Scene Modelling, Recognition and Tracking with Invariant Image Features

Iryna Gordon and David G. Lowe
University of British Columbia
Department of Computer Science
Vancouver, BC, Canada
{skrypnyk, lowe}@cs.ubc.ca

Abstract

We present a complete system architecture for fully automated markerless augmented reality (AR). The system constructs a sparse metric model of the real-world environment, provides interactive means for specifying the pose of a virtual object, and performs model-based camera tracking with visually pleasing augmentation results. Our approach does not require camera pre-calibration, prior knowledge of scene geometry, manual initialization of the tracker or placement of special markers. Robust tracking in the presence of occlusions and scene changes is achieved by using highly distinctive natural features to establish image correspondences.

1. Introduction

Until recently, accurate registration of virtual content has imposed substantial initialization and setup requirements on vision-based camera tracking in AR. Many successful systems have been developed [12, 1, 11, 24], which rely on strategically placing fiducial markers in the environment and performing manual camera calibration procedures. In contrast, our work is motivated by the need for more flexible and automated AR systems, capable of operating in unprepared settings and generally requiring less time and effort to set up. We present an approach which meets this goal, and at the same time is capable of robust 6 degree-of-freedom camera pose tracking, as well as convincing geometric consistency of augmentation. The only required input is a set of reference images, taken by a handheld uncalibrated camera from arbitrary viewpoints.

As an alternative to markers, we propose the use of stable natural features to aid the tracker. Generated from an image via the Scale Invariant Feature Transform (SIFT) algorithm [16], 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 can be 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 the reference images and multi-view correspondences are established. 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, calibration parameters and camera poses corresponding to image viewpoints are computed. Structure and motion recovery is performed with a simple yet powerful technique, based on non-linear least squares, which is able to construct models of arbitrary geometric complexity.

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 application, which allows the user to determine the appearance of the virtual object in the reference images.

The second stage of the system involves model recognition and computation of the current camera 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 camera. Jitter is minimized 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 without imposing constraints on the overall camera motion. The tracker is remarkably stable (Figure 1 demonstrates some of its capabilities), and can perform online scene recognition and recovery from failure, with the tracked scene going in and out of view. Automatic recognition is one of the strongest advantages of our system, making it suitable for a variety of mobile applications which involve augmentation of recognized scenes, such as computerized museum tour guides and site navigators.

The remainder of this paper is organized as follows. An

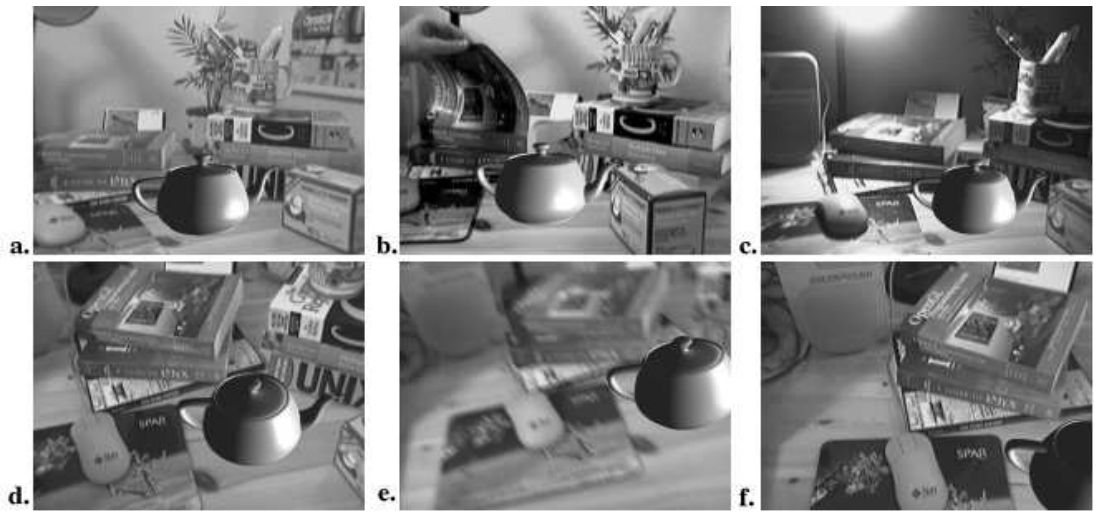

Figure 1. Typical performance of the tracker: a) a virtual teapot is placed in the modelled desk scene; b) the scene does not have to be fully static; c) changes in lighting are properly handled; d) so are drastic changes in viewpoint; e) moderate amounts of motion blur are acceptable; f) partial view of the scene is correctly recognized.

overview of related work is provided in Section 2. Section 3 discusses our approach to scene modelling and virtual object insertion. The details of camera pose tracking are presented in Section 4. System performance is discussed in Section 5, followed by final remarks in Section 6.

