Adaptive Tracking and Model Registration Across Distinct Aspects*

S. Ravela B. Draper J. Lim R. Weiss
Computer Vision Research Laboratory
University of Massachusetts, Amherst, MA 01002

Abstract

A model registration system capable of tracking an object through distinct aspects in real-time is presented. The system integrates tracking, pose determination, and aspect graph indexing. The tracking combines steerable filters with normalized cross-correlation, compensates for rotation in 2D and is adaptive. Robust statistical methods are used in the pose estimation to detect and remove mismatches. The aspect graph is used to determine when features will disappear or become difficult to track and to predict when and where new features will become trackable. The overall system is stable and is amenable to real-time performance.

1 Introduction

Maintaining object registration over time (*temporal registration*) can be defined as the ability to retain up-to-date object-sensor pose relationships over relative motion. Registration is useful in several domains; as an example consider *enhanced reality* applications such as an interactive repair manual. In this application technicians look through a visor at a correctly annotated object; together, the object's image and the overlaid annotations unambiguously provide directions to the next repair step. Given that there is relative camera-object motion (technicians can move), spatially accurate annotations can be overlayed only when the object is temporally registered.

Temporal registration can be achieved using two basic approaches. One relatively expensive yet proven technology is to instrument the real world with location beacons and position sensors. The other is to visually track modelled object features and use pose estimation to update the object-camera transform. This approach is cheaper, can be used in unmodified environments and permits annotation of independently moving objects[1].

In this paper a system for temporal registration of a modelled object using a single camera is developed and it is claimed that robust, real-time registration is possible through 360° of out-of-plane object rotation.

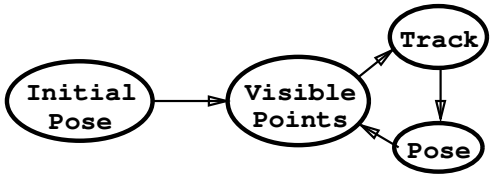


Figure 1: The interaction between components of the registration system

Our system is initialized with a known set of model-image correspondences, known camera parameters and a pre-compiled aspect table that associates discrete viewpoints[2] with object features visible from those views.[3]. Once initialized, the system follows the simple three step loop illustrated in figure 1. Initial correspondences are used to estimate an initial pose. Pose information is used to index into the aspect table and a list of visible features is extracted. Pose is used once again in conjunction with camera parameters to hypothesize image plane locations of feature templates. These templates are tracked and a new pose is computed. The cycle repeats.

System components (namely, view indexing, tracking and pose) individually and by virtue of their interaction contribute to the speed, stability and robustness of the system. As the object undergoes relative out-of-plane rotation in the camera, new features appear and old ones disappear. By construction, only points that are visible from a particular view are tracked and used to compute pose, resulting in continuity of registration across viewpoints. The tracker localizes feature templates in search windows around hypothesized locations using steerable filters and normalized cross-correlation. This technique is relatively insensitive to changes in lighting conditions and compensates fully for feature translations and rotations in the image-plane; it also compensates for some non-trivial out of plane rotations. So long as the actual feature is within the search window, un-occluded and free from specular reflections in the image, the tracker locates features correctly. Pose computation [14] uses camera parameters and the model-image correspondences to robustly solve for a rotation and translation that minimizes the

*The authors received support from ARPA and TACOM under contract DAAE07-91-C-R035 and NSF under grants IRI-9208920 and IRI-9116297.

[1] Automatic positioning systems may do this if each object has it's own beacon(s)

[2] In this paper it is assumed that the camera will roughly point towards the object throughout the relative motion

[3] An object feature is defined by two pieces of information; A model coordinate on the object and a template that captures the feature's appearance in the camera.

projection error of model points on the object. The robustness in this iterative algorithm is obtained by using a median-filter to detect and exclude outliers from the final pose. So long as a minimum of 4 non-coplanar points track correctly pose can be reliably computed.

One important result of the interaction of the pose and tracking components is system stability. The tracker compensates for feature motions, thereby suppressing errors in computed pose. Similarly, the median filter based pose computation is designed to suppress tracking errors. Neither tracking nor pose errors are fed back, thus making the system stable. A second important result of combining pose and tracking components is that tracking is adaptive. Feature templates are updated during registration without any slip from their intended features. Finally, real-time performance is possible on current hardware, within reasonable limits. For $11x11$ size search windows and $9x9$ templates, the speed of the tracker for 6 points is 8 Hz. If a maximum of one tracking outlier per frame is detected, the system can produce registration data at 7 Hz. Stability, adaptivity and speed are discussed in detail in section 4.

