Tracking Object Motion Across Aspect Changes for Augmented Reality*

S. Ravela B. Draper J. Lim R. Weiss
Computer Vision Research Laboratory
University of Massachusetts, Amherst, MA 01003
Email:ravela@cs.umass.edu

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 *augmented reality* applications such as an interactive repair manual. In this application technicians look through a visor at an annotation correctly aligned with the image of the 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 (since technicians will move), spatially accurate annotations can be overlaid 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 modeled object features and use pose estimation to update the object-camera transform. This approach is less expensive, can be used in unmodified environments and permits annotation of independently moving objects[1].

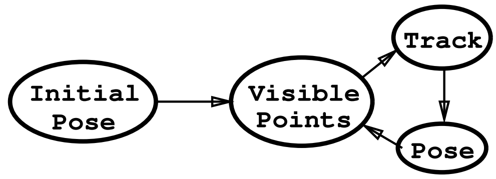


Figure 1: *The interaction between components of the registration system*

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

2 System Overview

The temporal registration system is initialized with a user specified 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]. User input is used to specify initial correspondences. These correspondences are used to estimate an initial pose. Once initialized, the system follows a simple three-step loop as shown in Figure 1: a) pose information is used to index into the aspect table and a list of visible features is extracted, b) the model coordinates of these features are projected into the next image plane as location hypotheses of feature templates, c) these templates are matched in this image 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 an 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.

* This work is supported in part by CSC, Booz Allen, under subcontracts CCC097IOM, 09005-0990-5847, ARPA (via TACOM) contract DAAE07-91-C-R035 and NSF under grants IRI-9208920, CDA-8922572 and IRI-9116297.

[1] Automatic positioning systems may do this if each object has its 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 a model coordinate on the object and a template that captures the feature's appearance.

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 [Kumar 92] uses camera parameters and the model-image correspondences to robustly solve for a rotation and translation that minimizes the projection error[4] of model points on the object in to the image plane. Robust pose estimates are obtained by using two alternative approaches. In the first approach, a median-filter is used to detect and exclude outliers during pose computation pose. The second approach is to use maximum likelihood estimation (M-estimation) of pose. In this paper an iterative re-weighting least squares (IRLS) form of M-estimation, namely, the modified weights method proposed by [Huber 81], is used.

One important result of the interaction of the pose and tracking components is system stability. Errors can be produced both during tracking and pose computation. Tracking errors arise when the template localizes incorrectly for reasons such as specularity or large inter-frame motions. Pose computation could be error prone if the object model or camera parameters are imprecise. The system can quite easily become unstable if these errors feed back in to the pose-tracking loop. The proposed system is shown to be stable. With appropriately sized search windows (based on expected inter-frame motion) the tracker compensates for feature motions which include errors induced as a result of the computed pose. Similarly, the median filter or IRLS based pose computation is designed to suppress tracking errors caused by mismatches. It is observed that neither tracking nor pose errors are fed back in to the pose-tracking loop, 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 drift from their intended locations. 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 on a Sparc-2[5]. 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 5.

[4] The projection error is defined as the distance between the actual versus the predicated location of a feature point.

[5] The registration system has also been ported to a Pentium laptop. Similar timing results were obtained.

3 Related Work

Temporal registration has been addressed by several researchers [Dickinson 94, Gennery 92, Lowe 92, Uenohara 95, Verghese 90]. Dickinson et. al. [Dickinson 94] use an aspect prediction graph together with a network of active contours introduced in [Kass 88]. 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 [Gennery 92, Lowe 92, Uenohara 95] does not address changing aspects. Uenohara and Kanade [Uenohara 95] use normalized cross-correlation for tracking and combine it with pose estimation, but do not handle changing aspects. Robustness is achieved by examining invariant geometric constraints between features. Gennery [Gennery 92] employs a Kalman filter for predictive pose and edge based tracking. We agree with Lowe [Lowe 92] in that a Kalman filter may not always be advantageous especially in augmented reality applications. This is because a low order dynamical model of human motion may not be always be appropriate except under very constrained scenarios. 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 of feature matches. Our technique is different from all these approaches in that it uses both intensity and edge information for tracking, and uses robust computation to detect mismatches in tracking.

