]^T$ (in a world coordinate frame) is obtained via the non-linear perspective projection function $\Pi(\cdot)$:

$$\mathbf{x}_{ij} = \Pi(R_i \mathbf{X}_j + \mathbf{t}_i) = \Pi\left(\begin{bmatrix} X'_j \\ Y'_j \\ Z'_j \end{bmatrix}\right) = \begin{bmatrix} f \frac{X'_j}{Z'_j} + p_u \\ af \frac{Y'_j}{Z'_j} + ap_v \end{bmatrix} \quad (3.6)$$

The structure-from-motion problem is formulated as a minimization of squared reprojection errors over all camera parameters and world point coordinates, given image projections of the world points:

$$\min_{\mathbf{a}_{ij}} \sum_i \sum_j \|w_j(\mathbf{x}_{ij} - \tilde{\mathbf{x}}_{ij})\|^2 \quad (3.7)$$

where w_j is a confidence weight assigned to \mathbf{X}_j , $\tilde{\mathbf{x}}_{ij} = [\tilde{u}_{ij} \ \tilde{v}_{ij}]^T$ are the measured image coordinates of \mathbf{X}_j in I_i , and \mathbf{x}_{ij} are the estimated image coordinates of \mathbf{X}_j , computed according to (3.6). The vector $\mathbf{a}_{ij} = [\mathbf{X}_j^T, f, a, p_u, p_v, \mathbf{p}_i^T]^T$ contains all of the unknown structure-and-motion parameters, including the 6 degrees of freedom of the camera pose \mathbf{p}_i .

The solution to (3.7) is found iteratively using the Levenberg-Marquardt (LM) algorithm [39], which computes incremental updates to the unknown parameters at each iteration³. Derivatives which form the Jacobian matrix are approximated numerically.

After a predefined number of initial iterations, the algorithm begins to reduce the confidence weights of world points with high reprojection errors, thus lowering the contribution of likely outliers to the final solution. At each iteration, the weight w_j is reduced if \mathbf{X}_j has an image residual e_{ij} more than twice as large as the total average reprojection error E :

$$E = \sqrt{\frac{\sum_i \sum_j \|w_j(\mathbf{x}_{ij} - \tilde{\mathbf{x}}_{ij})\|^2}{N}} \quad (3.8)$$

$$e_{ij} = \sqrt{\|\mathbf{x}_{ij} - \tilde{\mathbf{x}}_{ij}\|^2} \quad (3.9)$$

$$w_j = \min\left(w_j, \frac{2E}{e_{ij}}\right) \quad (3.10)$$

where N is the total number of image points. For the rest of the world points with low image residuals, the weights are reset to the default value of 1.0.

³Updates to the camera orientation take form of three rotations: around the x , y and z axes. The order of these rotations does not matter, since the angles are small at each LM iteration and any minor errors are corrected by future convergence.

To initialize the unknowns, reasonable default values are guessed for the calibration parameters. Image points are backprojected to the xy plane of the world frame, and all of the cameras are placed at the same distance along the z axis of the world frame, directly facing the points and having the same orientation. Following this simple initialization, the algorithm generally takes a few dozen iterations to converge to a reasonable solution (Figure 3.2).

Besides its simplicity, the direct bundle adjustment approach is attractive for several reasons. It produces a statistically optimal result, at the same time avoiding accumulation of error due to algebraic manipulations; it can robustly handle noisy measurements and missing correspondences, and is flexible in the number of parameters to be computed; and it can handle scenes of arbitrary geometry while requiring no *a priori* knowledge of scene structure or camera parameters.

3.2.1 Incremental Model Construction

The iterative optimization method, described above, has two major limitations. First, computation time increases dramatically with the number of unknown parameters. I have implemented a standard (non-sparse) version of the LM algorithm, in which the solution of normal equations is $O(N^3)$, a step that gets repeated many times. Second, the algorithm has trouble converging if the viewing angle captured by the cameras is too wide, i.e., the farthest cameras are more than 90° apart. I deal with both of these problems by splitting the input data, based on a method described in [42]. A subset of the point matches is selected from a limited number of images for the computation of the partial model, then the remaining matches are used to incrementally update and refine the model.

I begin by ordering the reference images, based on the number of their two-view point matches which have passed the epipolar constraint test. The two images with the most matches are placed at the start of the ordered list; the next image to be added is adjacent to the already ordered images by an edge in the spanning