2 Related Work

Temporal registration has been addressed by several researchers [5, 10, 15, 18, 19, 20]. Dickinson et. al. [5] use an aspect prediction graph together with a network of active contours introduced in [13]. Active contours are purely gradient based in that they minimize the error between the gradient maxima and the contour (external energy), and also the internal energy of the contour itself. This technique can be sensitive to undesirable local edge maxima. Work in [10, 15, 18, 20] do not address changing aspects. Uenohara and Kanade [18] use normalized cross-correlation for tracking and combine it with pose estimation, but do not handle changing aspects and claim robustness by examining invariant geometric constraints between features. Both Gennery and Walters [10, 20] employ a Kalman filter for predictive pose and edge based tracking. We agree with Lowe [15] in that a Kalman filter may not always be advantageous especially in enhanced reality applications. Lowe uses line data as image features with a weighted least squares fit to the model parameters. Matching itself is achieved via a best first search using Bayesian theory to measure the probabilities feature matches. Our technique is different from all these approaches in that it uses both intensity as well as edge information for tracking, uses least median squared pose computation to detect mismatches in tracking.

Tracking which is a central component in temporal registration has been addressed by using lines [15, 4, 16], edges [10, 20] including edge contours [5, 13] and intensity [18, 11, 17] including optic flow [1]. For example Crowley [4] used a set of parameterized line tokens which were matched using the Mahalanobis distance with predicted feature vectors. Sawhney [16] extended this approach to triples of lines, grouped under the shallow structure assumption under affine transformations. These techniques however are slow. Other model-based tracking such as [11] uses a hierarchy of features to represent a model. Constraints on the state of the feature are propagated down the hierarchy and at the lowest level tracking is accomplished using convolutions (edges) or SSD methods [1]. Hager does not explicitly address the issue of mismatches as is done by Shi and Tomasi [17]. Affine feature dissimilarity over multiple frames is used to identify good features to track. This method is image based, while ours is model based and can detect outliers after just one frame.

The pose estimation component in our paper is similar to [8] and the reader is referred to [14] for a detailed review. Although the use of aspect graphs is not new(e.g. [2, 3, 12]), most systems do not use coarse quantizations of the view sphere as employed in this system.

3 Registration System Components

In this section we describe each of the system components individually, paying particular attention to the tracking module which contains novel elements (the pose determination module is as presented by Kumar [14], and the feature indexing module is quite simple).

3.1 Tracking

The tracking module localizes a set of feature templates in a newly acquired image given hypothesized 2D feature locations. Within the context of the registration system, the hypothesized 2D locations are the positions of features (templates) obtained from the previous pose. The role of the tracking module is to find the position of the templates in the new image, by searching windows around their previous positions.

The basic algorithm for matching templates to image patches is a combination of normalized cross-correlation and steerable filters. The normalized cross-correlation of a template (image patch) $\tau(x, y)$ with an image $\gamma(x, y)$ at a location (i, j) is given in a computationally efficient form by

$$\tau * \gamma(i,j) = \frac{2 * \sum_{m,n} \tau(m-i,n-j) * \gamma(m,n)}{R1 * \sum_{m,n} \tau(m-i,n-j)^2 + R2 * \sum_{m,n} \gamma(m,n)^2} \quad (1)$$

$$R1 = \frac{\sum_{m,n} \tau(m-i, n-j)}{\sum_{m,n} \gamma(m,n)}$$

where, $R2 = \frac{1}{R1}$

Theoretically, this measure assumes that the surfaces in the environment are Lambertian, that they can be locally approximated by a plane, and that the illumination incident on the surfaces can be locally approximated by a constant. Under these assumptions the correlation measure is normalized in that it is independent of the illumination incident on the surface. However, good experimental results have been obtained with this measure on surfaces that do not