Tracking, which is a central component in temporal registration, has been addressed by using lines [Lowe 92, Crowley 90, Sawhney 92], edges [Gennery 92] including edge contours [Dickinson 94, Kass 88] and intensity [Uenohara 95, Hager 94, Shi 94] including optic flow [Anandan 89]. For example Crowley [Crowley 90] used a set of parameterized line tokens which were matched to predicted feature vectors using the Mahalanobis distance. Sawhney [Sawhney 92] extended this approach to triples of lines, grouped under the shallow structure assumption under affine transformations. However, these techniques tend to be slow. Other model-based tracking such as Hager [Hager 94] 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 [Anandan 89]. Hager does not explicitly address the issue of mismatches as is done by Shi and Tomasi [Shi 94]. 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.

Although the use of aspect graphs is not new(e.g. [Bowyer 91, Ikeuchi 88]), most systems do not use coarse quantizations of the view sphere as employed in this system.

4 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 [Kumar 92], and the feature indexing module is quite simple).

4.1 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 predicted 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

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

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

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 fit the Lambertian assumption(see [Fennema 91] for a derivation.

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 [Freeman 91] with the edge at the patch center. Templates are then localized within a search window γ in a new image as follows:

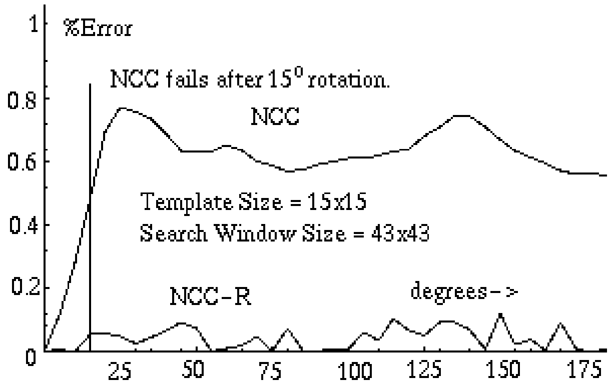


Figure 2: *Comparison of NCC and NCC-R algorithms under 2D feature rotation. 15x15 sized templates were correlated over a 43x43 area under several feature rotations.*

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 [Freeman 91]. 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. NCC-R can be used to track token features such as lines, curves and corners [Ravela 95]; the system described here tracks corners.

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; while NCC fails after 15^o of feature rotation, NCC-R finds the correct matches over all rotations. Note that for this case the NCC-R scores fluctuate due to the lack of uniform quantization of rotation space but never exceed 0.05%. Since the number of discrete angular bins increases with the radial extent larger windows will have a lower fluctuation. NCC-R can correlate under arbitrary 2D in-plane rotations up to the discrete quantization 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.

4.2 Pose Estimation

The pose estimation module finds the transformation (both rotation and translation) given at least four 3D-2D correspondences (although more correspondences are desirable). 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 [Kumar 92] 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-Marquardt 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 estimate for 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 provide a powerful means to detect mismatches and possibly categorize 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.

We have experimented with two alternative robust statistical methods for pose computation. The first is an IRLS technique, in particular one proposed by Huber as the modified weights method [Huber 81]. Using this method, the weights associated with each model-data pair are iteratively re-computed, so that outliers are simultaneously detected as the robust pose is computed. Upon convergence, a maximum likelihood estimate (M-estimate) of pose is produced. We use a variation adopted by Kumar where the weighting function takes the form of Tukey's bi-weight re-descending function [Kumar 92]. The IRLS technique (and other M-estimation techniques as well) is an excellent choice for the pose computation module of the registration system. Its asymptotic complexity is comparable to least mean squares but, in practice, IRLS tends to converge slower than least mean squares. However, the robustness provided by this technique far outweighs the reduction in speed. IRLS has a break down point of 30% and thus the method will not yield good results with over 30% outliers. It should be noted that while a 30% breakdown may seem modest from a statistical standpoint, this fraction of mismatches is rather large in a temporal registration system and perhaps reflects a poorly designed tracker.