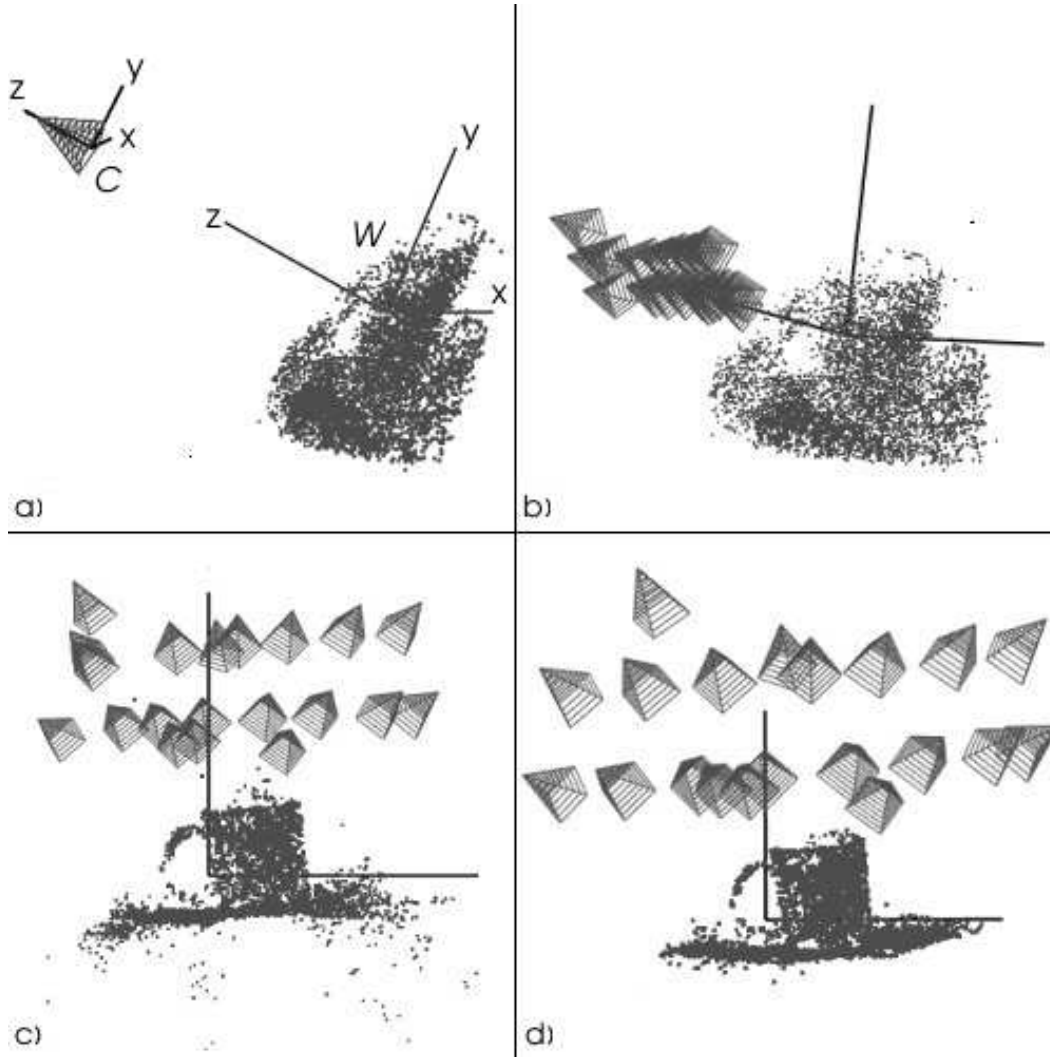


Figure 3.2: *Building a model of a coffee mug placed on top of a magazine, from 20 reference images: (a) initialized structure and cameras ($E = 62.5$ pixels); (b) results after 10 iterations ($E = 4.2$ pixels); (c) results after 20 iterations ($E = 1.7$ pixels); (d) final results after 50 iterations ($E = 0.2$ pixels). Bundle adjustment was used to simultaneously compute poses of all 20 cameras and coordinates of 100 world points with the most image matches (the rest of the world points were triangulated following LM convergence).*

tree (see Section 3.1.3), and has the largest number of matches among all adjacent images.

The first few (5 to 10) images from the ordered list are selected for the computation of an initial model. These images are likely to be in close spatial relationship with each other, as they have many shared points. The exact number of initial images depends on the data set: if the reference images were acquired from viewpoints closely located to each other, more images should be used for initial reconstruction to cover a sufficiently wide baseline for accurate results. To reduce problem size, at most 100 multi-view matches, linking the most image points, are used from the selected images as an input to the LM algorithm.

A partial model is computed from image points in $I_0 \dots I_{i-1}$ via bundle adjustment, as described at the beginning of this section. Afterwards, the correspondences $(\tilde{\mathbf{x}}_{ij}, \mathbf{X}_j)$ between already localized 3D world points and 2D image points in the next ordered image I_i are used to compute the camera pose for I_i . I use LM to minimize the residual sum:

$$\min_{\mathbf{p}_i} \sum_j \|w_{ij}(\mathbf{x}_{ij} - \tilde{\mathbf{x}}_{ij})\|^2 \quad (3.11)$$

where w_{ij} is a weight assigned to the measurement $\tilde{\mathbf{x}}_{ij}$. In my implementation, it is set to 1.0 for all image points⁴. The camera pose parameters \mathbf{p}_i are now the only unknowns. They are initialized to an already computed camera pose, which yields the lowest average reprojection error for the match set $(\tilde{\mathbf{x}}_{ij}, \mathbf{X}_j)$ (and therefore is likely to be close to \mathbf{p}_i).

Following the computation of each new camera, the unknown coordinates of world points, which appear more than once in the already processed images, are determined using straightforward triangulation [21].

Once all of the images have been processed, additional world point outliers are removed from the model. A world point is deemed an outlier if

⁴Alternatively, weights can vary to reflect the accuracy of feature localization, e.g. features at different scales can have different image uncertainties.

1. its reprojection error is too high;
2. it is located behind all cameras by which it is viewed; or
3. it is located too far from the majority cluster of world points.

Outliers can be caused by image noise leading to poor feature localization, or by occasional mismatches which have survived the epipolar constraint test.

Lastly, the full model is refined via the bilinear alternation strategy [32]. First, all camera poses are recomputed while holding world points constant, then all world points are refined while fixing the cameras, and this process is repeated until convergence. The same function as in (3.7) is minimized, but because cameras and points are handled independently, the number of unknown parameters is greatly reduced for each individual computation (6 unknowns for a camera, 3 for a world point). On the downside, bilinear alternation usually takes many iterations to converge, since only part of the data is considered at a time. The effects of bilinear alternation, observed during the experiments, are discussed in Section 5.1.1.

3.3 Inserting Virtual Content

The insertion of a virtual object into the real world is achieved by adjusting its projection in the reference images until it appears correctly rendered. First, the 3D coordinates of the virtual frame origin V are located via a simple triangulation method, as follows. The projection of V is specified in one of the reference images with a click of a mouse button (the virtual frame is “anchored” in 2D). Afterwards, the relative depth of V is adjusted by switching to a different view and moving the corresponding projection of V along an epipolar line imposed by the anchoring view. This is equivalent to moving V along a line connecting the camera centre and the projection of V in the anchoring image (Figure 3.3).

Next, the user is able to fine-tune the position, orientation and size of the virtual object in controlled variable-size increments. Figure 3.4 shows an example

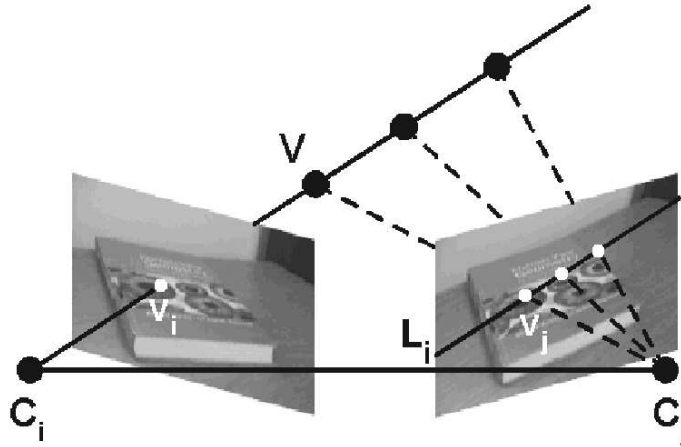


Figure 3.3: *The placement of the virtual frame origin \mathbf{V} in 3D is achieved by anchoring its projection \mathbf{v}_i in the image I_i and adjusting its projection \mathbf{v}_j in the image I_j along the epipolar line \mathbf{L}_i .*

of an insertion of the virtual frame and the pose adjustment. The virtual object is rendered onto the reference images using previously computed camera parameters. At any time the user can switch between the images to view the corresponding virtual projection, either as a 3D frame or a fully rendered object. Note that the geometric relationships between the real world, the virtual object and the cameras are defined in the same generic units, so that there is no need to recover the absolute scale of the real-world model.

