What and Where: 3D Object Recognition with Accurate Pose

Iryna Gordon and David G. Lowe

Computer Science Department,
University of British Columbia
Vancouver, BC, Canada
lowe@cs.ubc.ca

Abstract. Many applications of 3D object recognition, such as augmented reality or robotic manipulation, require an accurate solution for the 3D pose of the recognized objects. This is best accomplished by building a metrically accurate 3D model of the object and all its feature locations, and then fitting this model to features detected in new images. In this chapter, we describe a system for constructing 3D metric models from multiple images taken with an uncalibrated handheld camera, recognizing these models in new images, and precisely solving for object pose. This is demonstrated in an augmented reality application where objects must be recognized, tracked, and superimposed on new images taken from arbitrary viewpoints without perceptible jitter. This approach not only provides for accurate pose, but also allows for integration of features from multiple training images into a single model that provides for more reliable recognition[1].

1 Introduction

Many existing approaches to object recognition match new images to a database of individual 2D training images, and thereby determine the best matching object without any precise notion of their 3D pose. However, some common applications, such as augmented reality or robotic manipulation, require that recognition also include a precise 3D pose solution. In this chapter, we address the problem of augmented reality, in which synthetic graphics must be superimposed on real images to a high degree of accuracy. Human vision is highly sensitive to misregistration errors, so the accuracy must be sub-pixel and minimize any jitter due to sensor noise.

Our solution is based on using invariant local features to obtain point matches between multiple 2D images of a rigid 3D object or scene. These are then used as input to bundle adjustment to obtain a metrically accurate 3D solution for the locations of the features and cameras. This follows a similar approach to building 3D models from local feature matches that was previously developed

[1] The research in this chapter was first presented at the International Symposium on Mixed and Augmented Reality, 2004 [8].

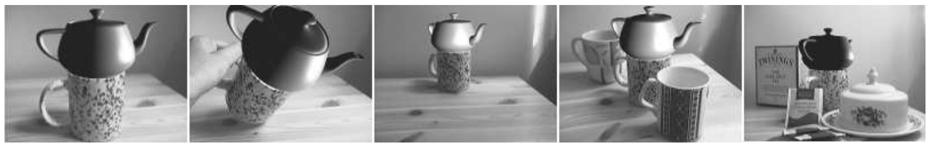

Fig. 1. *The coffee mug is recognized in each frame and its pose computed. The virtual teapot is superimposed to appear on top of the coffee mug. The last two frames demonstrate recognition of the partially occluded mug in cluttered scenes, without tracking from previous frames.*

by Schaffalitzky and Zisserman [20]. In recent work, Rothganger *et al.* [18] have built 3D models from multiple affine-invariant feature correspondences and used these models for recognition. They demonstrate that the 3D models are particularly valuable for recognition as they integrate features from multiple views and are therefore more complete and robust than any single view representation. In this chapter we describe a number of improvements to previous methods that we have found useful, including a simple approach to initializing bundle adjustment, methods for filtering subsets of the most useful features, and a novel approach to jitter reduction in augmented reality. We are able to reliably build models of complex objects and scenes from multiple hand-held images using an uncalibrated camera. The models can then be recognized and tracked in long video sequences while maintaining minimal jitter.

1.1 System overview

Our system operates in two stages. During the first, offline stage, SIFT features are extracted from the reference images and pair-wise correspondences are established. The process remains linear in the number of images by using fast approximate indexing and only linking image pairs forming a spanning tree. These correspondences are used to build a metric model of the real world to be augmented (which could be an individual object or a general scene). At the same time, camera calibration parameters and camera poses corresponding to image viewpoints are computed. Structure and motion recovery is performed with bundle adjustment using a simple initialization procedure.

Once the real world model has been obtained, the position, orientation and size of the virtual object must be specified relative to this model. For this purpose we provide an interactive procedure, which allows the user to determine the pose of the virtual object in the reference images.

The second stage of the system involves recognition and accurate solution of the model pose for live video augmentation. Features detected in the current video frame are matched to those of the world model, and these matches are used to compute the current pose of the model. Jitter is minimized by regularizing the solution using the pose computed for the previous frame. The influence of the previous solution on the current one is weighted without imposing constraints on the overall camera motion. The tracker is very stable in practice (Figure 1 demonstrates some of its capabilities), and it performs online scene recognition and recovery from failure of tracking. Unlike previous systems for augmented reality, our method performs automatic recognition of any of a library of objects using natural features, making it suitable for a variety of mobile applications which involve augmentation of recognized scenes, such as computerized museum tour guides and augmentation of individual objects.

