Adaptive Tracking and Model Registration Across Distinct Aspects

S. Ravela, B. Draper
J.Lim and R. Weiss

CMPSCI Technical Report 95-21
March, 1995

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
Internet: 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, is rotation compensated 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.

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

1 Introduction

Registration of moving real-world objects with their image space appearance has important applications in vision and robotics. The use of visual servoing in an assembly operation would entail continuous registration of the manipulated object model with the it's image over distinct aspects and preferably in real-time. Similarly, one can visualize an *enhanced reality* application such as an interactive repair manual. Instead of reading abstract descriptions that can easily be misinterpreted such as "remove the lateral stabilization screw", technicians could look through the clear visor of an enhanced reality helmet at the object being repaired. The object model's registration with it's appearance in the visor can be used to provide annotations or instructions such as "remove the screw" with an arrow pointing unambiguously at the screw that should be removed.

There are two basic approaches to the registration problem. One relatively expensive, yet proven technology is to instrument the real world with location beacons and position sensors. The other approach is to visually register the position of the helmet to the real world using image tracking and pose determination algorithms. This approach is cheaper and can be used in unmodified environments; it can also be used to track and annotate independently moving real-world objects, which the automatic positioning systems cannot (unless the moving object has its own beacon).

In this paper we combine a new tracking algorithm and a known pose determination algorithm [2] in a closed loop and demonstrate that robust, real-time registration is possible through 360° of camera or object rotation. In particular,

1. **Tracking Through 360° of Rotation.** Typically, when an object (or the camera) rotates through 90° of rotation or more, none of the features that were visible in the first frame are still visible (unless it is in the plane of the camera). Consequently, most image-based trackers cannot track objects through large rotations. By using the object model to predict when old points will drop out of view and when (and where) new points will become visible, our system is able to track objects through arbitrary rotations.

2. **Robustness and Stability.** The pose determination and tracking algorithms operate in a closed-feedback loop. Pose is used to hypothesize locations of object features that are tracked and the tracked features provide correspondences for the pose algorithm. Errors can be induced by both these components and system stability is an important

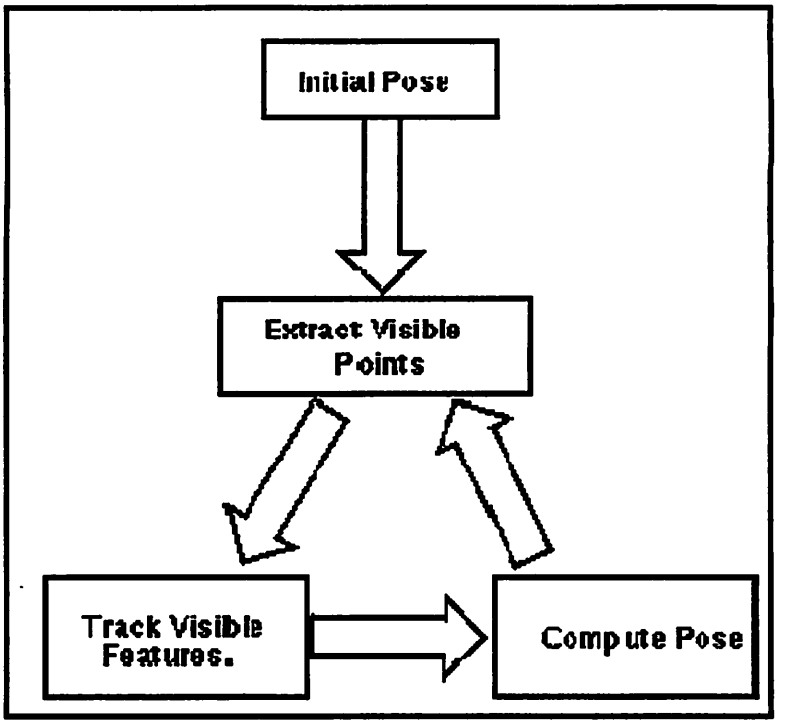


Figure 1: The interaction between components of the registration system

issue. We show that the system is stable (with a bounded error) in that tracking compensates' for pose errors and robust pose computation compensate for tracking errors.

3. **Real-Time Performance.** Enhanced reality applications must be able to keep pace with the real world. Although they do not necessarily have to produce pose estimates at frame rate – for some applications, the visor's screen may only have to be updated at 5 or 10 hertz, for example – the tracking system must track objects continuously as the helmet and/or objects move. The rate-limiting step is the image tracking, and we use steerable filters [19] to reduce the amount of search in the correlation process.

Our system achieves reliable, real-time object registration through 360° of rotation using a simple three (rather than two) step loop, shown in Figure 1. The loop starts with a set of features to track, expressed as templates with predicted (x, y) image locations.The first step is a tracking component that combines steerable-filter and normalized cross-correlation tracking algorithm that is able to track points through large translations and image-plane rotations, and through smaller (but non-trivial) out-of-plane rotations. The second step is a median-filter pose determination algorithm [2] that detects outliers in the tracking data while computing the camera pose that minimizes the object-to-image projection error. If the

residual error in this pose fit is too high, then more tracking errors have occurred than the median filter can compensate for, and an error is signalled. Otherwise the third step takes the pose computed by Kumar's algorithm and uses it as an index into a feature indexing table of visible points. This table (computed beforehand from the model, in a manner similar to aspect graphs [4]) maps viewpoints onto lists of visible model points. These points are then re-projected onto the image, and used as the set of (x, y) point locations for the next iteration of the cycle.

Within this cycle, it is the feature indexing table that allows the system to track objects through 360° of rotation. As the object rotates, new points become visible. The feature table maps the pose of the object onto a list of currently visible points, including those that were not visible from previous views. These points are then projected onto the image and given to the tracker as new points to track. Correspondingly, old points are dropped as they rotate out of view. Note that the feature indexing table also compensates for occlusions between modeled parts of the real world.

The aspect table also stores the templates required by the tracker. Feature templates are pre-compiled and associated with visible points for every aspect. However, the tracker is adaptive; every template that successfully tracks (i.e. is not an outlier after in the median pose computation step) is updated from the current image. This permits using wide aspects, reduces the storage requirement for pre-compiled templates and makes the system relatively insensitive to lighting changes.

Stability arises from the median-filter on the pose determination algorithm and the robustness of the tracker to unexpected image motions. Tracking errors, which can occur due to specularities, un-modeled occlusions or other image phenomena, are detected as outliers by the pose determination algorithm, which does not use them to compute the camera pose. Consequently, the predicted positions of image features in the next frame (which are computed from the pose) are unaffected by the tracking error, so the error is not fed back into the tracker. Pose errors, which can occur due to camera calibration errors, modeling errors and feature position noise, result in the predicted feature locations that are offset by the amount of the error. Such errors simply appear as additional image motion to the tracker, however, and as long as the pose error is not so extreme as to cause the new position of the feature to fall outside the tracker's search window, the pose error will not effect the result of the tracker, and therefore will not be fed back to the pose algorithm. (Stability is discussed in more detail in Section 4).

Finally, real-time performance is possible on current hardware, within reasonable limits. Exact timings depend on several factors, most notably the size of the search windows for tracking (which depends on the expected image motions), the size of the search window, and the maximum number of outliers to be detected. For $11x11$ size search windows and $9x9$ templates, the tracker can track 6 points at 8 Hz. If a maximum of one tracking outlier per frame is detected, the system can produce registration data at 7 Hz. The cycle times increase proportionally with the area of the search window and the number of points being tracked, and exponentially with the number of outliers detected.

2 Related Work

There has been some previous work on detecting mismatches during tracking. Shi and Tomasi [12] used a dissimilarity measure for detecting likely mismatches. They find the affine transform which maps a window around the tracked point in each image back to the window in the first frame. Then a measurement is computed over the sequence. This has the advantage that it is purely image-based, but it may take several frames before a mis-match is detected. Our model-based pose method can detect outliers after the first frame.

While most techniques employ an array of independent windows for tracking, other methods rely on a hierarchy, or network of features. Such is the approach adopted by Hager [13] who describes his objects in terms of basic (low-level) and composite (high-level) features, taking 3D projective information into account. Beginning at the top level, constraints on the state of the sub-feature (position and orientation) are propagated down the hierarchy. Tracking at the lowest level is accomplished using convolutions (edges) or SSD methods (image patches) [22], with epipolar constraints for stereo pairs. These results are then propagated upwards and used to evaluate the state of the composite feature. The advantage of using this approach in visual servoing is that the objective function and constraints are formulated in image coordinates. However, the reported work does not detect or handle mismatches. There has been some work based on extraction of symbolic image features such as line segments. Crowley [14] used a dynamic model composed of a set of line tokens, which were parametrized by position, orientation, and length. In each new frame the lines are matched based on Mahalanobis distance between feature vectors based on a Kalman filter prediction. Some issues not discussed in the paper include potential problems with detecting mismatches and real-time line extraction. If a line does not have a match based on its feature vectors, it is not immediately dropped, but its confidence factor is decreased.