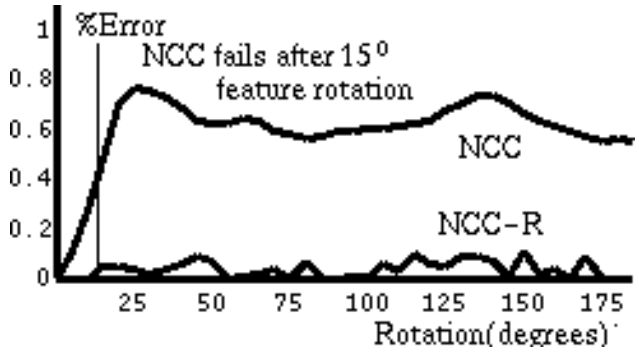


Figure 2: Comparison of NCC and NCC-R algorithms under 2D feature rotation

fit the Lambertian assumption(see [6] for a derivation and [7] for experiments with this measure).

Normalized cross-correlation degrades when there is a relative rotation between the templates and image patches. To compensate for 2D rotations it is sufficient to note that equation 1 is linear shift invariant in cartesian space and hence is translation invariant. Equivalently, linear shift invariance in polar space is equivalent to rotational invariance in cartesian space and we formulate an equivalent correlation expression in polar space.

A feature template is defined as a pair $\langle \tau, \theta_t \rangle$ where τ is an image patch centered over a dominant image edge and $\theta_t \epsilon [-\pi, \pi]$ is the phase of the maximum response of a steered Gaussian derivative filter [9] with the edge at the patch center. Templates are then localized within a search window γ in a new image as follows:

1. Spatial gradients and their orientations are computed by filtering γ with steerable Gaussian derivative filters and suppressing non-maximal edges within the search window.

2. Each local maximal edge location (i, j) in γ is a potential candidate for the new location of the template, and normalized correlation in polar space is used to identify the best match.

The advantage of using steerable filters is that they can be represented as a set of basis filters from which an arbitrary orientation of a template can be estimated [9]. For edges, first derivatives of Gaussian masks can be used. Their performance is better than that of box filters, for example, when there are non-step edges. While polar correlation compensates for any changes in orientation of the feature, there is however an issue of sampling and interpolation accuracy when going from cartesian coordinates of the image to the polar coordinates under which normalized correlation is performed. Accuracy is traded for speed to a certain degree in the real-time applications we have investigated, and sampling is performed without interpolation. In this system corners are routinely used as features. Observe however that first derivative operators will typically fail at a corner. Despite this, it is possible to obtain good tracking results by assigning to the template an angle sampled on any one of the edges leading to the corner. The reason this strategy works in tracking corners is because rotations are needed only to determine a sampling order and only the relative angle (between the template and the candidate match location) is important. During the localization process in a search window, a match is still found at the right spot, if there exists a location in the search window that corresponds to the sampled point. Alternatively, higher order filters may be used, at the expense of increased computation and noise.

The performance of rotation compensated normalized cross-correlation (NCC-R) is observed to be much better in terms of 2D rotational tolerance. Figure 2 shows the percentage correlation error (w.r.t. the auto-correlation value of the template) for varying rotations under NCC-R and normalized cross-correlation (NCC). $15x15$ templates correlated over $43x43$ search windows and while NCC fails after 15^o of feature rotation, NCC-R finds the correct matches over all rotations. Note that the NCC-R scores fluctuate due to the integer discretization of rotation space but never exceed 0.05%. NCC-R can correlate under arbitrary 2D rotations up-to the discretization of rotation and has been observed to work well with varying search windows and template sizes. Similar experiments with out-of-plane rotations showed that NCC-R can at best handle about 25^o of feature rotation.

3.2 Pose Estimation

The pose estimation module finds the transformation (both rotation and translation) given at least four 3D-2D correspondences. The transformation is what registers the artificial world to the real one, and is therefore the goal of the registration system. At the same time, this transformation is used as an index into the feature index table to predict what image features should be visible in the next image, and is used to project the corresponding points into the image frame as a starting point for the tracking module.

The pose estimation module uses Kumar's algorithm [14] to solve for the rotation and translation that maps a set of 3D model points onto corresponding 2D image points. Kumar's algorithm is an iterative approach that minimizes the squared image-plane distance from the data points to the projected model points. The Levenberg-Marquart method is used to solve this nonlinear optimization problem, starting from an initial "guess" of the approximate object pose. For small inter-frame motions, the change in pose between successive images is small enough that the pose from the previous image can be used as an initial guess in the pose algorithm.