2 Related research

In most previous research on marker-free systems for augmented reality, natural features are used only for establishing correspondences between consecutive frames in a video sequence. Some of the most common choices are the Harris corner detector [9], applied in [3, 4], and the Kanade-Lucas-Tomasi (KLT) tracker [16], used in [23, 7, 19]. To automate the initialization and failure recovery of a tracker, reliable wide baseline matching is desired, which in turn imposes a demand for a higher degree of feature invariance.

A recent approach [5] proposes tracking of parallelogram-shaped and elliptical image regions, extracted in an affinely invariant way, which can be used for scene recognition. Impressive results are presented, but the tracker relies on the presence of planar structures in the viewed scene. In [13] viewpoint invariance is achieved by applying an eigen-image approach to a set of local image patches, which capture the appearance of a real-world point in several views. Their method relies on the pre-built CAD model of the object to be augmented, and requires manual matching of model points to their 2D projections in reference keyframes. In [11] edges of a CAD model are matched to detected image edges. Their visual tracking system is combined with rate gyroscopes in order to handle rapid movements of a head-mounted camera.

Various other techniques have been suggested in augmented reality for acquiring a reference representation of the real world. In [3] two or more reference views are used to compute current camera pose from epipolar geometry constraints on natural feature correspondences. Markers are still used to pre-calibrate the reference frames with standard calibration tools. The initial camera pose must be very close to one of the reference images, due to wide baseline matching limitations. A learning-based strategy is proposed in [7], where the scene is represented by a set of natural features, detected and calibrated during an initial marker-based tracking phase. The system presented in [12] uses fiducial detection to represent the environment and its virtual contents in an affine frame of reference, with an aim to avoid metric camera calibration. This innovative approach achieves comparable results with minimum initialization effort, however it does not allow the modeling of perspective projection effects at close camera distances. In [21] the coordinate frame of the real world is manually inserted into reference views, by specifying image locations of control points. Line intersections on fiducials are tracked to estimate the motion of the camera. Completely markerless and general techniques are presented in [4] and [19], where virtual object registration is achieved based on the results of a global bundle adjustment and self-calibration,

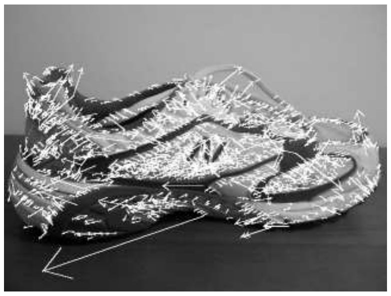

Fig. 2. SIFT keypoints extracted from a 640×480 image of a sneaker. The algorithm found 1533 features shown as white arrows, with size and direction corresponding to feature scale and orientation, respectively.

leading to metric camera motion and scene structure recovery. Both of these methods perform offline batch processing of the entire video sequence, with no support for online scene recognition or tracking.

3 Learning scene geometry

The preliminary stage of the system takes as input an unordered set of images of the real world scene or object to modeled. The images are acquired from unknown, spatially separated viewpoints by a handheld camera, which does not need to be pre-calibrated. At least two snapshots are required; using more allows the capture of more scene features and thus enables a wider-range and more reliable tracking. In our experiments, we have used 5 to 20 images which were gathered from up to a full 360° range of viewpoints, separated by at most about 45°. The scene is assumed to be mostly rigid, with no special markers or known structures present. The system uses these input images to build a sparse 3D model of the viewed scene and to simultaneously recover camera poses and calibration parameters. The virtual object can then be inserted into the modeled environment. The problem is divided into the following steps:

1. Local invariant 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 algorithm for structure and motion recovery.
4. The remaining matches are triangulated using computed camera parameters, and outliers are removed.
5. The position, orientation and size of the virtual object are defined relative to the coordinate frame of the recovered model.

3.1 Feature extraction and matching

We extract SIFT features [14, 15] from each input image for matching. The main attractions of SIFT features are their distinctiveness, invariance, and efficiency, resulting in a high probability of correct matches across a wide range of image variations. In addition, large numbers of these features can be found in a typical image (see Figure 2), making them suitable for recognition and tracking in the presence of occlusions, and generally increasing the robustness of recognition.