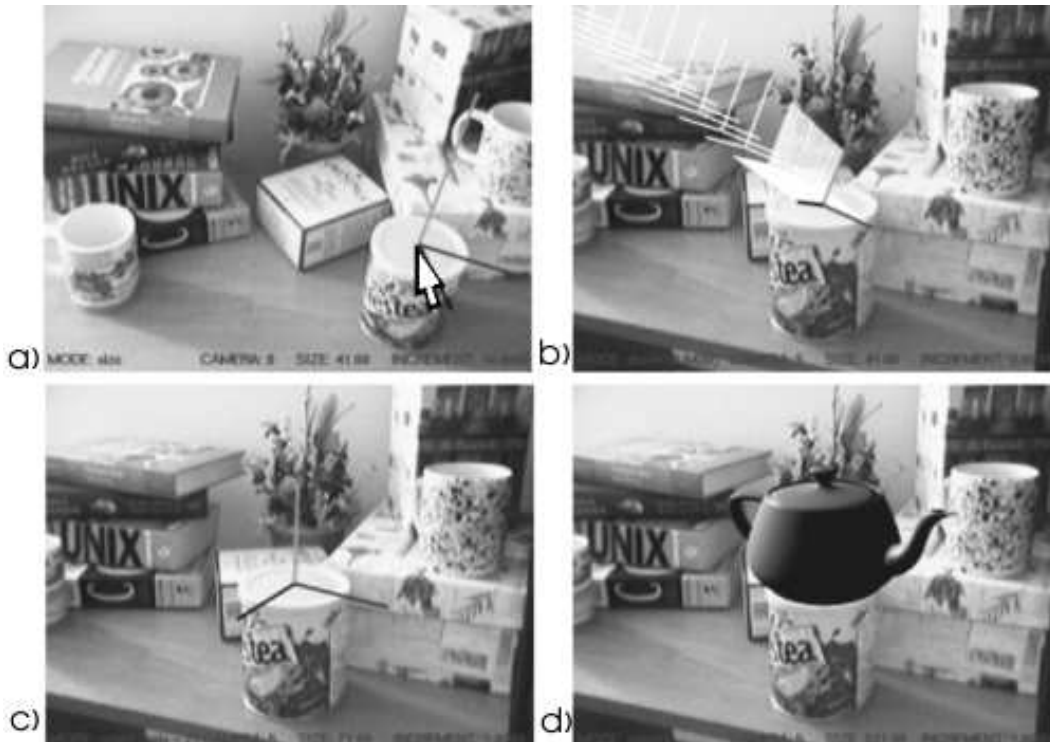


Figure 3.4: *Insertion of the virtual object into a desk scene: (a) initial placement into one of the reference images by specifying the desired location of the frame's origin; (b) the frame's trajectory along an epipolar line in another image; (c) subsequent orientation and size adjustments; (d) the rendered teapot is properly registered. The user controls 7 degrees of freedom (translation, orientation and size) of the virtual object relative to the world frame.*

Chapter 4

Camera Tracking

As pointed out in Section 1.2, the pose of the camera relative to the world coordinate frame must be accurately determined, in order to register a virtual object in a current video frame. Camera tracking refers to the computation of the scene-to-camera transformation while the camera is in use for live AR. Here tracking is model-based, as it relies on recognizing a projection of a previously constructed scene model in each frame of a video sequence. Below is a step-by-step summary of online computations:

1. SIFT features are extracted from the current frame of the video sequence.
2. The features from the video frame are matched against the projections of the model points in the reference images, leading to a set of 2D-to-3D correspondences.
3. The found matches are used to compute the current camera pose via non-linear optimization.

The online computations share a lot of similar building blocks with the algorithms described in the previous chapter, with special care taken to improve both speed and accuracy.

4.1 From Features to Camera

To initialize the system for camera tracking, a k-d tree is built from all of the image points of the constructed scene model. These are 2D projections of 3D world points, which appear as SIFT features in the reference images. Thus, for each world point at least two corresponding SIFT features from different reference images are stored in the tree¹. Each leaf in the k-d tree stores an image point and contains a link to the corresponding world point, an index of the reference image from which the feature originated, and a link to the camera pose computed for that image.

As part of online processing, the BBF search is performed on the k-d tree, to find a nearest and a second nearest neighbour pair for each newly detected feature, this time subject to the two neighbours belonging to different world points. The reliability of the best match is tested by comparing its Euclidean distance to that of the second best match.

Tracking failure is assumed if too few reliable matches are found. The minimum number of required matches is 3: with 2 image measurements per match, 6 degrees of freedom of the camera motion can be recovered. Tracking failure usually occurs when all or most of the model disappears out of sight, the model is too far from the camera or the frame contains too much motion blur.

As in Section 3.2.1, the camera pose parameters are computed by LM from the found 2D-to-3D matches $(\tilde{\mathbf{x}}_{tj}, \mathbf{X}_j)$, by minimizing the average reprojection error:

$$\min_{\mathbf{p}_t} \sum_j \|w_{tj}(\mathbf{x}_{tj} - \tilde{\mathbf{x}}_{tj})\|^2 \quad (4.1)$$

where t is an index of the current video frame. I initialize \mathbf{p}_t to \mathbf{p}_{t-1} , as there is likely to be little camera displacement between two consecutive video frames. For the first frame of the video sequence or the one immediately after tracking failure, as an initial guess I use the camera pose of a reference image with the lowest average reprojection error computed for $(\tilde{\mathbf{x}}_{tj}, \mathbf{X}_j)$.

¹To save memory, these features can be combined into a single representation, one per 3D point.

RANSAC is applied to make camera pose computations more robust to feature mismatches. Error minimization is performed for each RANSAC sample, and the final solution, consistent with the majority of matches, is refined using all of the inliers. Despite its iterative nature, this technique has proven to be sufficiently fast for online use. Since there are only 6 unknown parameters, corresponding to the 6 degrees of freedom of a camera pose, LM iterations are rapidly executed. Moreover, the non-linear computation of \mathbf{p}_t requires the minimum of only 3 matches, so that only a few random RANSAC samples are needed to stumble upon a good solution.

4.2 Reducing Jitter

The solution to (4.1) provides a reasonable estimate of the camera pose, yet typically leads to a jitter of the virtual projection in the video sequence, particularly noticeable when the camera is fully or nearly stationary. This inaccuracy can be a result of image noise, as well as too few or unevenly distributed point matches. In addition, the surface of the error function may be flat near a local minimum, as it may be difficult to distinguish between slight changes in rotation and translation parameters.