Sawhney [24] extended this approach to triples of lines that transform according to an affine transformation. This property is satisfied by lines belonging to objects which have small variation in depth, i.e. shallow structures.

An approach which avoids the extraction of symbolic tokens is the use of direct image methods, e.g. active contours or snakes. In their work on estimating surface shape and curve tracking, Cipolla and Blake [23] do not rely on an explicit representation of object geometry, and use a variation of active contours which they refer to as the B-spline snake. The flexibility provided by the cubic B-spline representation in terms of local-effect and smoothness discounts the need for internal energy and allows for a reduction in the number of state variables, down to the number of control points. Similarly, physics-based model of features, namely active contours, or snakes are developed by [17].

The tracking work described above does not handle predictable events where faces, lines, or points may appear or disappear. Dickinson et al. [28] use an aspect prediction graph together with a network of active contours. When the area of an active contour shrinks below a threshold, it signals that a face is going to disappear. The aspect prediction graph can be used to determine what changes should be made to the network of contours. In contrast, our system works top-down using the pose to predict what features to track.

The registration system's feature indexing module is a lookup table that maps camera positions onto sets of visible features, in a manner similar to an aspect graph [8]. Although the feature indexing table used in the experiments in this paper was derived by hand, algorithms have been published for calculating aspect graphs for polygonal [11, 6], curved [10], and articulated object models. Bowyer and Dyer [4] provide a thorough review of aspect graph computation techniques.

Although the use of aspect graphs is not new (e.g. [5, 7]), most systems do not limit their complexity by using a coarse quantization of the view sphere. A coarse quantization is acceptable for tracking because it is not necessary to track every feature that is visible from a given viewing position.

For computing pose from 2D-3D correspondences, there are a number of other standard algorithms that could be used, e.g. [3, 25, 26]. In addition, there has been some other work on using robust statistics for computing pose, e.g. [27, 1]. In the work of Fischler and Bolles, subsets of size three are used to compute estimates of the pose and maximal subsets which are consistent with each estimate are found. The value with the largest consistent subset is chosen. For a more detailed review see [2].

A few researchers have examined a complete registration system. Notable among them are [20, 18, 21]. Alternative approaches that are bottom-up and image independent have been studied by [28] and have been reviewed earlier in this section. Gennery describes a registration system where they tracks known 3D objects. Although they employ Kalman filter style prediction for pose, the tracking system is edge based. Similarly Walters employs a kalman filter for feature prediction and closes a feedback loop around the pose of the object. Gennery uses multiple frames to compute pose and uses features as weights in the covariance matrix. Verghese develops an interesting system where registration is performed using two approaches, the first is called a motion searching paradigm where changes in the feature configurations (i.e. edges) are used to hypothesize changes in the pose space and the pose with least change is selected. In the second method changes in the feature space are used to compute changes in pose space directly. In this sense the registration system presented in this paper is based on a motion calculation paradigm. However, unlike any of the above approaches we use a robust localization scheme that is tolerant to arbitrary feature rotations in 2D. Further we use both edge and image data to localize image features. Further, use of edge information significantly reduces the search space and feature mismatches. Pose computation from tracked model-image correspondences employed robust statistical methods that results in a stable registration system.

3 Registration System Components

As shown in Figure 1, the registration system is composed of three basic modules: tracking, feature indexing, and pose estimation. In this section we describe each of these components individually, paying particular attention to the tracking module which contains novel elements (the pose determination module is as presented by Kumar [2], and the feature indexing module is quite simple). In Section 4 we describe how the modules interact, focusing on the underlying sources of error and how each module compensates for the others.

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) in the previous image. 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 size of the search window is a critical parameter: it determines the maximum displacement between the previous and new feature position, and therefore the amount of image motion that is allowed between frames. Unfortunately, the rate of tracking is inversely proportional to the search window size, so there is a tension between allowing greater inter-frame motions and processing more frames per second.

The basic algorithm for matching templates to image patches is a combination of normalized cross-correlation and steerable filters. The intuition behind normalized cross-correlation is that the incident light falling on a small patch of a single image is approximately constant, but that the incident light may change from frame to frame. By normalizing each pixel, it is possible to correlate relative changes in intensity, which are less sensitive to temporal lighting variations than the intensities themselves. The normalized cross-correlation of a template (image patch) $\tau(x,y)$ over a set with an image $\gamma(x,y)$ over the set at a location (i,j) is given in a computationally efficient form by

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

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

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 are only weakly Lambertian (see [15] for a derivation and [16] for experiments with this measure).

However, normalized cross-correlation degrades when there is a relative rotation between the templates and image patches. Figure 2(i) illustrates this problem. A template of size $15x15$ was extracted in Frame 0 at the center of the small rectangle. This template is then correlated over a $43x43$ search window in each subsequent frame (numbered by the

rotation in degrees with respect to frame 0) within the large rectangles. After 15° of rotation the template begins to mismatch and never matches again. Figure 3 plots the percentage correlation error (w.r.t. the auto-correlation of the template) over these rotations. The error increases dramatically and then stays high at all the mismatches. Further the degree of rotational tolerance varies only slightly with changing template sizes and search windows with larger templates being less tolerant to rotations.

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 [19] 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.

Normalized correlation (described in equation 1) is computed in polar coordinates parametrized by radius(ρ) and angle(θ); however, the image and templates are parametrized by row and column. The expression in polar coordinates equivalent to the numerator in equation 1 is given by:

$$2 * \sum_{\rho=0,\theta=0}^{\frac{s^2}{2},2\pi} \tau(m_0, n_0) * \gamma(m_1 + i, n_1 + j)$$

where $m_0 = \rho * cos(\theta + \theta_t)$, $n_0 = \rho * sin(\theta + \theta_t)$,$m_1 = \rho * cos(\theta + \theta_i)$, $n_1 = \rho * sin(\theta + \theta_i)$ and $\theta_i \epsilon [-\pi, \pi]$ is the orientation response at (i, j). The equivalent term for the denominator in equation 1 can be determined similarly.