The best candidate match for a SIFT feature is its nearest neighbour, defined as the feature with the minimum Euclidean distance between descriptor vectors. The reliability of the nearest neighbour match can be tested by comparing its Euclidean distance to that of the second nearest neighbour from that image. If these distances are too similar, the nearest neighbour match is discarded as unreliable. This simple method works well in practice, since incorrect matches are much more likely to have close neighbours with similar distances than correct ones, due in part to the high dimensionality of the feature space.

The large numbers of features generated from images, as well as the high dimensionality of their descriptors, make an exhaustive search for closest matches extremely inefficient. Therefore we employ an approximate Best-Bin-First (BBF) algorithm, based on a k-d tree search [2]. A k-d tree is constructed from all SIFT features which have been extracted from the reference images. The search examines tree leaves, each containing a feature, in the order of their closest distance from the current query location. Search order is determined with a heap-based priority queue. An approximate answer is returned after examining a predetermined number of nearest leaves. This technique finds the closest match with a high probability, and enables feature matching to run in real time.

For each feature in a reference image, the BBF search finds its nearest and second nearest neighbour pair in each of the remaining images. Putative two-view matches are then selected based on the nearest-to-second-nearest distance ratio (with the threshold value of 0.8). We improve this set of matches by applying an epipolar geometry constraint to remove remaining outliers. For each selected image pair, this constraint can be expressed as

$$\mathbf{x}_i^T F_{ij} \mathbf{x}_j = 0 \tag{1}$$

where $\mathbf{x}_i = [u_i \; v_i \; 1]^T$ and $\mathbf{x}_j = [u_j \; v_j \; 1]^T$ are homogeneous image coordinates of the matched features in images i and j, respectively, and F_{ij} is a fundamental matrix of rank 2. The computation of F between each pair of N images has $\binom{N}{2}$ complexity, thus quickly becoming prohibitively expensive with increasing N. Therefore we apply a selective approach, similar to [20], which is linear in the number of images. Image pairs are selected based on a greedy algorithm, which constructs a spanning tree on the image set. Starting with the two images that have the most putative matches, we compute F consistent with the majority of matches using the RANSAC algorithm [6], discard outliers and join these images with an edge. This process is repeated for the image pair with the next highest number of matches, subject to the constraint that joining these images does not