In the alternative robust pose computation Kumar's approximation to least median squares is employed, where subsets of size N were sampled and transformations (both rotation and translations, calculated together) were computed from 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[6] (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.

It is instructive to compare the performance of IRLS and least median squares. It is clear that the complexity of least median squares is exponentially greater than that of IRLS. Thus, when large number of points may be tracked, IRLS will yield a faster system. However, for a small number of point sets (typically six to eight with one or two expected outliers) IRLS compares favorably with least median squares in speed and accuracy (see Figure 3).

4.3 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

[6] The system is currently running on Sparc-2 and Pentium processors.

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[7] encompassing the object, and is indexed using the latitude and longitude. In this paper the view hemisphere was discretized to 18 longitudes and 3 latitudes, leading to a total of 54 aspects and a 20^ox30^o viewing extent per aspect.

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 feature points are extracted. As the object rotates (or when the camera moves) aspect transitions will occur. During an aspect transition the new list of 3D points is compared with the current list. Templates are extracted for points that are new (i.e. appear in new list but not in current list). Old points (i.e. points in current list not in new list) are not tracked any further. This simple list management procedure ensures smooth transition between aspects.

5 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.

5.1 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 pose-tracking 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 matching 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 [Kumar 92], 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 pose estimation module identifies mis-tracked points and excludes them or minimizes their contribution 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 affected by pose error.

Catastrophic failures are possible, of course. If the tracking module produces more errors than the median filter or IRLS 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 size of the tracker's search window. 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 to re-initialize the system.

In the registration example illustrated in Figure 3, system performance under least mean square IRLS and least median square policies are compared. The figure plots the total projection error[8] for all the templates over a 92 frame out-of-plane object rotation. Given that there is about a 3 pixel error after pose computation, the expected value of the summed projection errors for seven templates used in this experiment is 21. At frame 64 (during 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

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

[8] Projection error is the distance between the actual and projected feature location in pixels.