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 [19]. For edges,

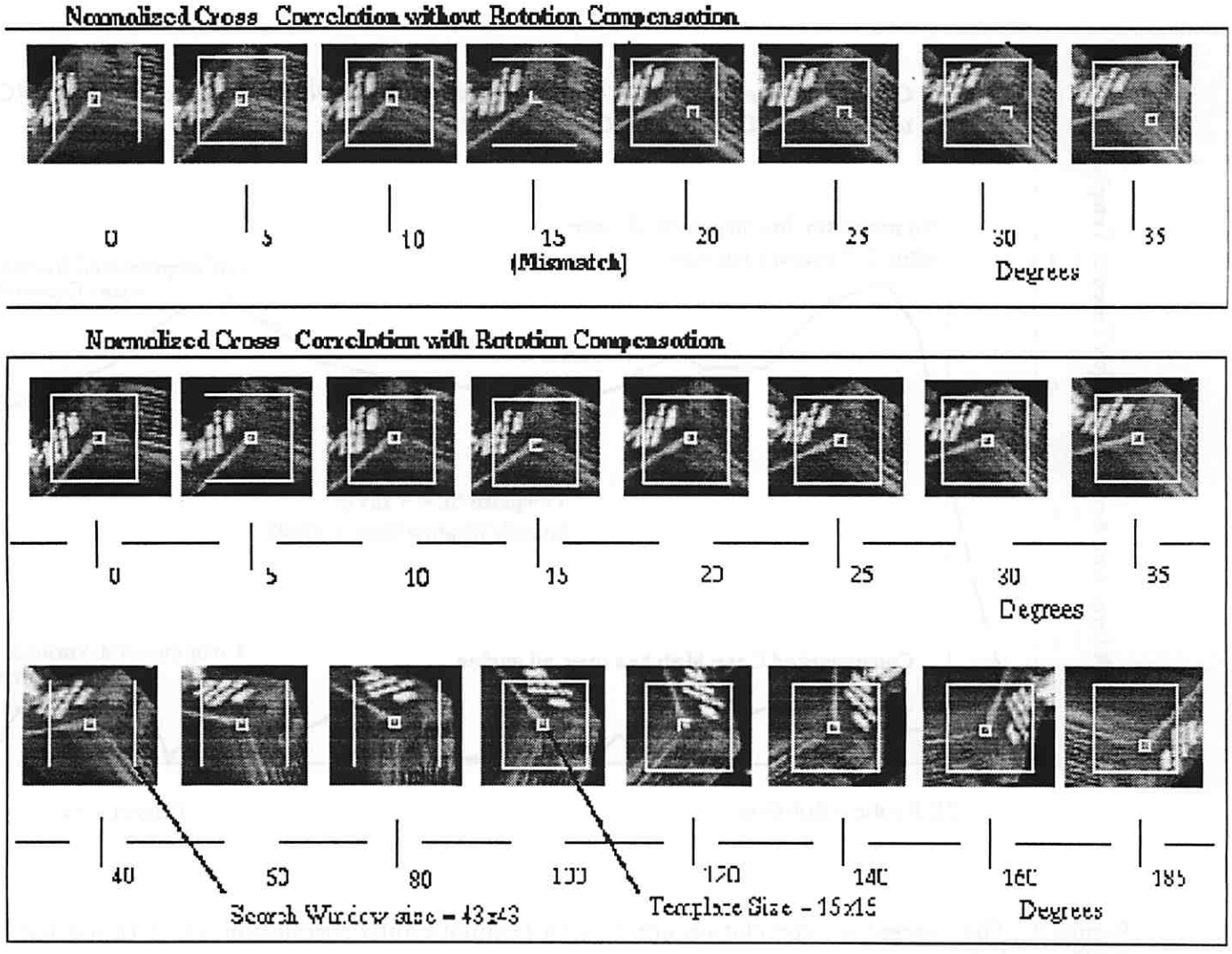


Figure 2: Comparison of NCC and NCC-R algorithms under 2D feature rotation

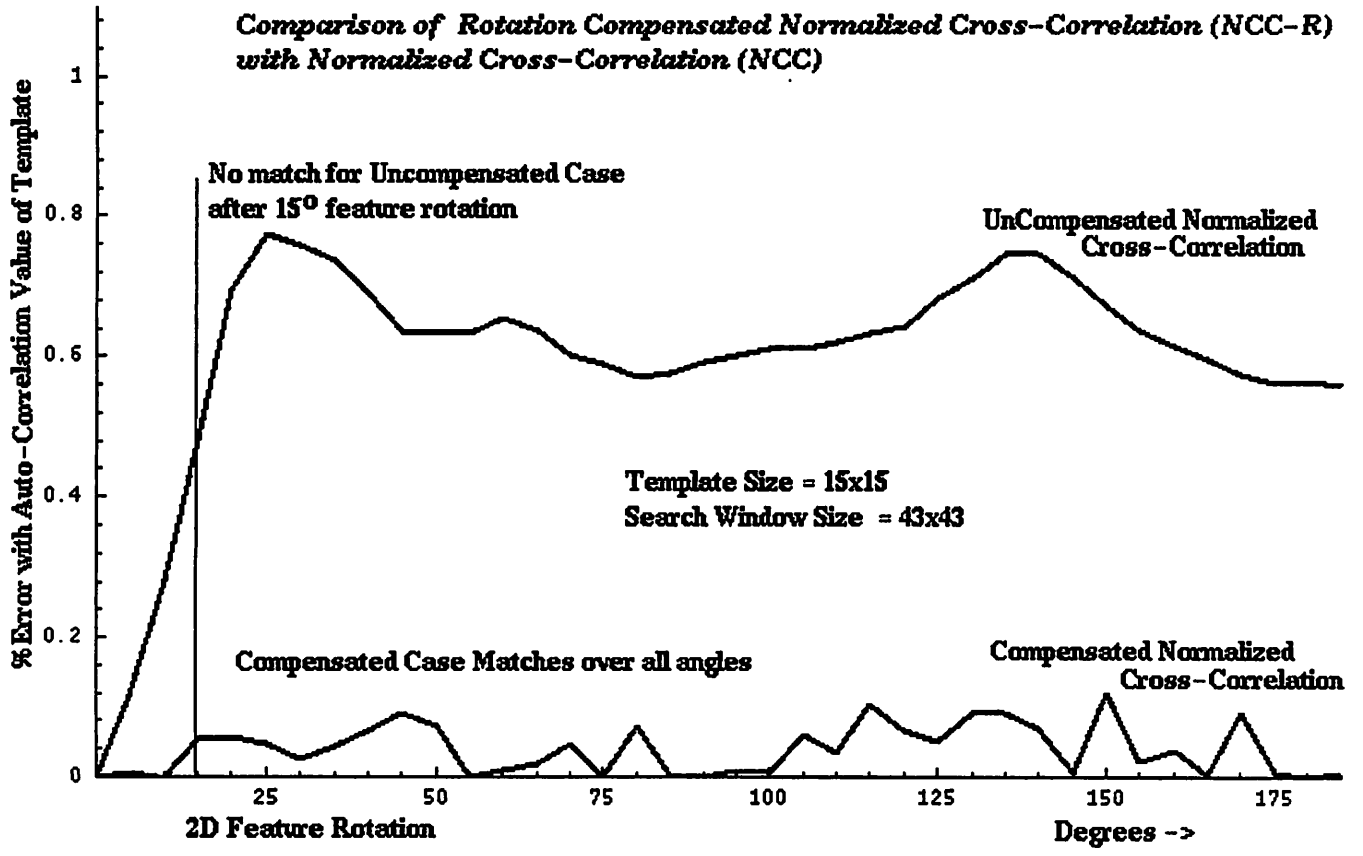


Figure 3: The percentage correlation error (with template auto correlation vs. rotation for NCC and NCC-R

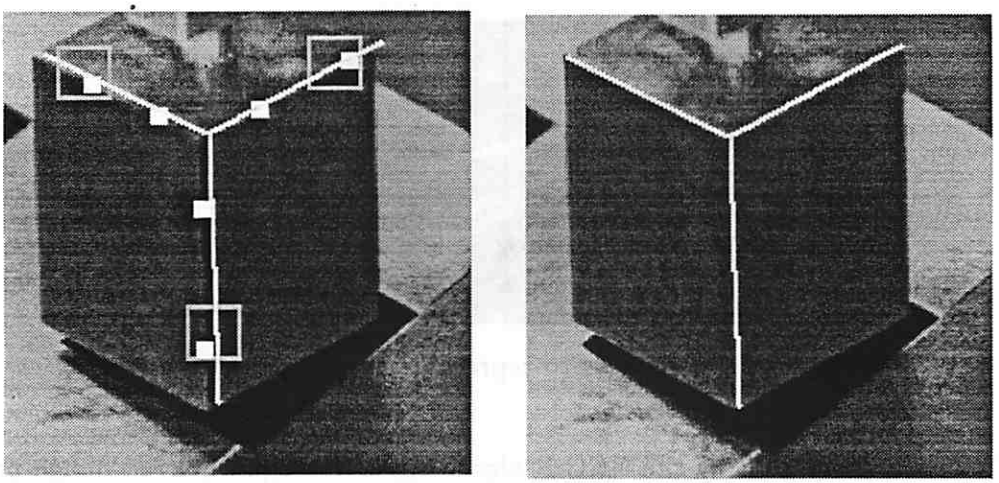


Figure 4: Two templates are used to capture lines.

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.

3.1.1 Performance of NCC-R

The performance of rotation compensated normalized cross-correlation (NCC-R) is observed to be much better in terms of rotational tolerance. Figure 2(ii) illustrates the performance of NCC-R under 2D rotation. Similar to the previous corresponding experiment templates of size $15x15$ were extracted from frame 0 and then correlated in search windows (represented by the large rectangles). The small rectangles indicate the location of best match and NCC-R is observed to correlate well over all rotations of the feature. Figure 3 shows the percentage correlation error (w.r.t. the auto-correlation value of the template) for varying rotations. Note that this value fluctuates due to the integer discretization of rotation space but never exceeds 0.05%. Unlike normalized cross-correlation however, 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.

3.1.2 Template Aggregates and Features

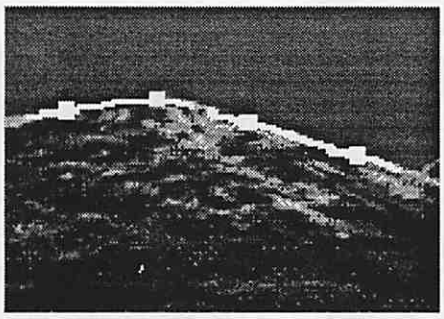


Figure 5: Using templates to represent an image curve

The definition of a template used by the NCC-R algorithm is an edge centered image patch. This specification can be used to capture features such as lines, curves or corners. It is sufficient to use two templates to define an image line. Figure 4 illustrates this case. The left picture shows a projected model, and the filled rectangles show the templates. The right picture shows the lines defined by the template. Similarly, curves can be approximated by using piecewise straight line approximations between templates. Points of high curvature are used as locations for templates and the curve is then represented as a list of line segments joining the templates. Figure 5 illustrates this case. The templates and the curve they define are plotted in the picture.

Corners are a special case. Note that first derivative operators will typically fail at the corner. However, it is possible to assign to the template an angle from 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 it does not matter what angle is chosen, so long as it falls on an edge. 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.