To stabilize the solution, I modify (4.1) by adding a regularization term which favours minimum camera motion between consecutive video frames:

$$\min_{\mathbf{p}_t} \sum_j \|w_{tj}(\mathbf{x}_{tj} - \tilde{\mathbf{x}}_{tj})\|^2 + \alpha^2 \|W(\mathbf{p}_t - \mathbf{p}_{t-1})\|^2 \quad (4.2)$$

where W is a 6×6 diagonal matrix of prior weights on the camera parameters, and α is a smoothness factor, which controls the tradeoff between the current measurements and the desired estimate. Each diagonal entry of W is set to the inverse of the standard deviation of the corresponding parameter, reflecting the expected frame-to-frame change in the camera pose (e.g. a few degrees for a change in rotation).

Instead of setting α to a constant value, I adjust it separately for each video frame, in order to prevent oversmoothing of camera motion (which would result in a virtual object drifting behind a faster moving scene). The amount of smoothing

is determined by controlling its contribution to the total reprojection error: the contribution is forced to stay within range of combined image feature noise. This can be expressed by the inequality

$$\alpha^2 \|W(\mathbf{p}_t - \mathbf{p}_{t-1})\|^2 \leq \sigma^2 N \quad (4.3)$$

where N is the number of image points and σ is the estimated uncertainty of an image measurement (a fraction of a pixel). It follows that the maximum allowable amount of smoothing is

$$\alpha^2 = \frac{\sigma^2 N}{\|W(\mathbf{p}_t - \mathbf{p}_{t-1})\|^2} \quad (4.4)$$

Because \mathbf{p}_t is unknown, α cannot be computed in advance; instead, it is gradually adjusted during LM iterations, as follows.

At first, \mathbf{p}_t is computed using (4.1), i.e. with $\alpha = 0$. Once a local minimum has been reached, the search explores its immediate neighbourhood, looking for a regularized solution. This is done by executing a few additional LM iterations, this time solving (4.2) with α recomputed at each iteration as per (4.4), given the most recent estimate of \mathbf{p}_t . The search for a regularized solution terminates when $\mathbf{p}_t - \mathbf{p}_{t-1}$ becomes very small (which is typical for a stationary camera), or if the search is about to wander too far from the original local minimum (the difference between the errors of regularized and unregularized solutions becomes too large).

Intuitively, as much smoothing as possible is applied while still trying to agree with the measured data, within the bounds of its uncertainty. As a result, larger values of α are used for slower frame-to-frame motions, significantly reducing jitter, while fast and abrupt camera motions are handled without drift.

4.3 Speeding Up Model Recognition

SIFT feature extraction is currently the most computationally intensive operation during tracking. In order to make it faster, I have implemented two simple modifications to the standard approach. One of them involves making the SIFT algorithm

skip the lowest level of the scale space pyramid when searching for DOG extrema. This decreases computation time by about a third, at the cost of finding fewer features at finer scales. The number of skipped pyramid levels can also be adjusted dynamically, based on the size of the image portion occupied by the tracked object, or the number of matched features.

The other modification involves extracting features only from the part of a video frame that is likely to contain most of the model projection. The very first frame and those immediately after a tracking failure are processed in full. Otherwise, a bounding rectangle is computed for the next frame from all successfully matched features (RANSAC inliers) in the current frame. The rectangle is centered at their mean image location and is sized up to include all of the features plus some additional space around them to account for frame-to-frame motion. Thus, in the next frame the model is predicted to be found somewhere nearby its current image location. This technique works quite well for recognizing scenes or objects which occupy only a small fraction of an image (Figure 4.1).

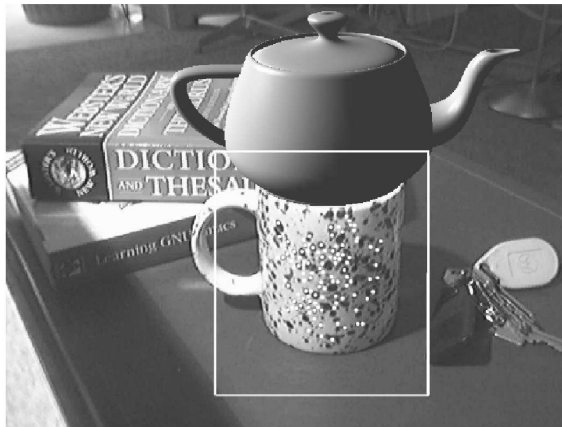


Figure 4.1: *A white rectangle shows which portion of the next video frame will be processed to find SIFT features belonging to the modelled mug. The features that have been successfully detected and matched in the current frame are shown as white dots.*

Chapter 5

Experiments and Results

The system prototype was implemented in C, using OpenGL and GLUT libraries for the graphics engine. ARToolKit [23] libraries were used for video frame capture, as well as for fiducial marker tracking in some of the experiments, described in this chapter (ARToolKit documentation and source code are available for download at <http://www.hitl.washington.edu/artoolkit/>). The system runs under Linux RedHat on an IBM ThinkPad with a Pentium 4-M processor (1.8 GHz) and a Logitech QuickCam Pro 4000 video camera.

In evaluating the performance and capabilities of the system, I aimed to address the following criteria:

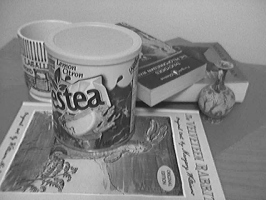

- versatility of the modelling and tracking strategies;
- accuracy and descriptiveness of constructed scene models, sufficient to support reliable model-based camera tracking;
- registration accuracy of virtual content; and
- robustness of the tracking method to occlusions, changes in the environment and unpredictable camera motion.

Table 5.1 provides a summary of several primary data sets collected for system testing (many more were used in the process of system development). The data sets contain reference images of scenes and individual objects of varying size, shape

and geometric complexity. In preparing the experimental data, I attempted to cover a wide range of scene types, as well as different ways of positioning viewpoints from which reference images could be acquired. For example, *boxes* images were taken by cameras carefully lined up in two concentric circles, one above the other, while *sneaker* images were taken from completely random positions. The *library* set was acquired outdoors, with people freely moving in the camera view.





In Section 5.1 I present and discuss the results obtained by the scene modelling approach, highlighting its strengths and weaknesses. Section 5.2 focuses on the performance of the camera tracking method.