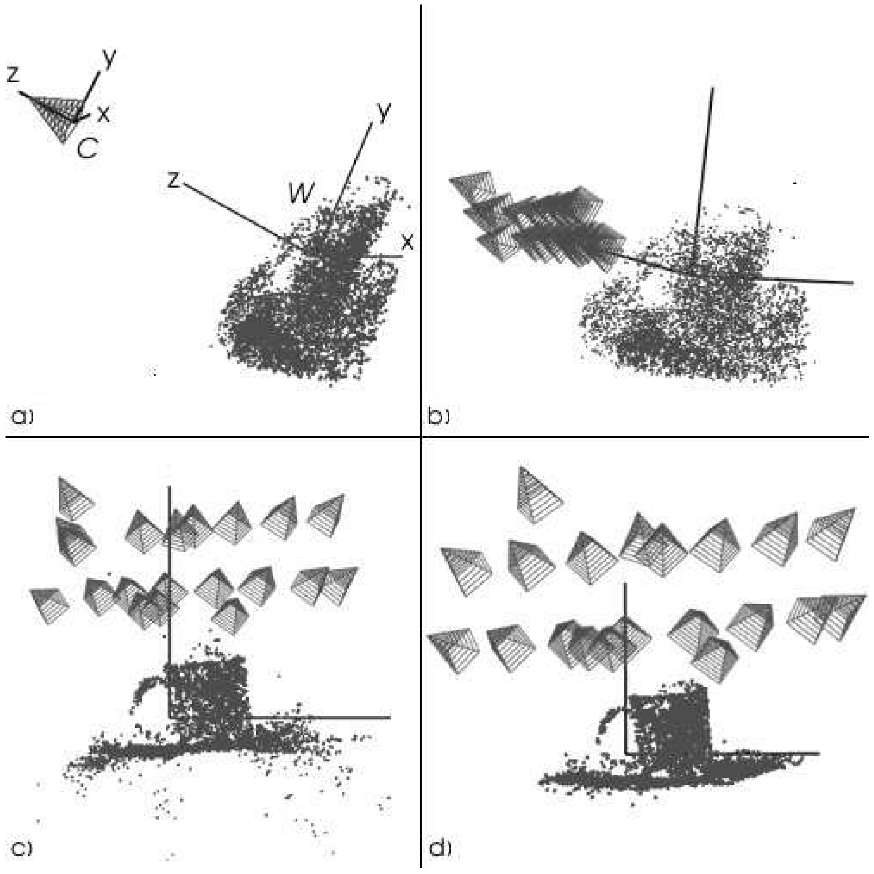

Fig. 3. *Building a model of a coffee mug placed on top of a magazine from 20 reference images. Cameras are shown as wire cones and image features as points: (a) initialization places all cameras at the same location and all points at the same distance from the cameras (average reprojection error = 62.5 pixels); (b) results after 10 iterations (error = 4.2 pixels); (c) results after 20 iterations (error = 1.7 pixels); (d) final results after 50 iterations (error = 0.2 pixels).*

create a cycle. In this manner, the expensive cleanup operation is applied only to the more promising candidates.

The entire image set is considered processed when the addition of any remaining candidate image pair would create a cycle in the tree. At this point we establish multi-view 2D point correspondences by traversing the tree and stitching together two-view feature matches. Because the tree structure is free of cycles, the generation of multi-view matches is straightforward and unambiguous.

3.2 Motion and structure recovery

Once the multi-view matches have been established, we seek to compute world coordinates of the corresponding 3D points, calibration parameters and camera poses for each reference view. Formally, a 2D projection $\mathbf{x}_{ij} = [u_{ij}\ v_{ij}\ 1]^T$ of a 3D point $\mathbf{X}_j = [x_j\ y_j\ z_j\ 1]^T$ in an image i is expressed as

$$\mathbf{x}_{ij} \sim P_i \mathbf{X}_j \tag{2}$$

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

$$P_i = K[R_i\ \mathbf{t}_i] \tag{3}$$

In the above equation, matrix K contains camera calibration parameters, such as focal length, aspect ratio and principal point coordinates; R_i and $\mathbf{t}_i$ are the rotation and translation of the world frame relative to the camera frame for image i.

A classical approach to this problem begins with an algebraic initialization of projective structure and motion, using two- or three-view epipolar constraints. This is followed by an upgrade to a metric framework with self-calibration techniques, as well as a solution refinement via an iterative bundle adjustment optimization [10]. We employ an alternative technique suggested by Szeliski and Kang [22], which omits the linear initialization step and solves for all of the unknown parameters iteratively, using a general-purpose optimization algorithm, such as Levenberg-Marquardt [17]. The problem is formulated as the minimization of the reprojection errors over all camera parameters and world point coordinates, given image projections of the world points:

$$\min_{\mathbf{a}_{ij}} \sum_i \sum_j \| w_j (\mathit{\Pi}(\mathbf{a}_{ij}) - \mathbf{x}_{ij}) \|^2 \tag{4}$$

where $\mathit{\Pi}$ is the non-linear projection function and the vector $\mathbf{a}_{ij} = [\mathbf{X}_j^T\ \mathbf{p}_i^T\ \mathbf{c}^T]^T$ contains the unknown parameters: 3D coordinates $\mathbf{X}_j$ of a world point j, camera pose parameters $\mathbf{p}_i$ for an image i, and global calibration parameters $\mathbf{c}$ (or $\mathbf{c}_i$, in case of varying calibration parameters). After 15 iterations to establish an initial solution estimate, the confidence weight w_j associated with $\mathbf{X}_j$ is lowered for world points with high reprojection errors using the Huber norm, thus reducing the contribution of outliers to the final solution.

To initialize the algorithm, we back-project the 2D points from an arbitrary view to an xy-plane of the world frame, place all cameras at the same default distance along the z-axis directly facing the plane, and use default values for the calibration parameters. It is possible that bundle adjustment will converge to a false local minimum due to depth reversal (as illustrated in the Necker cube illusion). As suggested by [22], this can be avoided by reflecting the depth of the first model solution about the xy-plane, restarting the bundle adjustment, and selecting the solution with the best final reprojection error. This simple initialization allows us to achieve proper convergence with the cameras as far as