Pose estimation techniques such as the simple version of Kumar's algorithm described above work well when the correspondence between model and data points is correct. Unfortunately, tracking errors will sometimes result in a model point being matched to an erroneous image point. Even a single such outlier can have a large effect on the resulting pose. Robust statistical approaches such as least median squares provide a powerful method to detect mismatches and cat-

egorize them as outliers. These methods are typically better than image based methods such as thresholding a correlation score which may vary from experiment to experiment and feature to feature. NCC-R for example, can produce a high correlation score at mismatches and thus a threshold will not work. Kumar used an approximation to least median squares, where subsets of size N(typically six) were sampled and transformations (both rotation and translations, calculated together) were computed for these samples. The sample which minimized the median of squares was used to eliminate outliers, and the final pose was computed using the remaining set of correspondences.

The computational cost associated with the least median squares filter over all subsets grows exponentially with the number of k sized subsets considered, where k ranges from the size of the original set down to 4 (the minimum required to compute pose). In practice, we generally compute pose from sets of five points, allowing the computation to grow with the number of points tracked (up to eight in the experiments investigated in this paper). With fast machines[4] (since the pose computation is purely compute bound) and for up to eight tracked points, the time expended in computing pose remains a fraction of the image acquisition and tracking time and is amenable to real-time performance.

3.3 Aspect Tables and Feature Indexing

As an object undergoes rotation with respect to the camera, features change in appearance, and new features may appear while old ones disappear. The range of viewpoints over which a set of features can be tracked is an aspect and an aspect table is used to encode sets of model points that are visible from each aspect. This table therefore contains lists of model points visible from the surface of a discretized sphere[5] encompassing the object, and is indexed using the latitude and longitude.

For this paper, model points were extracted manually and aspect tables were constructed off-line. In addition to the model points the aspect table is also used to store feature templates. At run-time, the current pose is used to determine the latitude and longitude. These angles are discretized to index into an aspect and a set of 3D points and possibly their corresponding templates are extracted for points that appear new for this aspect. In this paper the view hemisphere was discretized to 18 longitudes and 3 latitudes.

4 Discussion

The registration loop described in section 1 is examined for stability. Further (as discussed in 1) it is observed that pose and tracking can be combined to make the tracker adaptive. These properties of the registration system, issues concerning system speed, and an example demonstrating registration over distinct aspects are presented in this section.

[4] We are currently running on a Sparc-2.

[5] For the experiments in this paper, the feature index table encompassed a viewing hemisphere, since the object is not visible from below the table.

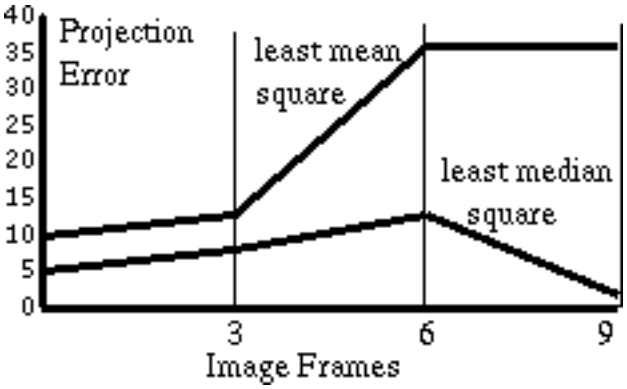


Figure 3: Least Mean Squares vs. Least Median Squares

4.1 Stability

To show that the system is stable, all possible sources of error or variation within the system need to be considered. There are three such sources.

The first source of variation (which in this case is not an error) is image motion. On each iteration of the loop, the system projects points based on the pose of the object in image N, and uses these positions as starting points for the tracker in image $N+1$. If the image motion of points is greater than the size of the tracker's search window, then tracking will fail.

The second source of variation is tracking error. There are several reasons why tracking might fail: specular reflections might distort the appearance of the feature, an un-modeled object might occlude the point being tracked, or the feature might not be unique, so that another point matches the template as well or better than the intended feature.

The third source of variance is pose computation error. As shown in simulation studies by Kumar [14], the residual pose error resulting from simple noise in the positions of point features is very small for most images. In practical systems, however, modeling errors and camera calibration errors can create non-trivial pose errors; in our experiments, we have noticed that projected model points may be off by as much as three or four pixels.