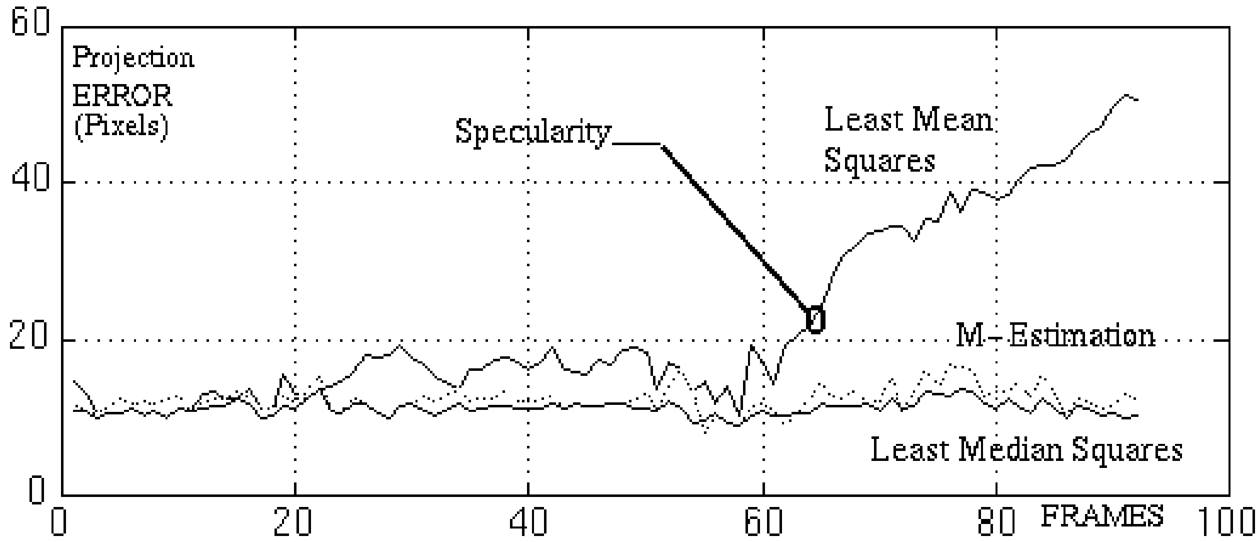


Figure 3: *Least Mean Squares vs. IRLS vs. Least Median Squares*

the fixed search windows of projected template locations. Thus by frame seventy (when the total projection error reaches about 38 pixels) the system neither tracks nor estimates pose correctly and becomes unstable. In contrast, both the IRLS and median computation detect and eliminate the mistracked feature from pose computation, all the way through the duration of the specularity. Predictably the total projection error (which includes the mistracked feature's projection error) remains small. This is explained as follows. Pose computation is unaffected by the tracking outlier and thus, the feature location is always within the search window centered around the projected template location.

5.2 Adaptive Tracking

Templates can be updated on the fly as tracking is performed. However, if a template tracks incorrectly, updating it can cause incorrect feature-template associations(template drift), resulting in system instability. Template drift can be avoided as follows. First, a policy is adopted wherein only correctly tracked templates(those that are not outliers[9]) 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 4.1). 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 building aspect tables without regard to the tracker's sensitivity to out-of-plane rotations. Consequently the discretization of the object view hemisphere used in this paper does not take into account the amount of rotation that the tracker can handle in 3D. It is based purely 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.

[9]Under IRLS outliers are determined by thresholding the final weights.

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

Table 1: *Time in milliseconds for Pose Computation*

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

Table 2: *Time in milliseconds for Tracking*

5.3 Registration Rates

A timing analysis of the system running on a sparc-2 processor in a VME cage is presented. The image acquisition hardware is a Datacube DigiMax frame grabber also mounted in the same cage. Images are grabbed through this digitizer and small regions corresponding to the size of the search window are transfered to the host. The time for this transfer is negligible. The tracking-pose loop is executed on the Sparc-2 host. In Table 2 the time in milliseconds for tracking under varying search and template sizes is presented. In Table 1 the time in milliseconds for least mean squares is presented. From these tables, it is observed that 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 slightly under 7 Hz with a least median squares policy. The computation with IRLS for 6 points is also 7 Hz because more iterations are needed to converge. Thus, within these limits registration is amenable to real-time performance.

5.4 Registration Example

Using NCC-R, adaptive templates and median filtering registration and annotation is demonstrated for an object undergoing 180^o out-of-plane rotation (frames 1 through 4 of Figure 4). The object (painted with a military green color) is the gas motor of a chain-saw that had been adapted for some other purposes. Note that this uniformity of color and the metallic nature of the engine make tracking a challenging task under natural lighting conditions.

An initial viewpoint corresponding to frame 1 in Figure 4 is assumed. This gives a nominal pose, from which templates are extracted, projected into 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 size five. In these figures an annotation "Open Gasket" is correctly aligned for the entire duration of object rotation and is meant to instruct the user (or technician) to open the gasket. The four snapshots in Figure 4 are sampled at approximately 0^o,45^o,95^o and 140^o. Feature 63 in frame 2 and feature 7 in frame 3 (marked in gray and black respectively) are outliers (the actual locations of feature 63 and 7 can be seen in frame 3 and 4 respectively) and are detected as such during pose computation.

6 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 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 augmented 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 in this kind of application would be an interesting experiment.

A second extension to the project is the automatic extraction of feature templates. Shi and Tomasi's work lays a basis for measuring feature dissimilarity over small frames of motion [Shi 94]. Small dissimilarities over a range of motion typically yields a good feature. This work is currently under examination.

References

[Anandan 89] Anandan, P., "A Computational Framework and an Algorithm for the Measurement of Visual Motion", *Int. J. Comput. Vision*, 2:283-310, 1989.

[Bowyer 91] 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).

[Crowley 90] 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.

[Dickinson 94] 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

[Fennema 91] Fennema, C. L., "Interweaving Reason, Action and Perception", *COINS TR91-56*, Dept. of Computer Science, Univ. of Massachusetts, Amherst, 1991.

[Freeman 91] 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.

[Gennery 92] Gennery, D., "Tracking known Three-Dimensional objects", *Int. J. of Comput. Vision*, 7(3):243-270, 1992.