3.2 Pose Estimation

The pose estimation module finds the transformation (both rotation and translation) that maps a set of 3D model points onto the observed image points. 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 [2] 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 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. Kumar used an approximation to least median squares, where subsets of size N 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 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 six 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[1] (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.

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

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[2] encompassing the object, and is indexed using the latitude and longitude. The number of aspects typically depends on how sensitive the tracker is to rotation. The tracking module described above can handle at worst up to 20° of 3D rotation (see section 5.1 for experiments).

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 their corresponding templates are extracted extracted for points that appear new for this aspect. As stated in the previous section these 3D points (transformed into the camera frame) are projected as feature location hypotheses.

3.4 Adaptive Templates

Templates can be updated on the fly as tracking is in progress. However, if a template mistracks updating it can result in system instability. Templates that match correctly i.e. are not outliers are updated by cutting out appropriate portions of the current image. The edge-detection and local-maxima operations ensure that this patch will be edge centered. The median filtering step ensures that mismatched templates are not updated and feature points that are detected as outliers continue to retain their old templates.

This technique has several advantages. First, fewer templates will have to be recompiled thereby reducing the amount of memory required and making the system more robust to lighting changes between experiments. Second, this allows for wider aspects. Thus a new aspect is needed only when a new feature point appears or a current one disappears. Third, adaptation makes the system handle incrementally the depth changes between the object and the camera. In this paper the view hemisphere is discretized into 18 longitudes (representing a 360° rotation) and 3 latitudes (representing a 90°) rotation. This discretization takes no

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

consideration of the amount of rotation that the tracker can handle in 3D. It is purely based on the appearance or disappearance of features.

4 Stability Analysis

Describing the three modules in isolation does not explain why the system as a whole has bounded errors. To show that the system is stable, we have to consider all possible sources of error or variation within the system. In this case, 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 enough, so that another point matches the template as well or better than the intended feature. Additionally, if the system was not configured properly, the underlying image motion could imply that the tracked point is outside of the search window.

The third source of variance is pose computation error. As shown in simulation studies by Kumar [2], 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 stability of the overall system is guaranteed because the tracking module compensates for pose error and image motion, while the pose modules compensates for tracking error. The result is a system with bounded errors. To be precise, the median filter of the pose estimation module identifies mistracked 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 the outliers were not used in the 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.

5 Experiments and Discussion

In this section we conduct experiments that demonstrate the performance of the tracking module under mid-range rotations, illustrate failure modes of the tracker under un-modelled events such as specularity, and demonstrate failure of tracking under least mean squares. Then we demonstrate tracking using least median squares over a 180° rotation of the object relative to the camera.

5.1 Midrange Feature Rotations and Tracking Failure Modes

Figures 6 and 7 show the performance of tracking under a 3D rotation. The object was rotated in approximately five degree steps about the vertical axis and the frame numbering indicates the rotation. Templates of size $9x9$ were extracted from the frame numbered 40, indicated by the small rectangle in the picture. The algorithm NCC-R was then applied in a $31x31$ window indicated by the large rectangle. In all frames except frame 40 the small rectangle then indicates the location of best match. These frames must be read left and right of frame 40 indicating positive and negative rotations about the midrange point. The correlation withstands about $+/-25°$ rotation about the midrange. As with the rotation compensation experiments, the search window size did not affect the degree of rotation tolerance. However, varying the the template size significantly changes the ability to tolerate rotations. Under a template size of $21x21$ for example the rotation about midrange drops to

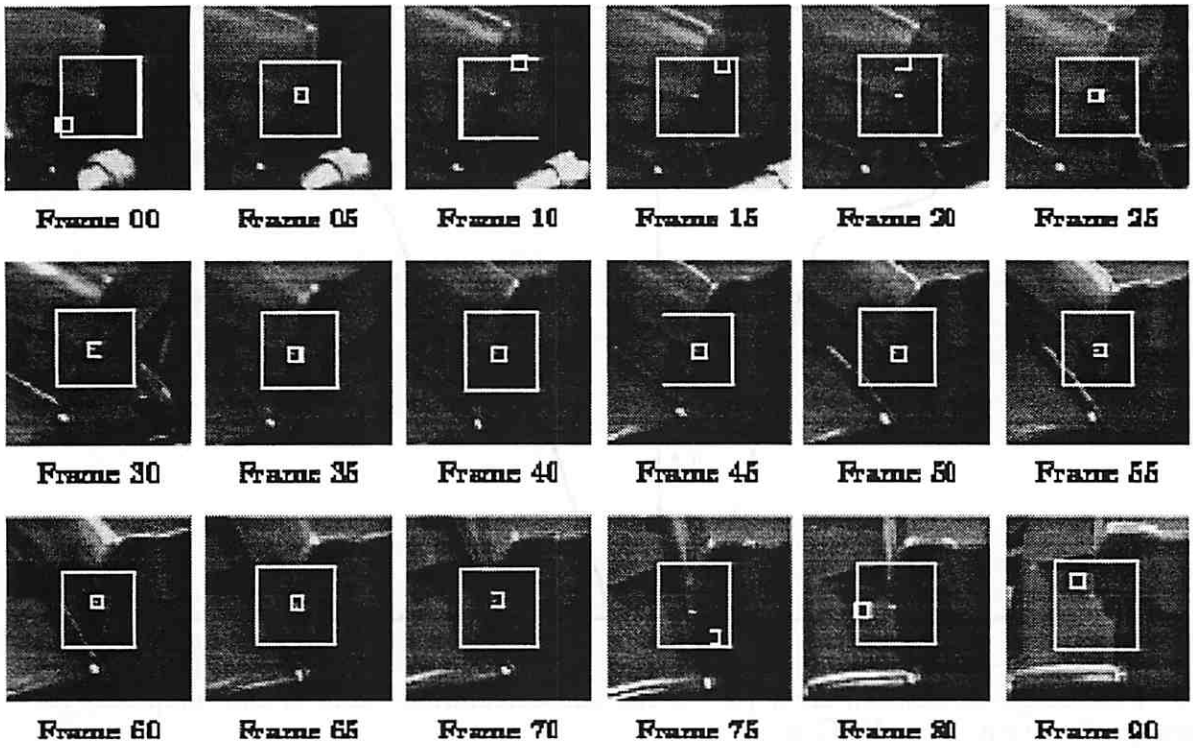


Figure 6: Illustration of performance of NCC-R under 3D feature rotation

$+/-20°$.

This experiment demonstrates that in the absence of unpredictable circumstances such as specularity and occlusion for example and when the feature does not undergo a geometric singularity under perspective projection, then NCC-R performs well even under 3D rotations. Further, since the window sizes do not affect the degree of rotation tolerance for any given template size, the tracking algorithm therefore can handle significant amounts of error in 2D feature hypothesis. These errors as noted earlier may arise both from image motion as well as pose errors.

5.2 Registration employing Least Median Squares(LMS)

In section 3 we argued for the use of least median filtering as a robust technique for detecting tracking errors that may arise from unpredictable events as opposed to least mean squares which may fail under these circumstances. Figures 8 and 9 illustrate the failure of least mean squares. In this experiment the object undergoes a rotation of about the vertical axis and template numbered 66 encounters a specularity. The tables adjoining the figures indicate the projection error associated with each template. The template sizes used in this experiment were $15x15$ and search window sizes were $11x11$. The frames shown here are