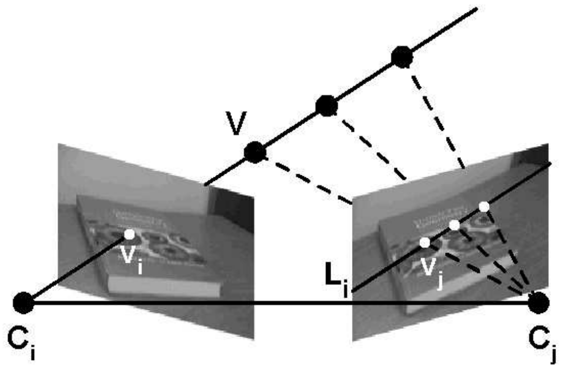

Fig. 4. The placement of the virtual frame origin **V** in 3D is achieved by anchoring its projection $\mathbf{v_i}$ in image **i** and adjusting its projection $\mathbf{v_j}$ in image **j** along the epipolar line $\mathbf{L_i}$.

90° apart, in a few dozen iterations. Figure 3 shows the sequence of convergence for even a large set of 20 images of a typical scene, although in practice, for efficiency, we only start with 5 images and then add others incrementally.

To reduce problem size, as an input to the Levenberg-Marquardt algorithm we select a limited number (at most 100) of the points with the most correspondences. Coordinates of the remaining points can be easily computed using standard triangulation techniques [10], once the camera parameters have been recovered. Lastly, we remove any model point outliers with large reprojection errors. The latter are usually a result of infrequent feature mismatches which have survived the epipolar constraint test.

3.3 Virtual object placement

For augmented reality, the insertion of the 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 established via triangulation, 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 (see Figure 4).

Next, the user is able to fine-tune the position, orientation and size of the virtual object in variable-size increments. Figure 5 shows an example of the virtual frame insertion and pose adjustment. The virtual object is rendered onto the reference images using previously recovered camera parameters. At any time the user can switch between the images to view the corresponding projection

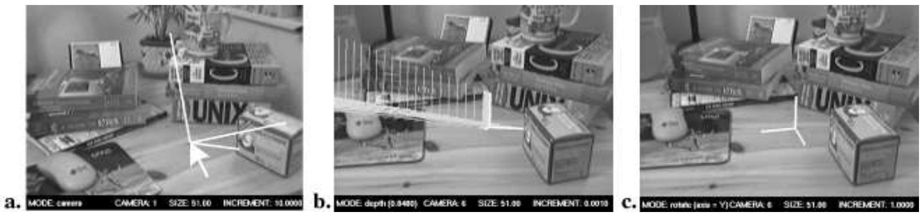
a.
MODE: camera CAMERA: 1 SIZE: 51.00 INCREMENT: 10.0000
b.
MODE: depth (0.0400) CAMERA: 6 SIZE: 51.00 INCREMENT: 0.0010
c.
MODE: rotate (axis = Y) CAMERA: 6 SIZE: 51.00 INCREMENT: 1.0000

Fig. 5. Insertion of the virtual frame 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 the epipolar line in another image; c) subsequent orientation adjustment.

of the virtual contents. Note that the geometric relationships between the real world, its virtual contents 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. If a metric object scale is required, this parameter can be provided by user input of a single known dimension or by presence of a calibrated object in one of the views.

4 Model recognition and camera tracking

The online computations of the system are summarized in the following steps:

1. SIFT features are extracted from the current frame of the video sequence.
2. The new features are matched to the image features of the world model using the BBF algorithm, resulting in a set of 2D-to-3D correspondences.
3. The correspondences are used to compute the current camera pose, via a robust approach which combines RANSAC and Levenberg-Marquardt algorithms.

To initialize the tracker, a k-d tree is constructed from the image features of the world model. Each image feature is a 2D projection with links to its 3D world coordinates, a reference image in which it was found and the corresponding recovered camera pose. During tracking, this structure is used to efficiently detect model point projections in each new frame. A nearest and a second nearest neighbour pair is found for each feature from the current frame via a BBF search, with the two neighbours belonging to different model points. As in Section 3.1, 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 the number of reliable best matches falls below a predefined threshold (set to 15 in our experiments, although a much lower threshold could be used with more careful verification). This occurs when all or most of the model disappears out of sight, or the frame contains too much motion blur. In such cases the rendering of virtual contents is postponed until enough model points are detected.