Table 5.1: *Data sets used for the modelling and tracking experiments. The “view” column specifies the approximate view range, captured by the images. The last column shows one of the reference images.*

<i>name</i>	<i>images</i>	<i>view</i>	<i>description</i>	<i>image example</i>
<i>tabletop</i>	35	360°	a random collection of objects on a table surface	
<i>boxes</i>	40	360°	two boxes, one rectangular and one oval-shaped	

continued on next page

Table 5.1 – *continued from previous page*

<i>name</i>	<i>images</i>	<i>view</i>	<i>description</i>	<i>image example</i>
<i>mug</i>	40	360°	a coffee mug on top of a magazine	
<i>sneaker</i>	30	180°	a sneaker on top of a magazine	
<i>book</i>	15	90°	a hard cover children's book	
<i>library</i>	17	60°	an entrance to the UBC library	

5.1 On Modelling

Figures 5.1 through 5.6 show the constructed 3D models for each of the data sets from Table 5.1. The modelling algorithm was able to capture planar surfaces of the book and magazine covers, cylindrical shapes of the mugs and the Nestea container, and the rectangular shape of the box with its right angles. The reader is invited to make visual judgements of the realism of each model: although the models are sparse, it should be possible to identify and differentiate various shapes present in

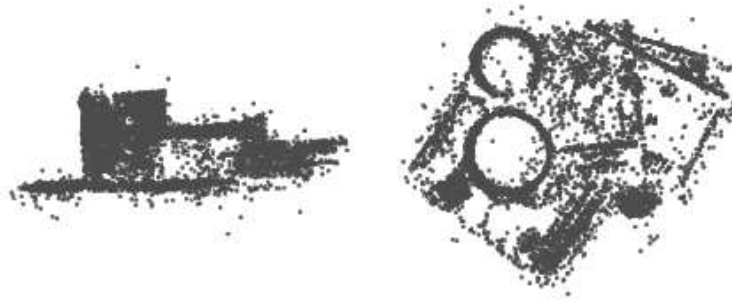


Figure 5.1: *Tabletop model: side view (left) and top view (right).*



Figure 5.2: *Boxes model: side view (left) and top view (right).*

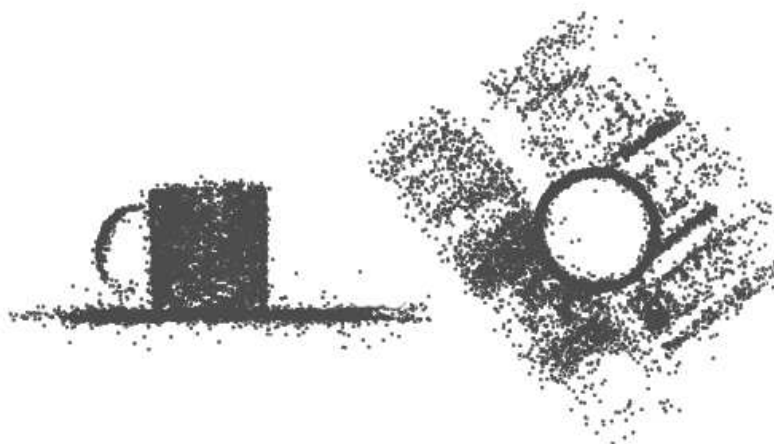


Figure 5.3: *Mug model: side view (left) and top view (right).*



Figure 5.4: *Sneaker model: side view (left) and top view (right).*



Figure 5.5: *Book model: side view (left) and top view (right).*



Figure 5.6: *Library model: side view (left) and top view (right).*

each scene, as well as their spatial layout. Note that the models have a few points on the boundaries, which appear to have strayed from the majority cloud. These are either actual points from the surroundings that do not belong to the model, or simply model outliers which have not been identified as such by the algorithm. One should keep in mind that the models are ultimately intended for camera tracking via model-based scene recognition, rather than for image-based rendering.

Table 5.2: *Performance summary of the modelling algorithm: final average reprojection error (in pixels), number of iterations until convergence to build a partial model, total modelling time (including refinement stage, but excluding feature extraction and matching), number of 3D world points in the model and percentage of detected outliers.*

<i>data set</i>	<i>error</i>	<i>iterations</i>	<i>time</i>	<i>world points</i>	<i>outliers</i>
<i>tabletop</i>	0.3742	48	9 min	9,575	2%
<i>boxes</i>	0.4085	42	15 min	10,710	2%
<i>mug</i>	0.3619	43	20 min	14,112	1%
<i>sneaker</i>	0.7961	42	13 min	9,133	2%
<i>book</i>	0.7668	42	4 min	4,181	3%
<i>library</i>	0.3602	45	6 min	4,915	2%

Table 5.2 shows some numerical results for each data set, such as the final reprojection error, the number of LM iterations, taken to construct an initial model, and the size of the final model. The times taken by SIFT and BBF algorithms, which are not shown in the table, are typically under a minute per data set¹. Recall that only a subset of reference images participates in the initial structure-from-motion

¹Finding additional matches after RANSAC iterations is done with a linear search, which is time-consuming. However, it should be straightforward to implement it using BBF as well.

computations, while the rest of the images are incrementally added afterwards to extend the partial model (Section 3.2.1). For all of the data sets, a minimum of 40 and a maximum of 100 LM iterations were enforced. For the first 20 iterations, no downweighting of 3D world points was performed, to avoid premature penalization of non-planarities. The model refinement stage via bilinear alternation was applied each time 4 new cameras were added to the partial model, in an attempt to avoid error accumulation, as well as at the very end (after all of the cameras were computed and world point outliers were removed). Refinement was limited to 20 iterations², each including recomputation of both camera poses and point coordinates.

5.1.1 Challenges in Shape Reconstruction

Realistic modelling results are more likely to be achieved if reference images are taken by three or more cameras, whose locations are not too close together, and whose lines of sight point in different directions. The variety in camera placement is likely to provide sufficient and non-ambiguous 3D information about a viewed scene, needed for a successful Euclidean reconstruction.

Occasionally bundle adjustment has been observed to converge to a false local minimum, producing a scene model that is reversed in depth. Depth reversal, also known as Necker reversal [21], is an inherent ambiguity in shape perception of objects with Lambertian reflectance under orthographic projection. Essentially, inverting the surface depth and reflecting the light source direction about the line of sight results in an identical image [26]. The depth reversal ambiguity can also occur under perspective projection, likely due to imperfect measurements and too few, closely positioned viewpoints. As a general remedy I employ a simple strategy, which involves reflecting the constructed model about the xy world plane, bundling again with the reflected model used to initialize the algorithm, and selecting a final solution with the lowest average reprojection error.

²Using more than 20 iterations did not result in a noticeable improvement.