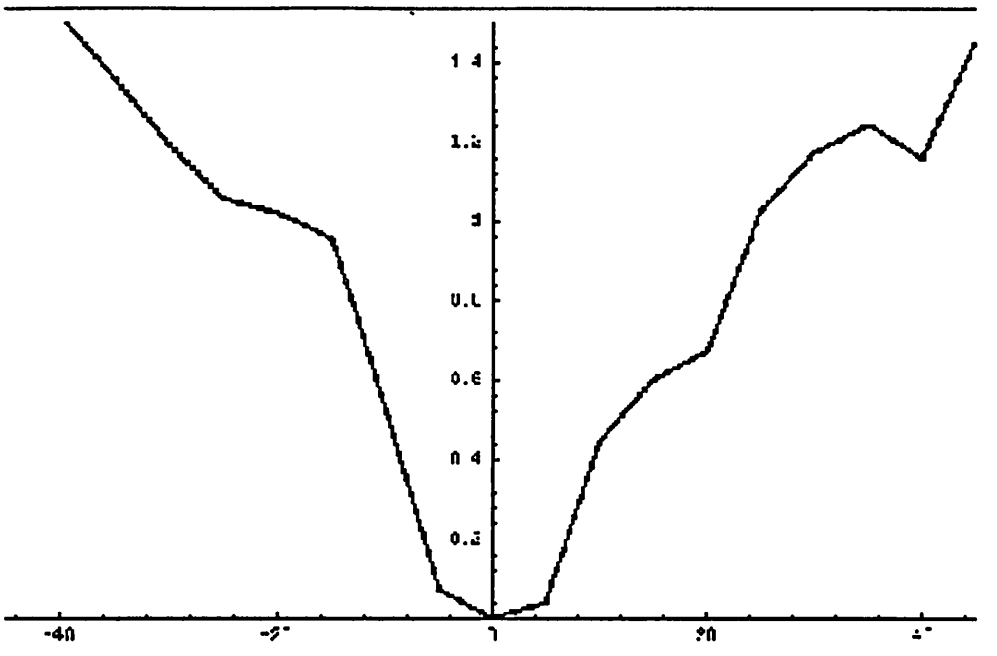
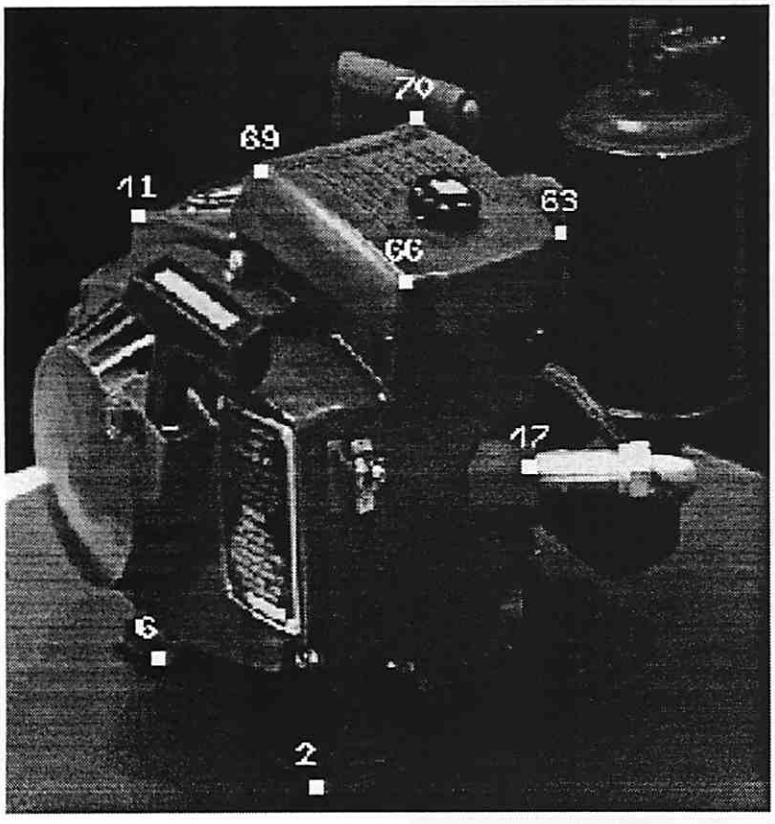


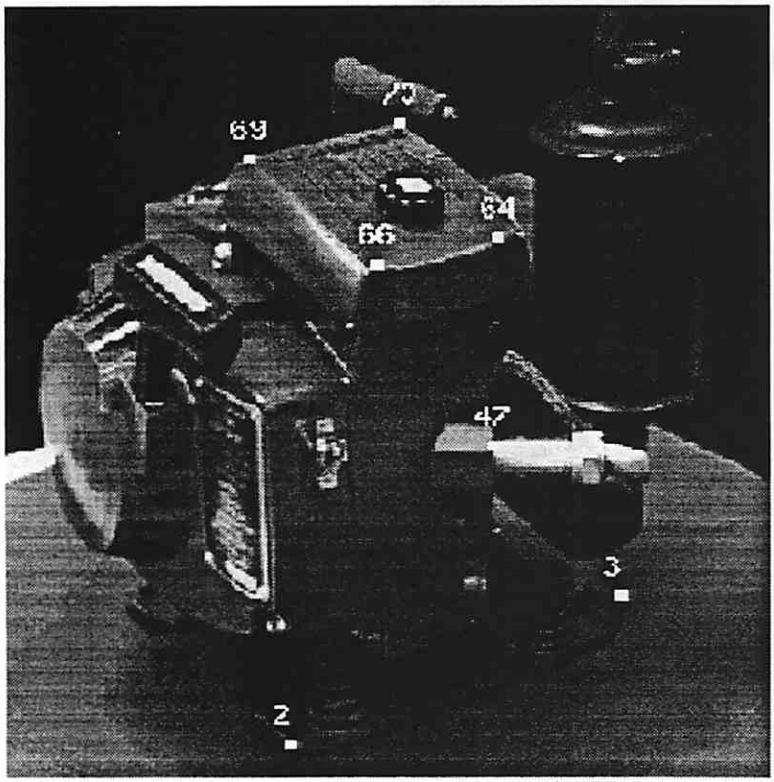
Figure 7: Percentage correlation error (w.r.t. the autocorrelation value of the template) vs. 3D midrange Rotation using NCC-R

sampled every third frame out of a rotation sequence of approximately 1°. The first frame indicates that all features are tracked correctly and the corresponding projection error is low. In frame 2 specularity reaches the corner numbered 66 and the template mistracks. This is indicated in the high projection error observed in the corresponding table. By the time the rotation reaches frame sample 3 (in figure 9 - three frames have past in between but have not been shown for lack of space) almost all features except 2 and 47 mistrack as a result of incorrectly template updates. The reason for this is that the pose is significantly off by this frame especially features 66, 69 and 70. In frame 4 we see that the projection errors drop. At this point both the pose and tracking are in error and the system has become unstable.

Figures 10 and 11 demonstrate the corresponding sequence employing least median squares. The template and search window sizes are exactly the same as above. In this sequence however, the marked feature 66 is the projection after least median pose computation as opposed to the tracked location. Throughout the entire sequence feature 66 is projected correctly and in frame 3 both the projection and the tracked location are shown for this feature. Observe that while the feature mistracks (the corresponding pose error is large) the pose computed drops this feature from least median squares and as a result the projection of the feature appears correctly albeit within a 3 pixel tolerance from the actual feature location. Further none of the other features have mistracked and their corresponding pose