Given a set of putative 2D-to-3D matches $(\mathbf{x}_{tj}, \mathbf{X}_j)$ for the frame t, we can compute the corresponding camera pose parameters by minimizing the residual sum:

$$\min_{\mathbf{p}_t} \sum_j \|w_{tj}(\Pi(\mathbf{a}_{tj}) - \mathbf{x}_{tj})\|^2 \tag{5}$$

where the weight w_{tj} describes the confidence in the measurement $\mathbf{x}_{tj}$ and is set to the reciprocal of its estimated standard deviation in the image. Since SIFT features with larger scales are computed from more blurred versions of the image, they have lower location accuracy. Therefore, we set w_{tj} inversely proportional to the scale of each feature. This time the camera pose parameters $\mathbf{p}_t$ are the only unknowns in the vector $\mathbf{a}_{tj}$ (assuming unchanging calibration parameters). We initialize $\mathbf{p}_t$ to $\mathbf{p}_{t-1}$, computed for the previous frame. For the first frame of the video sequence or the one immediately after tracking failure, as an initial guess we use the camera pose of the reference image contributing the most 2D feature matches from the BBF search.

We apply RANSAC to compute the camera pose consistent with the most matches. The minimization given by (5) is performed for each RANSAC sample, and the final solution is computed using all of the inliers as input. Despite its iterative nature, this approach has proven to be sufficiently fast for online use. The small number of unknown parameters results in a rapid execution of Levenberg-Marquardt iterations. Very few RANSAC samples are needed, since the non-linear computation of 6 elements of $\mathbf{p}_t$, corresponding to the 6 degrees-of-freedom of the camera pose, requires the minimum of only 3 matches. Furthermore, the input set of matches usually contains a very small fraction of outliers due to the fact that ambiguous matches have already been removed by the distance ratio check.

4.1 Jitter reduction

The solution to (5) 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 feature matches. In addition, the surface of the error function may be flat near its minimum, as it may be difficult to distinguish between slight changes in rotation and translation parameters for near-planar objects.

To stabilize the solution, we modify (5) by adding a regularization term which favours minimum camera motion between consecutive video frames:

$$\min_{\mathbf{p}_t} \sum_j \|w_{tj}(\Pi(\mathbf{a}_{tj}) - \mathbf{x}_{tj})\|^2 + \alpha^2 \|W(\mathbf{p}_t - \mathbf{p}_{t-1})\|^2 \tag{6}$$

where W is a 6×6 diagonal matrix of prior weights on the camera pose parameters, and α is a scalar which controls the tradeoff between the current

feature extraction (SIFT algorithm)	150 ms
feature matching (BBF algorithm)	40 ms
camera pose computation	25 ms
frames per second	4

Fig. 6. Average computation times for a video sequence with 640 × 480 frame size. The real world model contains about 5000 3D points.

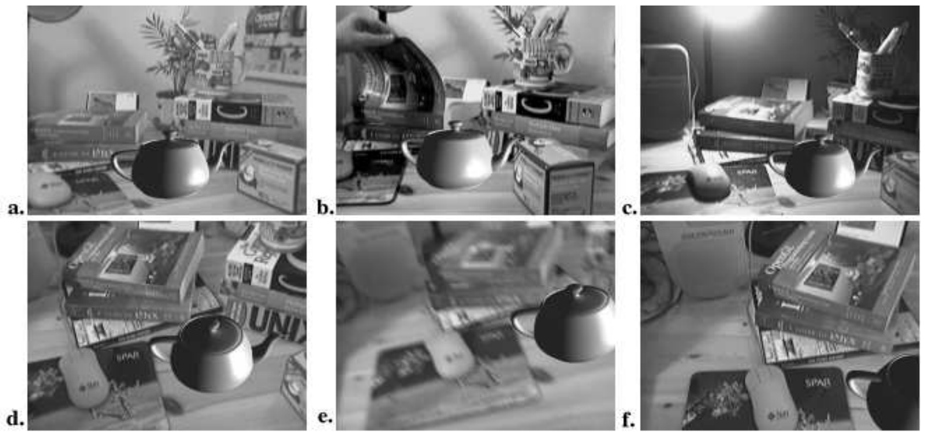

Fig. 7. Examples of tracking within a complex scene: a) a virtual teapot is placed in the modeled desk scene; b) the scene does not have to be fully static; c,d) recognition is robust to changes in lighting and viewpoint; e) moderate amounts of motion blur are acceptable; f) a partial view of the scene is correctly recognized.

measurements and the stable solution. Each diagonal entry of W is set to the inverse of the standard deviation of the corresponding parameter, reflecting the relative range of expected frame-to-frame change in the camera pose (e.g., a few degrees for a change in rotation).