Problems can also arise if all of the cameras are positioned at the same height and with the same orientation around their x axis. In such cases, and particularly when modelling objects with simple shapes, the reconstruction can suffer from a bas-relief ambiguity: a confusion between the relative scene depth and the perceived amount of rotation. Concretely, for each image of an object illuminated by a distant light source, there exists an identical image of the affinely transformed object illuminated by a similarly affinely transformed light source [6]. A commonly observed effect of the bas-relief ambiguity is a distorted model that appears vertically stretched and expanded at the bottom, with cameras looking down on it (Figure 5.7). It seems that the algorithm prefers to increase the relative scene depth in favour of not moving the cameras farther apart from their shared initial location (i.e., underestimating the amount of their rotation with respect to the scene).

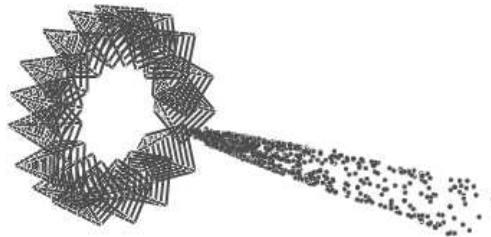


Figure 5.7: *An example of a stretched model: 16 cameras in a perfect circle, surrounding a cylindrical vase.*

Aside from the scenarios mentioned above, bundle adjustment can still converge to a wrong local minimum. This occurs infrequently and may be due to various reasons, such as a few too many feature mismatches. Bad results are easy to spot: the reprojection error is high (above 1 pixel) and the model itself is clearly distorted, with world points widely spread out. A quick and easy fix involves selecting a different number of images for model construction: introducing or removing one or more images often provides sufficient data to stir the algorithm in the right direction. In the current implementation rerunning the modelling algorithm is done by hand; a

future improvement may involve detecting and fixing the problem automatically.

Not surprisingly, the most challenging camera setup is a full 360° view of a scene. To build a model incrementally, the algorithm starts with a cluster of images from close viewpoints, then incorporates information from images neighbouring the initial cluster and so on, until the loop is closed. Thus gradual error buildup is a concern. The result may be somewhat miscalculated camera poses corresponding to one or two images which are processed last (closing the loop). The *boxes* data set is an example of this occurrence (Figure 5.8). Modelling inaccuracies lead to unstable augmentation of the problem side of the scene, such as occasional jumps and jitter of the virtual projection.



Figure 5.8: *The virtual cube appears correctly aligned with the corner of the box in the left and middle images. The right image shows misalignment due to the inaccurately computed camera pose.*

The refinement stage of the modelling algorithm was intended to correct miscalculations due to error buildup. Unfortunately, the effect of the bilinear alternation has been found insignificant and notoriously slow: after the first 20 to 40 iterations, the average reprojection error is typically lowered by about 0.2 pixels, after which the amount of improvement becomes negligible. Matters may be improved by investigating a different refinement approach. One of the alternatives is a sparse LM method which simultaneously refines world point coordinates and camera parameters from a subset of most reliable feature matches.

5.2 On Tracking

Designing experiments for evaluating online performance of an AR system has proven to be not a straightforward matter. The end goal of AR technology is to provide a visually pleasing and stable augmentation effect in a variety of tracking conditions, which is difficult to measure precisely and quantitatively. In order to demonstrate online behaviour of the system, I have directed and produced a number of movies which are available at <http://www.cs.ubc.ca/~skrypnyk/arproject/>. Examples of augmented video frames from the tracking experiments are also shown in Figures 5.9 and 5.10.



Figure 5.9: *Virtual teapot on the coffee mug. The middle frame shows scale invariance. The last two frames demonstrate successful recognition of the partially occluded mug in cluttered scenes.*



Figure 5.10: *Virtual cube on top of the book. The book is partially occluded in the last two frames and shadowed in the last frame.*

Tracking a Mug movie features partial occlusions, object disappearances and reappearances, and jerky camera motion. *Tracking a Library Entrance* was filmed outdoors with a handheld camera, walking in front of the building. Lastly, *Tracking a Table Scene* has a camera make a full 360° trip around the scene. The flickering of the virtual cube at the end of the last movie is due to sporadic motion blur in the video frames. Note that the movies were recorded at a higher frame rate than

that achieved by the system.

5.2.1 Computation Times

An example of current computation times for the camera tracker is given in Table 5.3. More effort in system optimization is required to achieve real-time performance of 20-30 fps. Note that the existing non-optimized implementation of YUV-to-RGB colour conversion, required by ARToolKit, takes about 70 ms.

There is certainly room for improvement, not the least of which is hardware upgrade. Software optimization is hoped to provide additional speedup. If SIFT extraction remains computationally expensive, alternative tracking techniques can be explored. For example, full SIFT-based model recognition can be performed every few frames or whenever needed, while in between using faster narrow baseline feature matching via simple image correlation (assuming smooth frame-to-frame camera motion).

Table 5.3: *Average computation times for a video sequence with 640×480 frame size. The real-world model contains around 5,000 scene points.*

frame acquisition	70 ms
feature extraction (SIFT)	150 ms
feature matching (BBF)	40 ms
camera pose computation (RANSAC and LM)	25 ms
frames per second	3-4

5.2.2 Registration Accuracy

To test the accuracy of virtual object registration and the effect of the jitter reduction approach (Section 4.2), I set up an experiment in which a virtual square

was aligned with a square ARToolkit marker, which was part of a modelled scene (Figure 5.11).



Figure 5.11: *ARToolkit marker in the scene (left). Virtual square, superimposed onto the marker during tracking (right).*

The scene was observed with both a moving and a stationary camera. The corners of the marker in each frame were detected using the ARToolKit routine *arDetectMarker*, which is based on pixel intensity thresholding and subsequent edge detection. Image coordinates of the marker's corners were used as ground truth for the registration of the overlaid virtual square. Figures 5.12 and 5.13 compare image trajectories for one of the corners (the remaining corners produce equivalent results). The offset of about 1.5 pixels between the trajectories of the virtual square and the marker is due to the initial slight error in the manual placement of the virtual square.

Figure 5.12 shows the image trajectories for 300 frames when the camera is stationary (mounted on a table surface). The y -coordinate of the marker's corner is compared to that of the virtual square, for both the regularized and unregularized camera solution. Clearly, the jitter of the virtual corner is significantly reduced by the camera pose regularization. The regularization results in a trajectory which is on the whole smoother than the ground truth, with very few small peaks.

Figure 5.13 demonstrates results when the handheld camera is in arbitrary motion. The image coordinates were again measured over 300 frames. The results

for the first 30 frames, when the camera moved very little, are also shown, for a closer look at the trajectories. The trajectories of the real and virtual corners are in close correspondence, with varying camera motion handled without noticeable drift. The alignment is maintained well even when shaking the camera (frames 110 to 190).