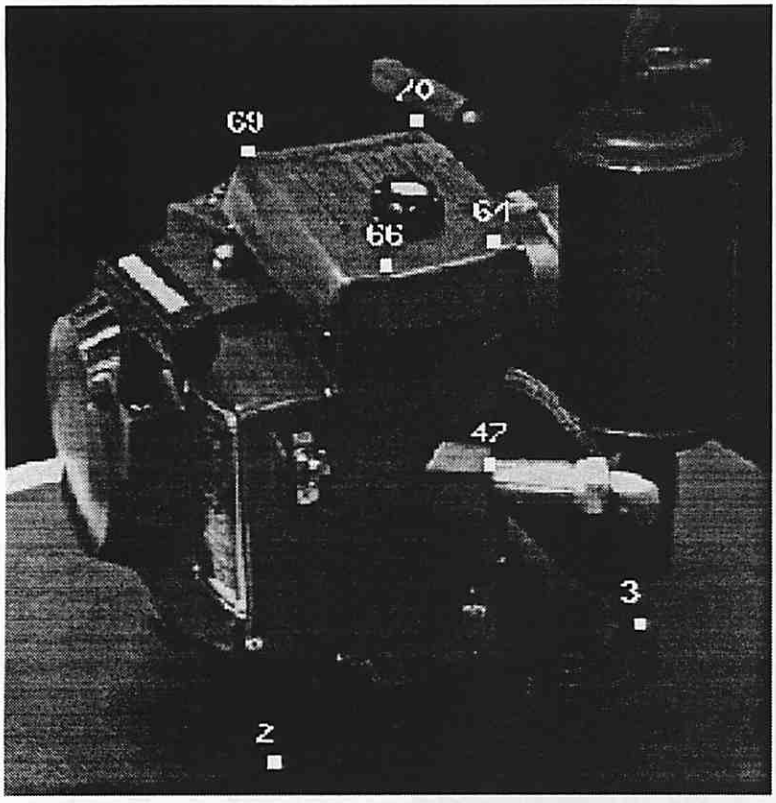


Template	Error
2	2
6	1
41	0
47	1
63	0
66	0
69	3
70	1

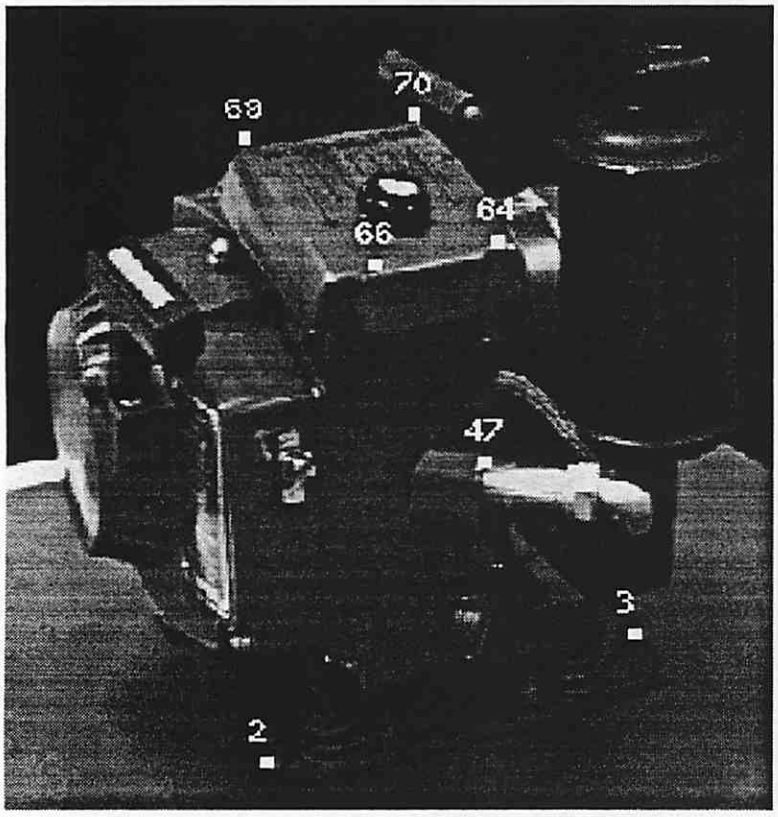


Template	Error
2	2
3	2
47	2
-	-
64	0
66	6
69	0
70	1

Figure 8: Failure under least mean squares pose computation



Template	Error
2	2
3	6
47	2
-	-
64	3
66	11
69	6
70	6



Template	Error
2	2
3	4
47	7
-	-
64	3
66	5
69	7
70	8

Figure 9: Failure under least mean squares (contd.)

error is low indicating that the system is stable.

5.3 System Performance

Here we demonstrate the registration system traversing approximately 180° of rotation of the object around the vertical axis. The tracker employed adaptive NCC-R as the localization scheme and least median squares were used for pose computation. The number of features tracked per aspect ranges from a low of 5 to a high of 8. Least median squares was used when the number of features was greater than 5 and in other cases least mean squares was employed. Figures 12,13,14 and 15 must be read in sequence and each of the figures must be read in a text book fashion. The white markings are those of the tracked location on each frame and there is approximately a $10° - 15°$ rotation from frame to frame.

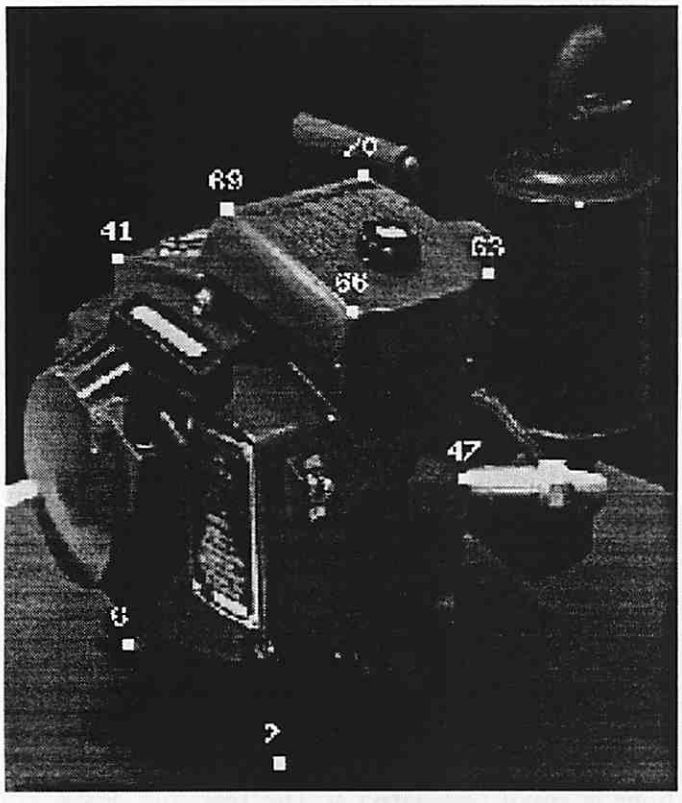
Another important system issue is the measure of time spent in the tracking and pose computation components. The timing results of NCC-R per template and the pose computation are tabulated in figure 16. Note that for about eight templates the time spent in computing pose is only a fraction (5%) of the time spent in tracking. The additional overhead of acquiring the image from the frame grabber is small compared to the tracking costs, since only subsections of the image are transferred to the host.

6 Conclusions

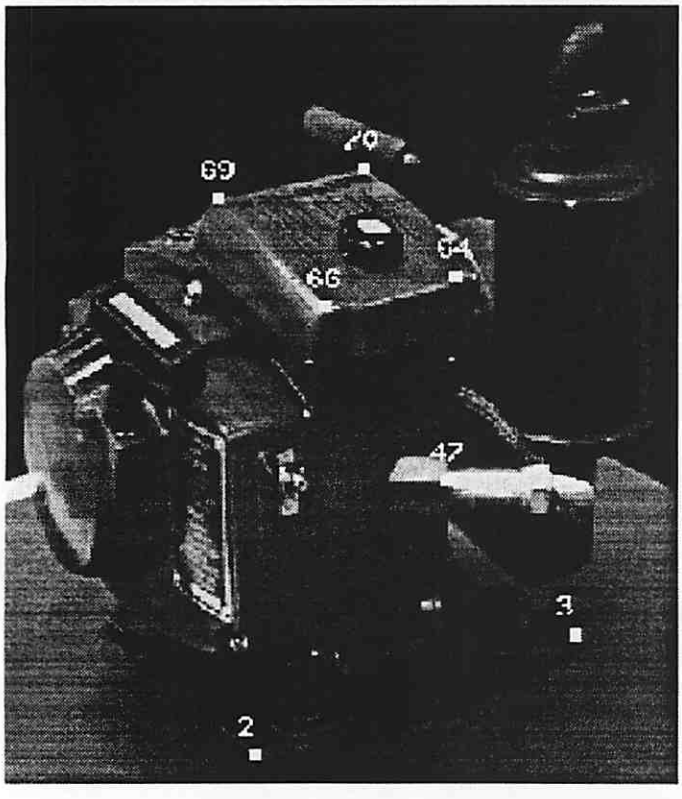
Enhanced reality applications such as an interactive manual require the registration of the real world with the observed imagery. Registration entails the computation of pose and the ability to track object features over distinct aspects. Using an existing pose algorithm, a new tracking algorithm and aspect tables we have shown that it is possible to construct such a registration system that is stable and operates in real time. The tracking algorithm is capable of handling rotations both in the image plane and in depth. Further, the tracker compensates for pose errors that arise from modelling and calibration inaccuracies and the least median pose algorithm compensates for tracking errors.