Instead of adopting the usual approach of setting α to a constant value, we adjust it separately for each video frame. We would like high levels of smoothing for slow motions while avoiding over-smoothing of large camera motions 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 required to be no higher than that of the image feature noise. This can be expressed by the inequality

$$\alpha^2 \|W(\overline{\mathbf{p}}_t - \mathbf{p}_{t-1})\|^2 \leq \sigma^2 N \tag{7}$$

where N is the number of matching image points, $\overline{\mathbf{p}}_t$ is the final new camera solution, and σ is the estimated uncertainty of an image measurement (e.g., 0.5

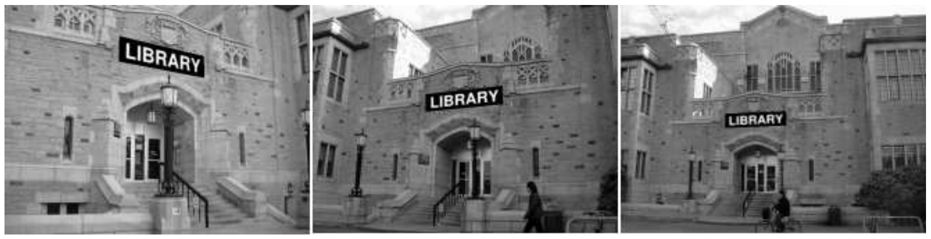

Fig. 8. The augmentation of the entrance to the university library with a new sign.

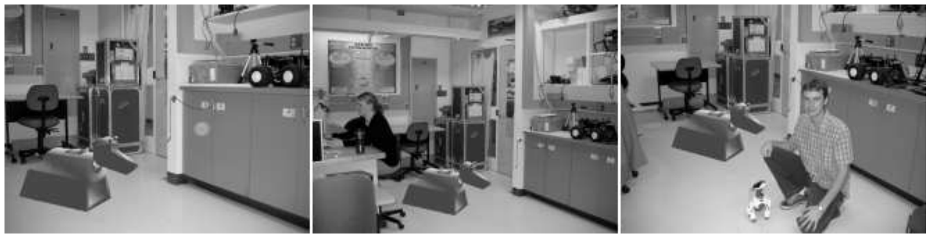

Fig. 9. A virtual robotic dog in the modeled corner of the lab room. Successful results were achieved with people freely moving around the room.

pixels). It follows that the maximum allowable amount of smoothing is

$$\alpha^2 = \frac{\sigma^2 N}{\|W(\overline{\mathbf{p}}_t - \mathbf{p}_{t-1})\|^2} \tag{8}$$

Because $\overline{\mathbf{p}}_t$ 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 $\alpha = 0$. 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 (6) with α recomputed at each iteration as per (8), using the most recent estimate of $\mathbf{p}_t$ to approximate $\overline{\mathbf{p}}_t$. The search for a regularized solution terminates when $\mathbf{p}_t - \mathbf{p}_{t-1}$ becomes very small (which would occur for a camera that appears stationary within measurement noise) or no longer changes significantly.

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. This method has worked very well in practice to almost eliminate perceived jitter, and experiments described below show that it leads to a large reduction in measured jitter (Figure 11).

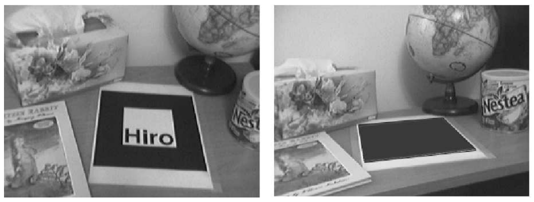

Fig. 10. ARToolkit marker in the scene (left). Virtual square, superimposed onto the marker during tracking (right).

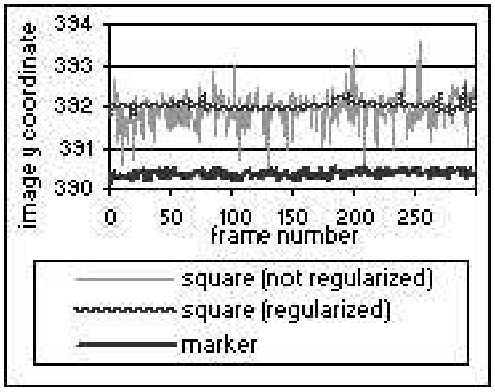

Fig. 11. Stationary camera results for 300 frames. Jitter of the virtual square is significantly reduced by camera pose regularization.