2. Related research

In most marker-free AR systems, natural features are used for establishing correspondences between consecutive frames in a video sequence, i.e., for narrow baseline matching. Some of the most common choices are the Harris corner detector [9], applied in [4, 5], and the Kanade-Lucas-Tomasi (KLT) tracker [18], used in [25, 8, 20]. 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. This problem is closely related to the task of object recognition in computer vision, yet to our knowledge little effort has been made to introduce it to AR.

A recent approach [6] 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, however the tracker lacks generality, as it relies on the presence of planar structures in the viewed scene. In [15] viewpoint invariance is achieved by applying eigen image methodology 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 [13] 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.

Although user-supplied CAD models have been shown to be effective tools for camera tracking, they are not always readily available, and their use is limited to objects that can be easily modelled by hand. In contrast, our method primarily targets arbitrary shapes and textures, often present in natural environments and man-made settings. Moreover, no additional commercial tools or equipment, other than an off-the-shelf video camera, is required.

Various techniques have been suggested in AR for acquiring a reference representation of the real world. In [4] 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 [8], 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 [14] 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 modelling of perspective projection effects at close camera distances. In [22] 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 [5] and [20], where virtual object registration is achieved based on the results of a global bundle adjustment and self-calibration, 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 an input an unordered set of images of the real world scene or object to be augmented. 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 front, side and top 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 modelled environment. The problem is divided into the following steps:

1. Local 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 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

The main attractions of SIFT features are their distinctiveness and invariance, resulting in a high probability of correct matches across a wide range of image variations. In addition, many of these features densely populating a typical image can be efficiently extracted (see Figure 2), making them suitable for recognition and tracking in the presence of occlusions, and generally for algorithms benefiting from a large number of feature matches. In this section we give a brief overview of the feature extraction algorithm (see [16, 17] for more details), followed by the discussion of the multi-view feature matching approach.

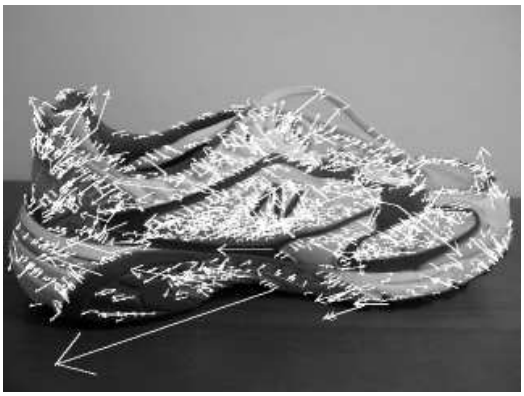

Figure 2. SIFT keypoints extracted from a 640×480 image of a sneaker. The algorithm found 1533 features shown as white arrows, with size and direction corresponding to feature scale and orientation, respectively.

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 variable-scale Gaussians, after which adjacent Gaussian images are subtracted to produce the DOG images. Each point in the latter is then 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. Besides image location and scale at which it was found, each stable feature is assigned an image orientation and a feature descriptor vector, which reflect local image properties. Feature orientation and descriptor vector are computed from gradient magnitudes and orientations, sampled within a circular Gaussian-weighted window around the feature. 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, however smaller values are acceptable. Fea-

ture descriptors are represented in a scale- and orientation-invariant manner, and are normalized to reduce the effects of illumination changes.

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 the one of the second nearest neighbour. If these distances are too similar, the nearest neighbour match is discarded as an outlier. 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 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 [3]. A k-d tree is constructed from all SIFT features which have been extracted from the reference images. The search examines bins, or 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, and an approximate answer is returned after examining a predetermined number of nearest leaves. This technique finds a 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 the remaining outliers. For each 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 [21], 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 [7], 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 create a cycle. In this manner, the expensive cleanup operation is applied only to the more promising candidates: images from less separated viewpoints generally lead to more putative matches, most of which are likely to be correct.

The entire image set is considered processed when an addition of any remaining 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 in [23], 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 [19]. 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:

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

where Π 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). The confidence weight w_j associated with $\mathbf{X}_j$ is lowered for world points

Figure 3. One of 20 reference images of a coffee mug on top of a book (left) and the recovered model (right), with cameras shown as wire cones. The outermost cameras were placed about 60° apart. The model correctly captures the planarity of the book surface and the roundness of the mug.

with high reprojection errors after a predefined number of iterations, thus reducing the contribution of likely outliers to the final solution.