The overall system is stable because the tracking module compensates for pose error and image motion, while the pose modules compensates for tracking error. The median filter of the pose estimation module identifies mis-tracked points and excludes them from the pose computation. Since the starting point of the tracker on the next iteration is the projection of the model points from the computed pose (rather than the tracking results from the previous image) and outliers are not used for pose computation, tracking errors are not fed back into the tracker and the system remains stable. Pose errors, on the other hand, are indistinguishable to the tracker from image motion; they simply imply a disparity between the (slightly inaccurate) projected feature positions for frame N and their actual positions in frame $N+1$. One way to look at it is that the tracker never knows that the pose was wrong

– it just tracks the motion from the inaccurate computed positions to the new positions. As long as the tracker's search windows are big enough to accommodate the largest expected image motion plus the pose error, the result of tracking is not effected by pose error.

Catastrophic failures are possible, of course. If the tracking module produces more errors than the median filter was set to detect, an outlier will be included in the pose computation and produce a grossly inaccurate pose. One circumstance that might create such simultaneous tracking failures is if the image motion is larger than the tracker's search window size. This is essentially a configuration problem: the search windows must be large enough to account for the image motion plus the expected (typically small) pose error. Fortunately, if such a catastrophic failure occurs it will be detected in the residual error of the pose algorithm, and the user will be informed.

In a registration example illustrated in Figure 3 system performance under least mean square and least median square policies are compared. The figure plots the projection error sum for all the templates over a 9 frame out-of-plane object rotation, sampled every 3 frames. Given that there is about a 3 pixel error after pose computation, the expected value of projection error sum for seven templates used in this experiment is 21. At frame three during the object motion a specular reflection obscures one of the features. Over subsequent frames the least mean square policy cascades to failure starting with incorrect pose estimates and culminating in features falling outside the fixed search windows of projected template locations. Thus by frame six, the projection error goes high to 36, the system neither tracks nor estimates pose correctly and becomes unstable. In contrast, the median computation detects and eliminates the mis-tracked feature from pose computation, all the way through the duration of specularity (frame 6). Predictably the total projection error (which includes the mis-tracked feature's projection error) goes high at frame 6 but drops subsequently. This is explained as follows; Pose computation is un-affected from the tracking outlier and thus, the feature location is always within a search window from the projected template location. When the specularity disappears the tracker re-localizes the template and the projection error drops.

4.2 Adaptive Tracking

Templates can be updated on the fly as tracking is in progress. However, if a template mis-tracks updating it can cause incorrect feature-template associations(template slip), resulting in system instability. Template slip can be avoided as follows: First, a policy is adopted wherein only correctly tracked templates(those that are not outliers) are updated (by cutting out appropriate portions of the current image). Second, non-maximal suppression during the localization process ensures a correct sampling angle for the template (see section 3.1). Together, these two steps result in correctness of adaptivity, i.e. updating templates without introducing instability.

Adaptivity provides the system with several advantages. First, it allows for building aspect tables without regard to the tracker's sensitivity to out-of-plane rotations. Consequently the discretization of the object view hemisphere in this paper takes no consideration of the amount of rotation that the tracker can handle in 3D. It is purely based on the appearance or disappearance of features. A second important effect of adaptivity is that scale changes can be handled incrementally. Third, templates will need to be loaded once per every viewpoint transition for each new feature.