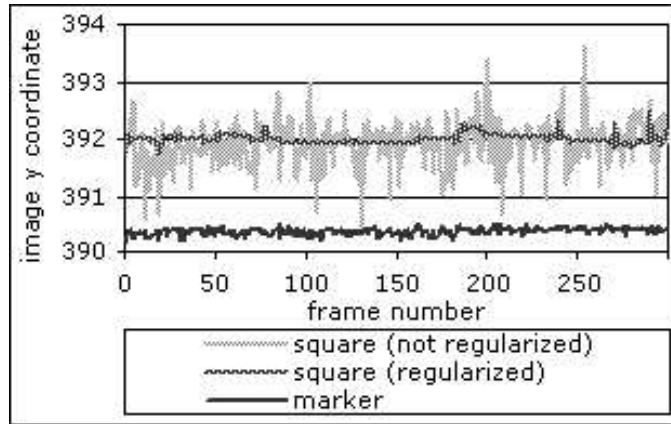


Figure 5.12: Stationary camera results for 300 frames.

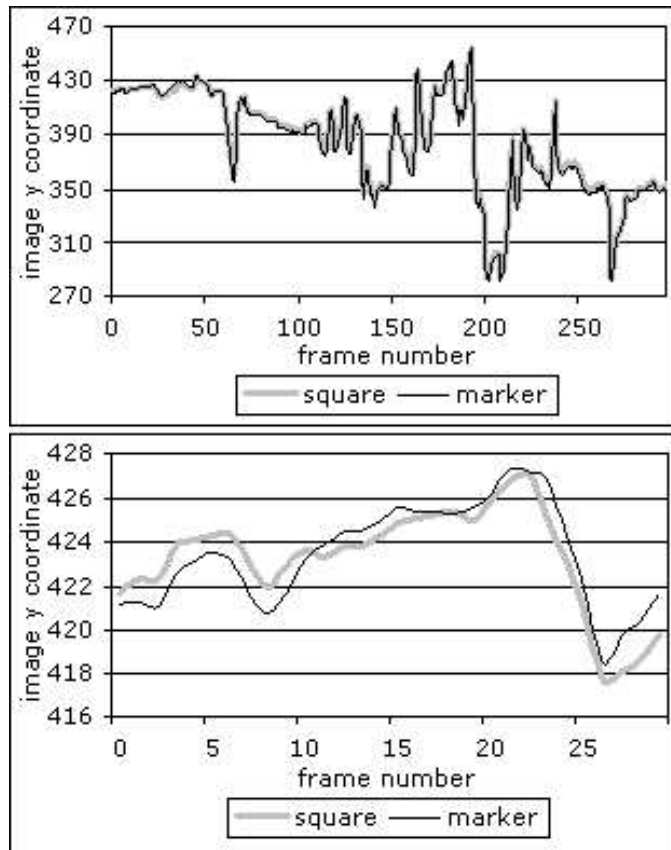


Figure 5.13: Moving camera results for 300 frames (top) and the first 30 frames (bottom).

Chapter 6

Concluding Remarks

In this thesis I explore a versatile and user-friendly approach to Augmented Reality, a technology which registers computer-generated virtual objects into a live video of the real world. The AR system, developed in the course of this project, is aimed at meeting the following goals:

- automate initial camera localization and recovery from tracking failure;
- provide fully markerless, non-intrusive means for vision-based tracking of the camera;
- handle a vast variety of operating environments; and
- increase robustness and stability of the camera tracker.

The system is designed to operate in two stages. The preliminary stage, performed offline, involves construction of a sparse Euclidean model of the operating environment and autocalibration of the camera, via purely passive computer vision techniques. The online stage performs computation of the current camera pose which, together with the results of the offline processing, is used by the graphics engine to render a virtual object onto a video frame. The scene modelling and camera tracking algorithms rely on extraction and matching of stable local features, naturally occurring in images.

The presented technology has potential as a framework for developing affordable, low-maintenance AR applications for rapid deployment. It is suitable for mobile AR in both indoor and outdoor settings, requires no environment modification and very little hardware equipment, and imposes no limitations on the operating environment, camera motion or the nature of virtual content.

6.1 Future Work

This section discusses a number of possible modifications to the current implementation that could, or perhaps should, be considered, in order to improve the system's performance or introduce new capabilities.

The feature extraction algorithm relies on the presence of texture in images. Image examples, shown in this thesis, contain ample texture and as such yield from a few hundred to a few thousand SIFT features per 640×480 image. This is more than adequate for reliable feature matching. Several modifications and improvements can be made to achieve good performance when dealing with image regions of lower contrast. For example, a trivial adjustment is to lower the DOG threshold value for peak localization, which would make feature extraction more sensitive to intensity variations (although image noise may become more of a concern). SIFT features can be combined with contour-based image descriptors (a few of which are mentioned in Section 2.1.1), which would make the system respond equally well to both object textures and shapes.

Currently, the implementation of the camera tracker runs at 3-4 fps on average, which is too slow for real-time operation. The main bottleneck in online processing is acquisition of video frames and feature extraction. Future development efforts should include overall system optimization, involving both hardware and software improvements.

As mentioned in Section 5.1.1, accurate 360° shape recovery has been a challenge due to gradual error accumulation, which bilinear alternation was unable

to correct. Future improvements should include the investigation of alternative methods for incremental model construction and refinement.

The system has been able to achieve successful modelling and recognition of scenes of varying size and complexity, from handheld objects to rooms and buildings. The next step in performance testing must focus on the system scalability for operation in large environments, such as a campus or a museum. A potential enhancement may involve modelling individual buildings, rooms or objects, and providing a mechanism for online switching between these models as the user travels around his or her surroundings.

It should be pointed out that the discussed techniques do not provide support for handling occlusion of inserted virtual content by real objects in the world. To achieve this effect, a dense model of the observed scene is required. The construction of fully textured models from images is an important subject of research in computer vision, but it is beyond the scope of this thesis.

Bibliography

- [1] Mirko Appel and Nassir Navab. Registration of technical drawings and calibrated images for industrial augmented reality. *Machine Vision and Applications*, 13(3):111–118, 2002.
- [2] A. Azarbayejani, B. Horowitz, and A. Pentland. Recursive estimation of structure and motion using relative orientation constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 294–299, 1993.
- [3] R. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.
- [4] R. Azuma, Y. Baillet, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001.
- [5] Jeffrey S. Beis and David G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, 1997.
- [6] Peter N. Belhumeur, David J. Kriegman, and Alan L. Yuille. The bas-relief ambiguity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1060–1066, 1997.
- [7] Serge Belongie, Jitendra Malik, and Jan Puzicha. Matching shapes. In *Proceedings of the 8th IEEE International Conference on Computer Vision*, pages 454–463, 2001.
- [8] M. Billinghurst, H. Kato, and I. Poupyrev. The MagicBook: a transitional AR interface. *Computers and Graphics*, 25(5):745–753, 2001.
- [9] C.S. Chen, C.K. Yu, and Y.P. Hung. New calibration-free approach for augmented reality based on parameterized cuboid structure. In *Proceedings of the 7th International Conference on Computer Vision*, pages 30–37, 1999.