[Huber 81] Huber, P. J., "Robust Statistics", *John Wiley & Sons*, N.Y. 1981.

[Hager 94] 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.

[Ikeuchi 88] 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).

[Kass 88] Kass., M., Witkin, A. and Terzopolous, D, "Snakes: Active contour models", *Int. J. Comput. Vision*, 1(4):321-331, 1988.

[Kumar 92] Kumar, R. "Model Dependent Inference of 3D Motion From a Sequence of 2D Images", *PhD Dissertation, CmpSci TR92-04*, Department of Computer Science, University of Massachusetts, Amherst.

[Lowe 92] Lowe, D. G., "Robust Model-based Motion Tracking Through the Integration of Search and Estimation", *Intl. Jrnl. Comput. Vision*8:(2)113-122, 1992.

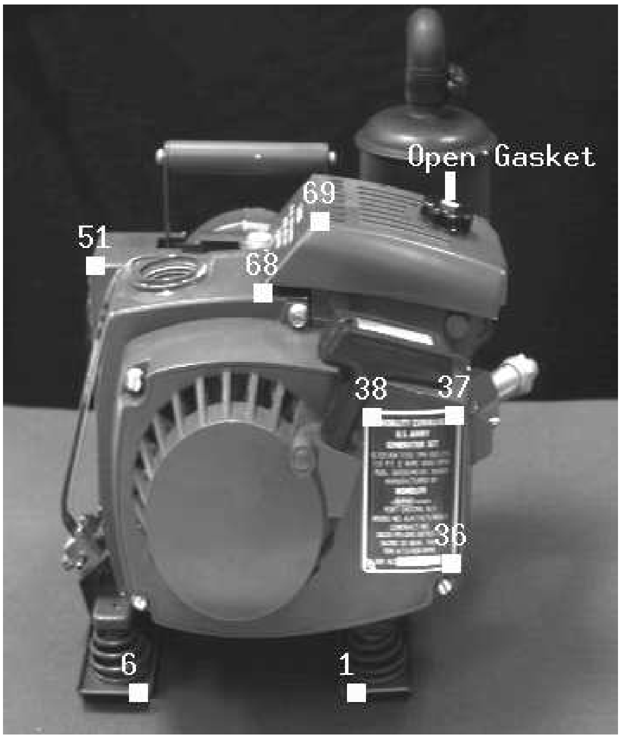
[Ravela 95] S. Ravela, B. Draper, J. Lim and R. Weiss, "Adaptive Tracking and Model Registration Across Distinct Aspects", *Proc. IEEE/RSJ Conf. Intelligent Robots and Systems*, pp. 174-180, Pittsburgh, Aug. 5-9, 1995.

[Sawhney 92] 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.

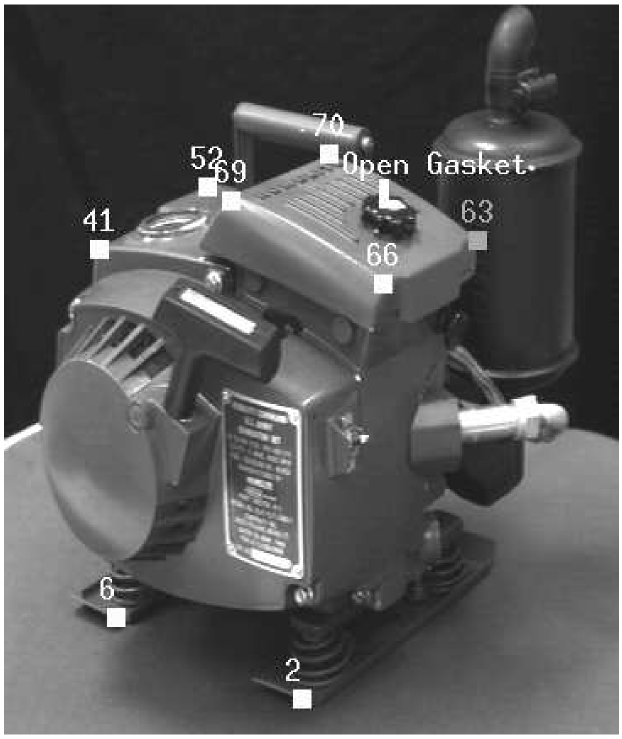
[Shi 94] Shi, J. and Tomasi, C., "Good features to track",*Proc. Comput. Vision Patt. Recognition*, pp. 593-600, 1994.