7 Limitations and Future Work

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 fil-

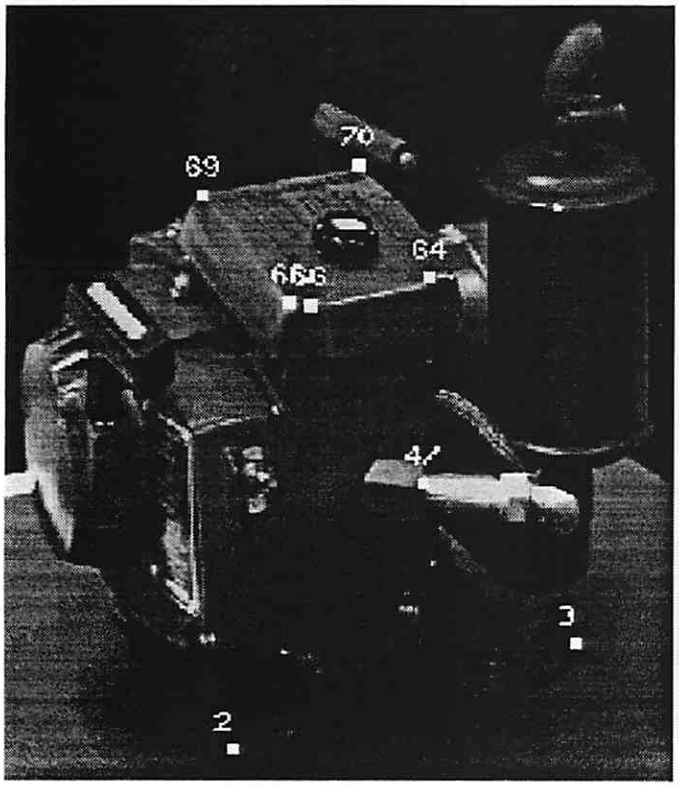


Template	Error
2	2
6	1
41	0
47	2
63	0
66	0
69	0
70	0

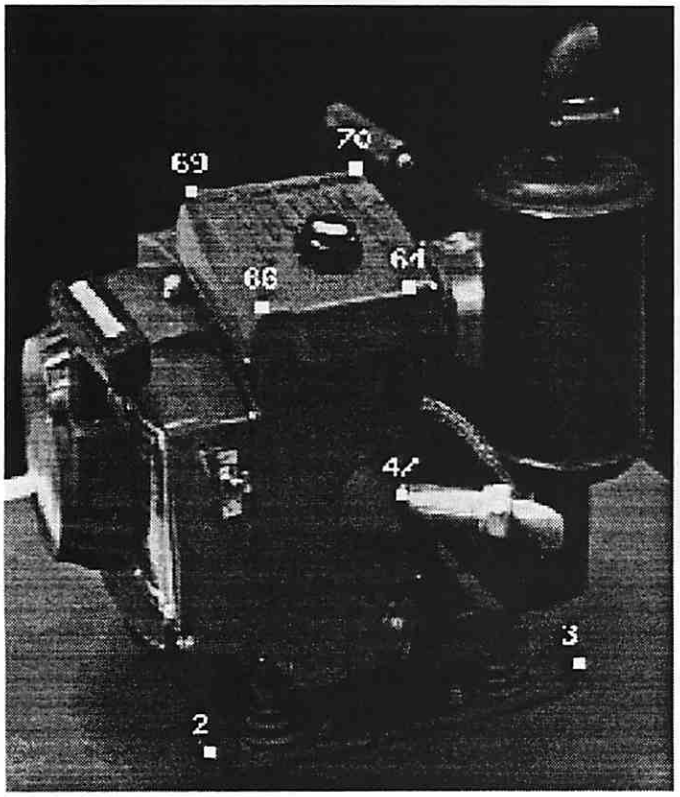


Template	Error
2	2
3	1
47	0
-	-
64	1
66	3
69	0
70	1

Figure 10: Performance under least median squares



Template	Error
2	1
3	1
47	0
-	-
64	0
66	10
69	0
70	1



Template	Error
2	1
3	1
47	0
-	-
64	0
66	0
69	0
70	0

Figure 11: Performance under least median squares(contd.)

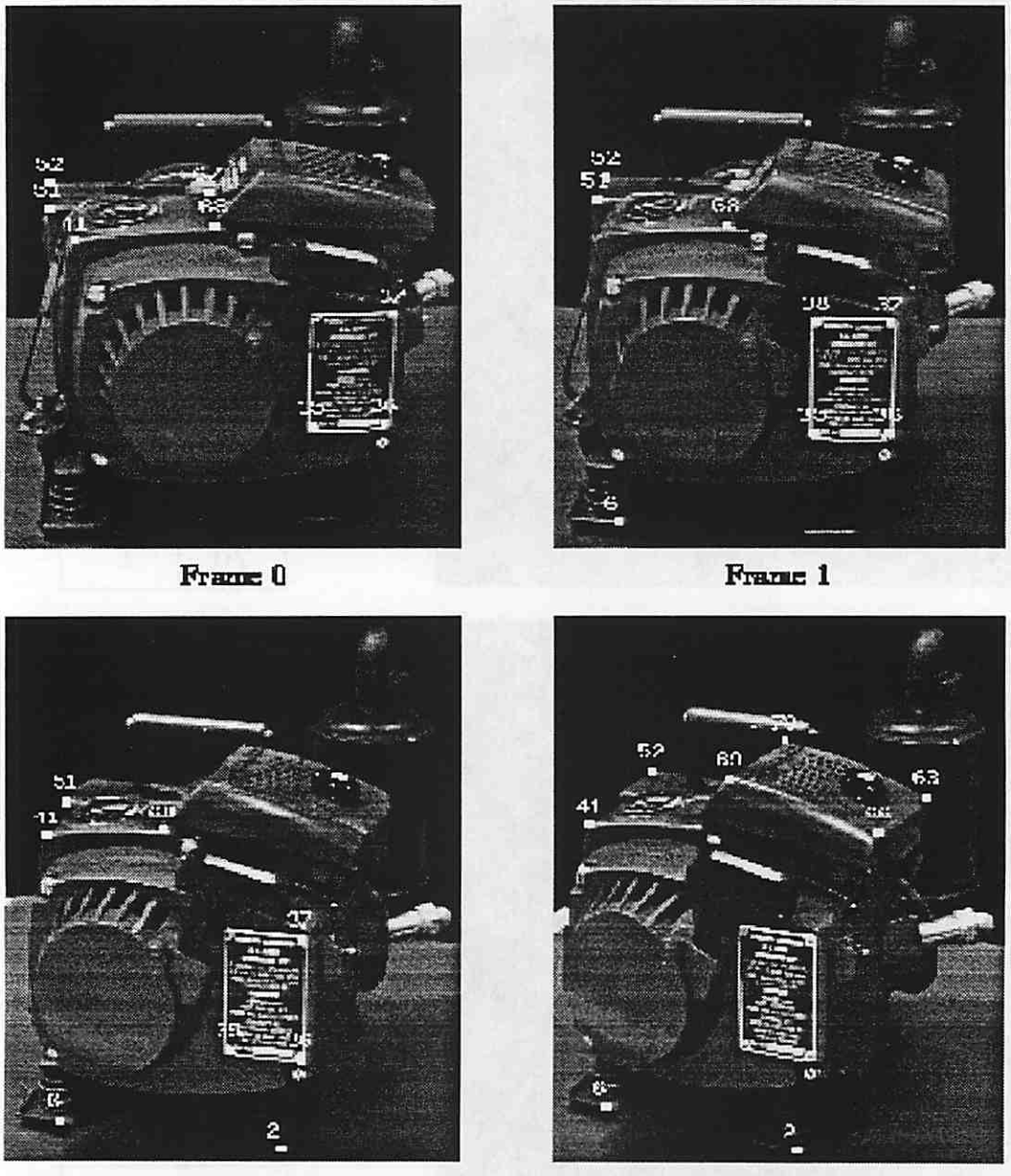
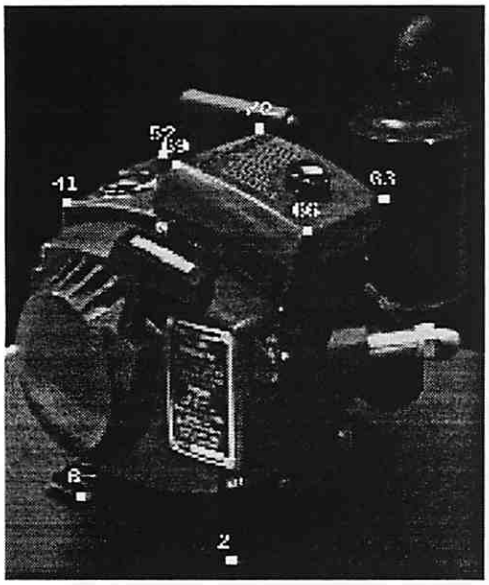
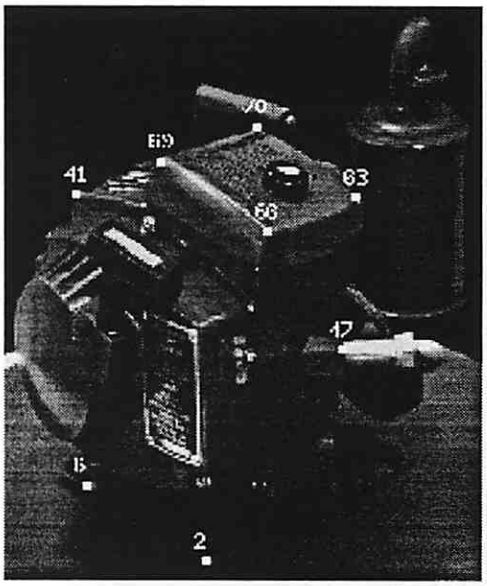


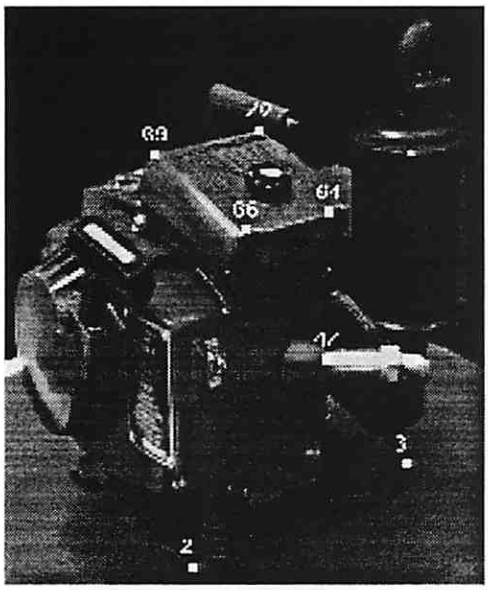
Figure 12: Registration under a 180° rotation