Set Size	11	8	6	4
Time(ms)	15	8.2	6.3	4

Figure 4: Time for Pose Computation

Template Size \ Search Size	19	15	11
15	180	150	70
11	137	77	54
9	80	47	20

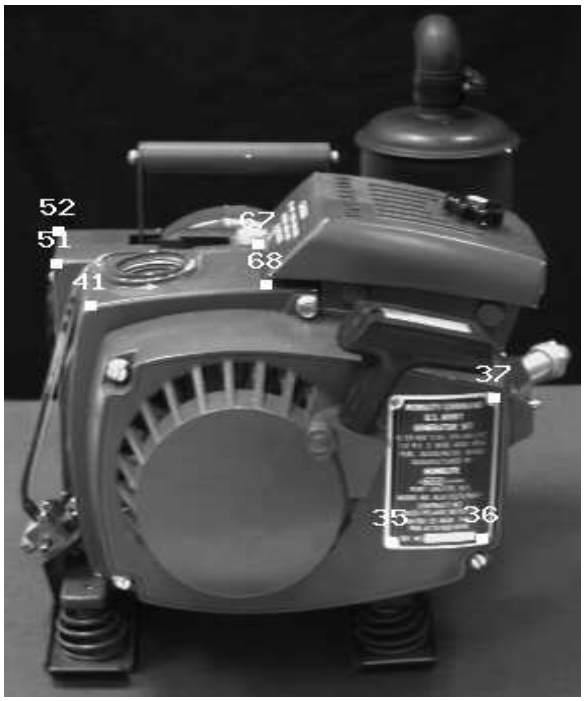
Figure 5: Time for Tracking

4.3 Registration Rates

The search window size and template sizes determine the speed of tracking. The rate of tracking in the worst case drops quadratically with increasing search window and template sizes. The median filter in pose computation is combinatorial and imposes an exponential increase in computation time with decreasing feature set size. Figures 4 and 5 tabulate the computation times on a Sparc-2 host for pose and tracking respectively. Note that in comparison to the tracking time, image acquisition time is much smaller since only windows around hypothesized locations are copied from the frame buffer. Similarly, pose computation times for eight templates is around 5% of the time spent tracking. For $11x11$ size search windows and $9x9$ templates, the computation speed of the tracker for 6 points is 8 Hz. If a maximum of one tracking outlier per frame is detected, the system can produce registration data at 7 Hz. Thus, within these limits registration is amenable to real-time performance.

4.4 Example: Registration across distinct aspects

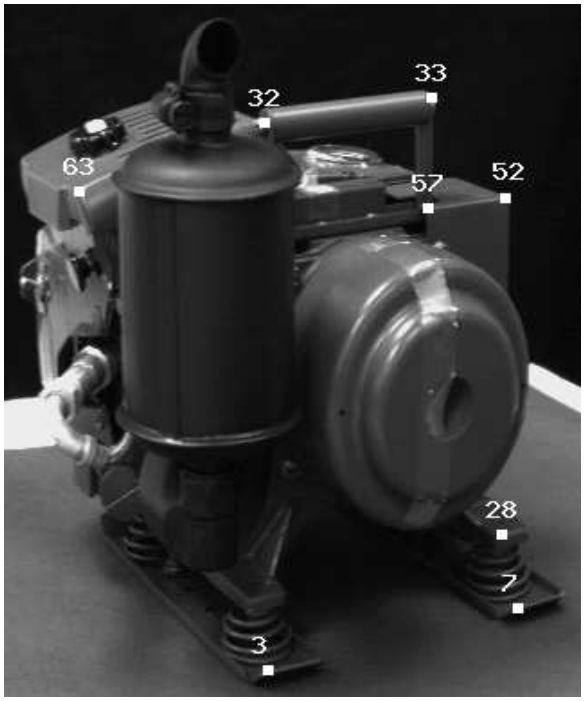
Using NCC-R, adaptive templates and median filtering registration of an object across 180^o out-of-plane rotation is demonstrated in frames 1 through 4 of Figure 6. An initial viewpoint corresponding to frame 1 in figure 6 is assumed. This gives a nominal pose, from which templates are extracted, projected in the image, and localized. Once initialization is complete the registration loop is automatic. Five to eight templates were used per aspect. Least median squared pose computation was used with subsets of five. Figure 6 must be read in text book fashion the snapshots of the registration sequence are sampled approximately at 0^o,45^o,90^o and 170^o.



Frame 1

Frame 2

Frame 3



Frame 4

Figure 6: Snapshots of Registration across 180° rotation

5 Conclusions and Future Work

Using an existing pose algorithm, a new tracking algorithm and aspect tables we have shown that it is possible to construct such a temporal registration system that is stable, operates in real time and handles changing views.

One of the limitations of our system is that pose computation is not predictive and therefore, search window sizes must not only encompass pose errors but also image motion. Kalman filtering style prediction has been studied by a number of authors and incorporating a Kalman filter is the next immediate step. Note however that in an enhanced reality application the agent is normally a human and may not conform to a low-order dynamical model. Examining the performance of a extended-Kalman filter would be an interesting step.

A second extension to the project is the automatic extraction of feature templates. Tomasi's work lays a basis for measuring feature dissimilarity over small frames of motion [17]. Small dissimilarities over a range of motion typically yield a good feature. This work is currently under examination.