To initialize the algorithm, we back-project the 2D points 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. This simple initialization allows us to achieve proper convergence with the cameras as far as 90° apart, in a few dozen iterations. As shown in [23], full 360° models can be recovered by starting with a subset of nearby cameras and incrementally adding more cameras and point matches to the intermediate solution. Occasional depth reversals are remedied by reflecting the depth of the model about the xy-plane, and selecting the solution with the best final reprojection error.

Besides its simplicity, this approach is attractive for several reasons. It produces a statistically optimal estimate within a broad region of convergence, it robustly handles noisy measurements and incomplete multi-view correspondences, and it is flexible in the number of parameters to be computed. It can converge to a realistic estimate of camera parameters, including scenarios with a varying focal length. In addition, scenes with arbitrary geometric shapes (including planes) are correctly reconstructed (see Figure 3).

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 and unlikely 3D coordinates. The latter are usually a result of infrequent feature mismatches which have survived the epipolar constraint test. They either appear behind the line of sight of all cameras, or have an unusually large depth, noticeably deviating from the cluster of inliers.

3.3. Virtual object placement

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 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 (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 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.

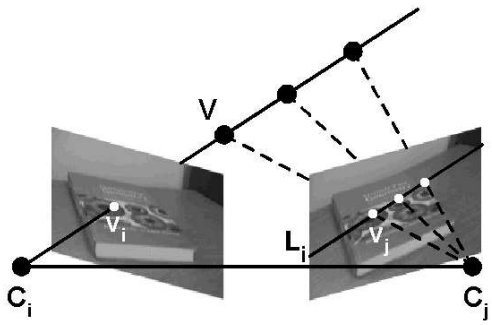

Figure 4. The placement of the virtual frame origin $\mathbf{V}$ in 3D is achieved by anchoring its projection $\mathbf{v_i}$ in image $\mathbf{i}$ and adjusting its projection $\mathbf{v_j}$ in image $\mathbf{j}$ along the epipolar line $\mathbf{L_i}$.

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 points' 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). 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:

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

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. 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, most of which 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 a local minimum, as it may be difficult to distinguish between slight changes in rotation and translation parameters.

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

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

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 measurements and the desired estimate. Each diagonal entry of W is set to the inverse of the experimentally estimated standard deviation of the corresponding parameter. Instead of setting α to a constant value, we attempt to adjust it separately for each video

Figure 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.

frame, in order to prevent oversmoothing of camera motion (which would result in a virtual object "drifting" behind a faster moving scene). The adjustment of α is performed as follows. At first, we solve for $\mathbf{p}_t$ using (5), i.e. with $\alpha = 0$. Once a local minimum has been reached, we explore its immediate neighbourhood, searching for a regularized solution. This is done by executing a few additional Levenberg-Marquardt iterations, this time solving (6) with gradually increasing values of α. For each additional iteration, α is computed as

$$\alpha = \sqrt{\frac{MN}{\|W(\mathbf{p}_t - \mathbf{p}_{t-1})\|^2}} \qquad (7)$$

where N is the number of 2D-to-3D matches and M is an estimated error of an image measurement (set to a fraction of a pixel). We stop when the next iteration would result in a total reprojection error that deviates too much from the error yielded by the original solution of (5). In essence, as much smoothing as possible is applied while still agreeing with the measured data. 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.

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. More work needs to be done to optimize the tracker in order to achieve real-time performance.

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 10 through 13. Video examples are available at http://www.cs.ubc.ca/∼skrypnyk/arproject/.

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

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

In order to test the accuracy of registration, we aligned a virtual square with an ARToolKit marker [2], which was present in a modelled scene (Figure 7).

Figure 7. ARToolkit marker in the scene (left). Virtual square, superimposed onto the marker during tracking (right).

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 8 and 9 compare the results for one of the corners.

6. Conclusions and future work

In this paper we presented a versatile approach to AR, which performs registration of virtual objects into a live

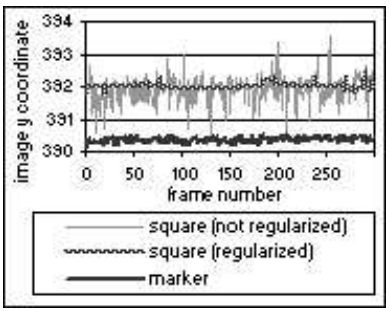

Figure 8. Stationary camera results for 300 frames. Jitter of the virtual square is significantly reduced by camera pose regularization.