- [10] Kar Wee Chia, Adrian David Cheok, and Simon J.D. Prince. Online 6 DOF augmented reality registration from natural features. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 305–313, 2002.
- [11] Kurt Cornelis, Marc Pollefeys, Maarten Vergauwen, and Luc Van Gool. Augmented reality using uncalibrated video sequences. *Lecture Notes in Computer Science*, 2018:144–160, 2001.
- [12] Olivier D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proceedings of the European Conference on Computer Vision*, pages 563–578, 1992.
- [13] V. Ferrari, T. Tuytelaars, and L. Van Gool. Markerless augmented reality with a real-time affine region tracker. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*, pages 87–96, 2001.
- [14] M. Fischler and R. Bolles. RANdom SAmple Consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications of the Association for Computing Machinery*, 24(6):381–395, 1981.
- [15] Andrew W. Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In *Proceedings of the European Conference on Computer Vision*, pages 311–326, 1998.
- [16] M. Fjeld and B.M. Voegtli. Augmented Chemistry: an interactive educational workbench. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 259–321, 2002.
- [17] Y. Genc, S. Riedel, F. Souvannavong, C. Akinlar, and N. Navab. Marker-less tracking for AR: a learning-based approach. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 295–304, 2002.
- [18] Michael Grafe, Raphael Wortmann, and Holger Westphal. AR-based interactive exploration of a museum exhibit. In *Proceedings of the First IEEE International Augmented Reality Toolkit Workshop*, 2002.
- [19] Mei Han and Takeo Kanade. Scene reconstruction from multiple uncalibrated views. Technical Report CMU-RI-TR-00-09, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 2000.
- [20] C.J. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

- [21] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
- [22] William A. Hoff, Khoi Nguyen, and Torsten Lyon. Computer vision-based registration techniques for augmented reality. In *Proceedings of Intelligent Robots and Computer Vision XV, SPIE*, pages 538–548, 1996.
- [23] Hirokazu Kato and Mark Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, pages 85–94, 1999.
- [24] Georg Klein and Tom Drummond. Robust visual tracking for non-instrumented augmented reality. In *Proceedings of the 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 113–122, 2003.
- [25] Kiriakos N. Kutulakos and James R. Vallino. Calibration-free augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):1–20, 1998.
- [26] M.S. Langer and H.H. Bühlhoff. Measuring visual shape using computer graphics psychophysics. In *Proceedings of the Eurographics Workshop on Rendering Techniques*, pages 1–10, 2000.
- [27] Vincent Lepetit, Luca Vacchetti, Daniel Thalmann, and Pascal Fua. Fully automated and stable registration for augmented reality applications. In *Proceedings of the 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 93–102, 2003.
- [28] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(10):133–135, 1981.
- [29] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision*, pages 1150–1157, 1999.
- [30] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [31] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

- [32] Shyjan Mahamud, Martial Hebert, Yasuhiro Omori, and Jean Ponce. Provably-convergent iterative methods for projective structure from motion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1018–1025, 2001.
- [33] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, pages 384–393, 2002.
- [34] P.F. McLauchlan, I.D. Reid, and D.W. Murray. Recursive affine structure and motion from image sequences. In *Proceedings of the European Conference on Computer Vision*, pages 217–224, 1994.
- [35] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Proceedings of the European Conference on Computer Vision*, pages 128–142, 2002.
- [36] Randal C. Nelson and Andrea Selinger. Large-scale tests of a keyed, appearance-based 3-d object recognition system. *Vision Research*, 38(15):2469–88, 1998.
- [37] Marc Pollefeys. 3D modeling from images. Tutorial in conjunction with the *European Conference on Computer Vision*, 2000.
- [38] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999.
- [39] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.
- [40] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. Submitted to the *International Journal of Computer Vision*, 2004.
- [41] Harpreet S. Sawhney, Y. Guo, J. Asmuth, and Rakesh Kumar. Multi-view 3D estimation and applications to Match Move. In *Proceedings of the IEEE Workshop on Multi-View Modeling and Analysis of Visual Scenes*, pages 21–28, 1999.
- [42] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *Proceedings of the 7th European Conference on Computer Vision*, pages 414–431, 2002.

- [43] M. Schnaider, B. Schwald, H. Seibert, and T. Weller. Medarpa - a medical augmented reality system for minimal-invasive interventions. In *Proceedings of the 11th Medicine Meets Virtual Reality Conference - NextMed: Health Horizon*, pages 312–314, 2003.
- [44] Steven M. Seitz and Charles R. Dyer. Complete scene structure from four point correspondences. In *Proceedings of the 5th International Conference on Computer Vision*, pages 330–337, 1995.
- [45] Yongduek Seo and Ki Sang Hong. Calibration-free augmented reality in perspective. *IEEE Transactions on Visualization and Computer Graphics*, 6(4):346–359, 2000.
- [46] Ivan E. Sutherland. A head-mounted three-dimensional display. In *Proceedings of the Fall Joint Computer Conference*, pages 757–764, 1968.
- [47] Richard Szeliski and Sing Bing Kang. Recovering 3D shape and motion from image streams using non-linear least squares. Technical report, Cambridge Research Laboratory, 1993.
- [48] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams: a factorization method, full report on the orthographic case. Technical Report CMU-CS-92-104, CMU, March 1992.
- [49] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – a modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.
- [50] Mihran Tuceryan, Douglas S. Greer, Ross T. Whitaker, David E. Breen, Chris Crampton, Eric Rose, and Klaus H. Ahlers. Calibration requirements and procedures for a monitor-based augmented reality system. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):255–273, September 1995.
- [51] V. Vlahakis, N. Ioannidis, J. Karigiannis, M. Tsotros, M. Gounaris, D. Stricker, T. Gleue, P. Daehne, and L. Almeida. Archeoguide: an augmented reality guide for archaeological sites. *IEEE Computer Graphics and Applications*, 22(5):52–60, 2002.
- [52] Annie Yao and Andrew Calway. Robust estimation of 3-D camera motion for uncalibrated augmented reality. Technical Report CSTR-02-001, Department of Computer Science, University of Bristol, March 2002.