5 Experiments

The system prototype has been implemented in C using OpenGL and GLUT libraries, on an IBM ThinkPad with a Pentium 4-M processor (1.8 GHz) and a Logitech QuickCam Pro 4000 video camera. An example of current computation times for the tracker is given in Figure 6. Current speed of recognition and tracking is about 4 frames/sec.

To demonstrate the capabilities of the system, we have tested its performance on a variety of scenes and tracking scenarios. Some of the augmented video frames are shown in Figures 7 through 9. Video examples are available on the authors' web pages.

In order to test the accuracy of registration, we aligned a virtual square with an ARToolKit marker [1], which was present in a modeled scene (Figure 10).

While tracking the scene, the corners of the marker were detected using the ARToolKit library, and their image coordinates were used as the ground truth for the registration of the virtual square. Figures 11 and 12 compare the results for one of the corners.

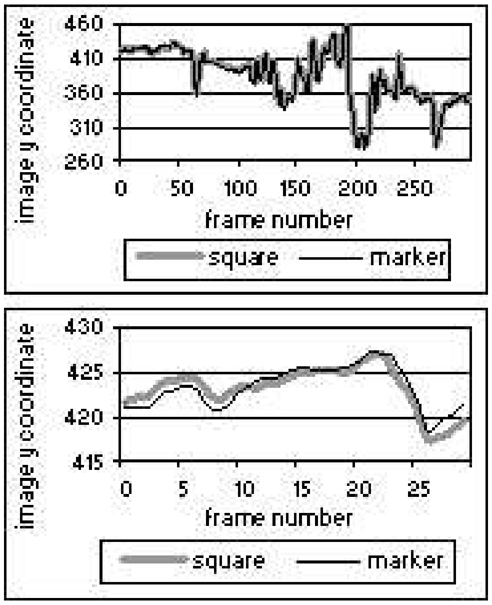

Fig. 12. Moving camera results for 300 frames (top) and the first 30 frames (bottom). The trajectories of the real and virtual corners are in close correspondence, with varying camera motion handled without noticeable drift.

6 Conclusions and future work

In this chapter we presented an approach to augmented reality that performs registration of virtual objects into a live video sequence using local image features. The system consists of two parts. The offline part involves recovery of metric scene structure and camera parameters from a set of reference images. The online part performs camera pose tracking and virtual object registration using models resulting from the offline processing. Some of the novel aspects of this work include a simple approach to initializing bundle adjustment, an efficient incremental method for 3D structure recovery that starts with subsets of images and features, and a successful method for jitter reduction.

Our system has been able to achieve successful modeling and recognition of scenes of varying size and complexity, from handheld objects to buildings (Figures 7 through 9). The next step in performance testing will focus on the system scalability for operation in large environments, such as a campus or a museum. A potential enhancement involves modeling many buildings, rooms or objects, and providing database management to switch between models as the user travels around his or her surroundings.

Acknowledgements

We would like to gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Institute for Robotics and Intelligent Systems (IRIS).

References

1. ARToolKit: http://www.hitl.washington.edu/artoolkit/.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. Iryna Gordon and David G. Lowe, Scene modeling, recognition and tracking with invariant image features. *International Symposium on Mixed and Augmented Reality (ISMAR),* Arlington, VA, pages 110–119, 2004.
9. C.J. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.
10. R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
11. 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.
12. Kiriakos N. Kutulakos and James R. Vallino. Calibration-free augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):1–20, 1998.
13. 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.
14. David G. Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, pages 1150–1157, 1999.
15. David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision,* 60(2): 91-110, 2004.
16. B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

17. 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.
18. Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3D Object Modeling and Recognition Using Affine-Invariant Patches and Multi-View Spatial Constraints. *IEEE Conference on Computer Vision and Pattern Recognition,* Madison, WI, pages 272-277, 2003.
19. 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.
20. 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.
21. Yongduek Seo and Ki Sang Hong. Calibration-free augmented reality in perspective. *IEEE Transactions on Visualization and Computer Graphics*, 6(4), pages 346–359, 2000.
22. Richard Szeliski and Sing Bing Kang. Recovering 3D shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation,* 5(1), pages 10–28, 1994
23. 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.