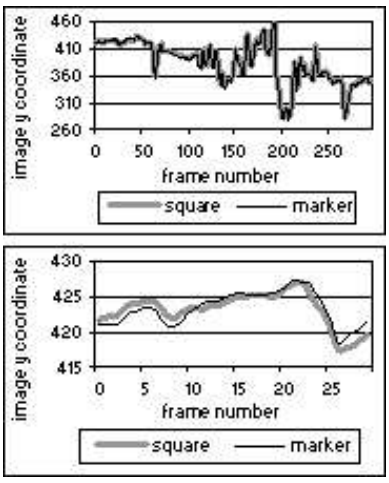

Figure 9. Moving camera results for 300 frames (top) and the first 30 frames (bottom). The trajectories of the real and virtual corners are in close correspondence, with varying camera motion handled without noticeable drift.

video sequence using stable 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, via purely passive computer vision techniques. The online part performs camera pose tracking and virtual object registration using results of the offline processing. No manual initialization of the tracker is required. The system can handle scenes with moving elements, occlusions and illumination changes. No constraints are imposed on the camera motion, or the type of the operational environment, other than the presence of some texture for feature extraction.

Relying on high contrast in scene images may impose certain restrictions on the applicability of this method. Examples shown in this paper yield from a few hundred to a few thousand SIFT features per 640×480 image, which is more than adequate for reliable matching. Several modifications and improvements can be made to achieve good performance when dealing with regions of lower contrast, such as lowering the DOG threshold value for keypoint extraction and combining SIFT features with edge-based image descriptors.

It should be noted that our approach is unable to handle proper occlusion of inserted virtual content by real objects in the world. To achieve this effect, a full model of the observed scene is required. The construction of complete models from images is an important subject of research in computer vision, however it is beyond the scope of this project.

Currently, the implementation of the tracker runs at 4-5 fps on average, which is too slow for real-time operation (20-30 fps). The main bottleneck in online processing is acquisition of video frames and feature extraction. Future research efforts will include overall system optimization, involving both hardware and software improvements.

Our system has been able to achieve successful modelling and recognition of scenes of varying size and complexity, from handheld objects to buildings (Figures 10 through 13). 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 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.

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] M. Appel and N. Navab. Registration of technical drawings and calibrated images for industrial augmented reality. *Machine Vision and Applications*, 13(3):111–118, July 2002.

[2] http://www.hitl.washington.edu/artoolkit/.

[3] J. S. Beis and D. 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.

[4] K. W. Chia, A. D. Cheok, and S. J. 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.

[5] K. Cornelis, M. Pollefeys, M. Vergauwen, and L. Van Gool. Augmented reality using uncalibrated video sequences. *Lecture Notes in Computer Science*, 2018:144–160, 2001.

[6] V. Ferrari, T. Tuytelaars, and L. Van Gool. Markerless augmented reality with a real-time affi ne region tracker. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*, pages 87–96, 2001.

[7] M. Fischler and R. Bolles. RANdom SAmple Consensus: a paradigm for model fi tting with application to image analysis and automated cartography. *Communications of the Association for Computing Machinery*, 24(6):381–395, 1981.

[8] 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.

[9] C. 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] W. A. Hoff, K. Nguyen, and T. Lyon. Computer vision-based registration techniques for augmented reality. In *Proceedings of Intelligent Robots and Computer Vision XV, SPIE*, pages 538–548, 1996.

[12] H. Kato and M. 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.

[13] G. Klein and T. 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.

[14] K. N. Kutulakos and J. R. Vallino. Calibration-free augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):1–20, 1998.

[15] V. Lepetit, L. Vacchetti, D. Thalmann, and P. 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.

[16] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision*, pages 1150–1157, 1999.

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. Accepted for publication in the *International Journal of Computer Vision*, 2004.

[18] B. 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.

[19] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[20] H. S. Sawhney, Y. Guo, J. Asmuth, and R. 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.

[21] 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.

[22] Y. Seo and K. S. Hong. Calibration-free augmented reality in perspective. *IEEE Transactions on Visualization and Computer Graphics*, 6(4):346–359, 2000.

[23] R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using non-linear least squares. Technical report, Cambridge Research Laboratory, 1993.

[24] M. Tuceryan, D. S. Greer, R. T. Whitaker, D. E. Breen, C. Crampton, E. Rose, and K. 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.

[25] A. Yao and A. 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.

Figure 10. The augmentation of the entrance to the university library with a 2D textual annotation.

Figure 11. A virtual teapot on a spotted coffee mug. The last two frames demonstrate successful recognition in cluttered scenes.

Figure 12. The placement of a 3D object (the cube) on top of a 2D surface (the book cover). The book is partially occluded in the last two frames.

Figure 13. A virtual robotic dog in the modelled corner of the lab room. Successful results were achieved with people freely moving around the room.