[Uenohara 95] Uenohara, M., and Kanade, T., "Vision-Based Object Registration for Real-Time Image Overlay", *(In Press) Jrnl. Comput. Bio. and Med.*

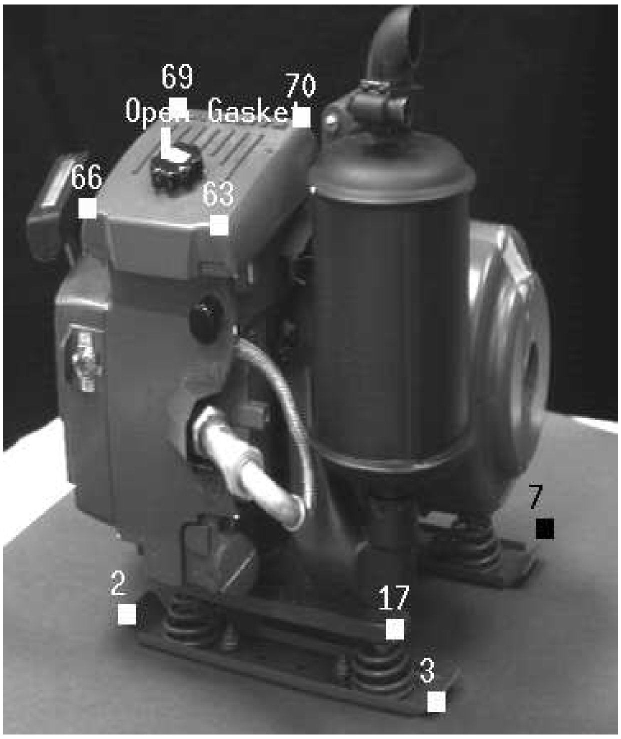
[Verghese 90] Verghese, G., Gale, K. L., and Dyer, C. R.,"Real-time motion tracking of three dimensional objects", *Proc. IEEE conf. Robotics and Automation*, pp. 1998-2003, 1990.



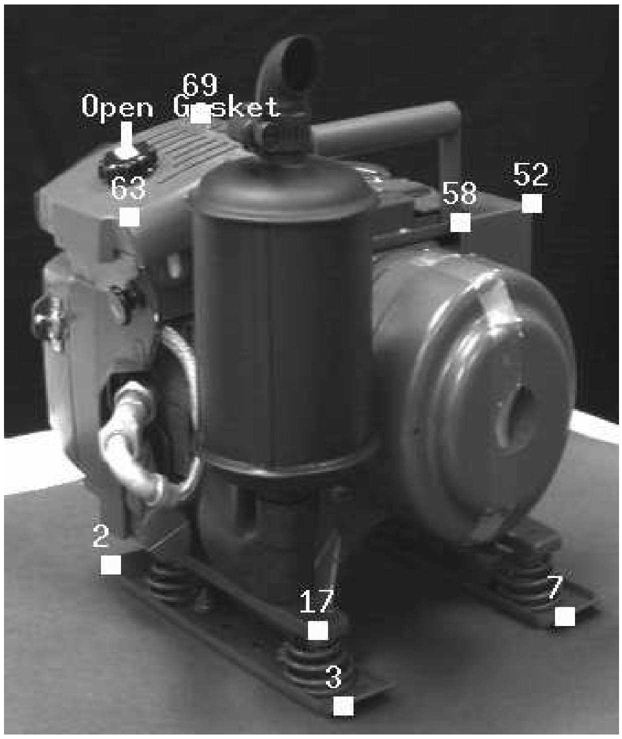
Frame 1



Frame 2



Frame 3



Frame 4

Figure 4: Snapshots of Registration across 180^o rotation