Finally alternative techniques such as M-estimation that provide a faster computation and yet are robust statistical methods need to be examined for pose computation.

Acknowledgement

The authors thank the Laboratory of Perceptual Robotics for providing space and hardware to conduct experiments. They are also grateful to Profs. Riseman and Hanson for their support through the development of this system.

References

[1] Anandan, P., "A Computational Framework and an Algorithm for the Measurement of Visual Motion", *Int. J. Comput. Vision*, 2:283-310, 1989.

[2] Bowyer, K. W. and Dyer, C. R. "Aspect Graphs: An Introduction and Survey of Recent Results," *International Journal of Imaging Systems and Technology*, 2:315–328 (1990).

[3] Burns, J. B. and Leslie L. Kitchen. "Recognition in 2D Images of 3D Objects from Large Model Bases Using Prediction Hierarchies," *IJCAI-10*, Milan, Italy, 1987.

[4] Crowley, J. L. and Stelmaszyk, P., "Measurement and integration of 3-D structures by tracking edge lines", *Proc. European Conf. on Comput. Vision*, pp. 269-280, 1990.

[5] Dickinson, S. J., Jasiobedzki, P., Olofsson, G. and Christensen Henrik I., " Qualitative Tracking of 3-D Objects using Active Contour Networks", *Proc. of Comput. Vision and Patt. Recognition*, pp. 812-817, June 1994, Seattle, Washington

[6] Fennema, C. L., "Interweaving Reason, Action and Perception", *COINS TR91-56*, Dept. of Computer Science, Univ. of Massachusetts, Amherst, 1991.

[7] Fennema, C. L., "Finding Landmark Features Under a Broad Range of Lighting Conditions", *Proc. SPIE Intelligent Robots and Computer Vision X11: Algorithms and Techniques*, 2055:181-191, Boston, MA, Sept. 1993.

[8] Fischler, M.A. and R.C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM,* Vol 24, pp 381-395, 1981.

[9] Freeman, W. T. and Adelson, E. H., "The Design and Use of Steerable Filters", *IEEE Trans. Patt. Anal. Machine Intell.*, 13(9):891-906, Sept., 1991.

[10] Gennery, D., "Tracking known Three-Dimensional objects", *Int. J. of Comput. Vision*, 7(3):243-270, 1992.

[11] Hager, G. D., "Real-Time feature tracking and projective invariance as a basis for hand-eye coordination.", *Proc. Comput. Vision Patt. Recognition*, pp. 533-539, 1994.

[12] Ikeuchi, K. "Generating an Interpretation Tree from a CAD Model for 3D-Object Recognition in Bin-Picking Tasks," *International Journal of Computer Vision* 1:145–165 (1987).

[13] Kass., M., Witkin, A. and Terzopolous, D, "Snakes: Active contour models", *Int. J. Comput. Vision*, 1(4):321-331, 1988.

[14] Kumar, R. and Hanson, A. R. "Determination of Camera Position and Orientation," *Proc. of the DARPA Image Understanding Workshop,* Palo Alto, CA., May 1989, pp. 870–881.

[15] Lowe, D. G., "Robust Model-based Motion Tracking Through the Integration of Search and Estimation", *Intl. Jrnl. Comput. Vision*8:(2)113-122, 1992.

[16] Sawhney H. and Hanson A., "Tracking, Detection and 3D representation of potential obstacles using affine constraints", *Proc. Img. Understanding Wkshp.*, pp. 1009-1017, San Diego California, January 1992.

[17] Shi, J. and Tomasi, C., "Good features to track",*Proc. Comput. Vision Patt. Recognition*, pp. 593-600, 1994.

[18] Uenohara, M., and Kanade, T., "Vision-Based Object Registration for Real-Time Image Overlay", *(In Press) Jrnl. Comput. Bio. and Med.*

[19] Verghese, G., Gale, K. L., and Dyer, C. R.,"Real-time motion tracking of three dimensional objects", *Proc. IEEE conf. Robotics and Automation*, pp. 1998-2003, 1990.

[20] Walters, W. J., "Visual Servo Control of Robots using Kalman Filter Estimates of Pose Relative To Work-pieces", *in "Visual Servoing", Hashimoto, K., (ed.), World Scientific*, 1994.