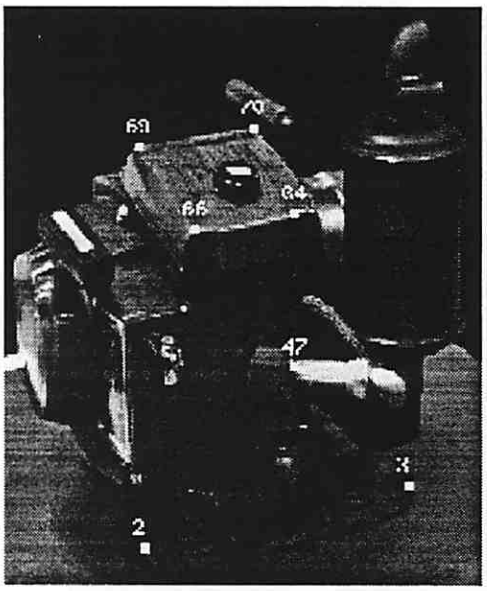
Frame 4



Frame 5

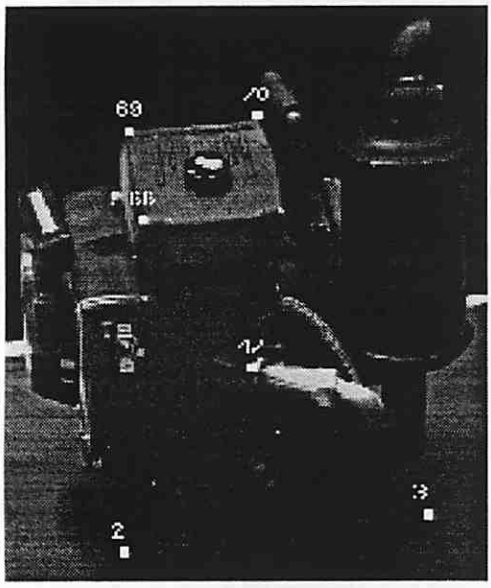


Frame 6

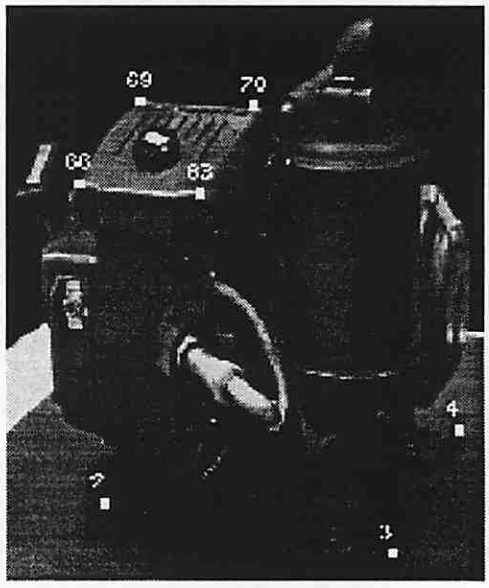


Frame 7

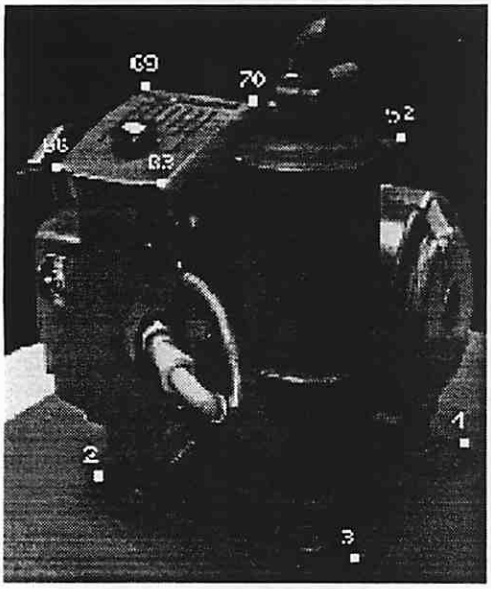
Figure 13: Registration under a 180° rotation(contd.)



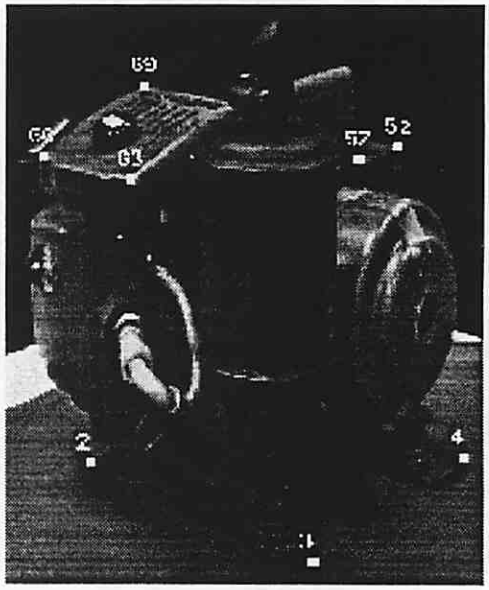
Frame 8



Frame 9

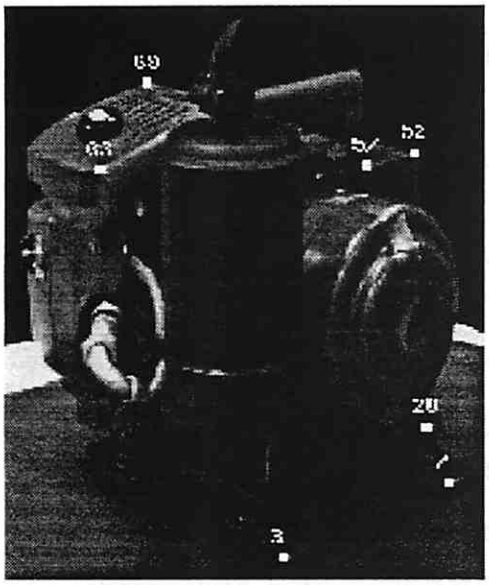


Frame 10

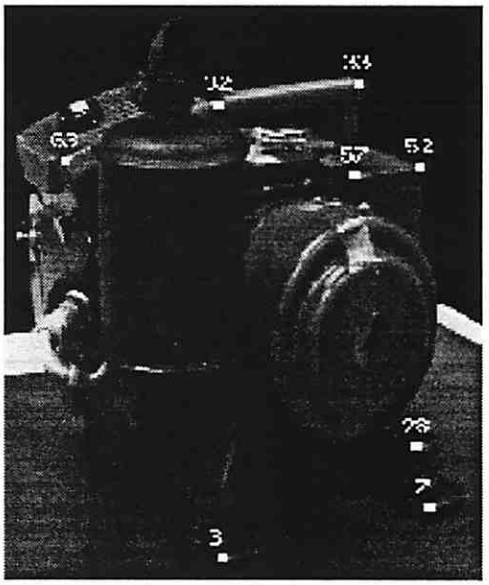


Frame 11

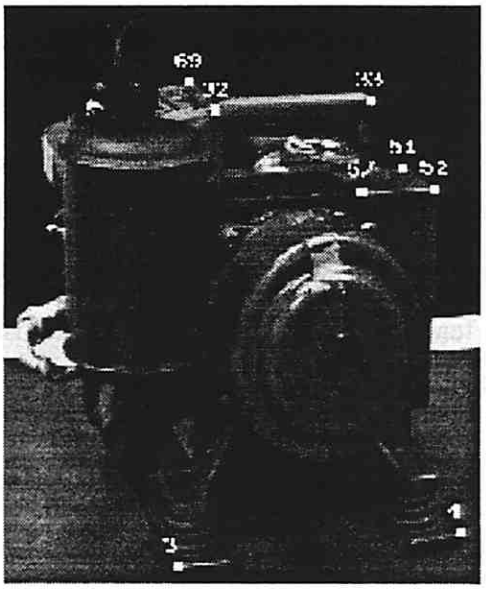
Figure 14: Registration under a 180° rotation(contd.)



Frame 12



Frame 13



Frame 14

Figure 15: Registration under a 180° rotation(contd.)

Time in Milliseconds for pose computation

Set size	11	8	6	4
Time	15	8.2	6.3	4

Time in Milliseconds per template

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

Search Window Size

All tests were conducted on a Sparc-2

Figure 16: Execution times for Tracking and Pose on a host in Milliseconds

tering style prediction have 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 it may therefore not be possible to use 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. 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 wish to thank Prof. Edward Riseman and Prof. Allen Hanson for their continued encouragement and guidance during the development of the registration system. The help of Bob Heller and the Robotics laboratory at UMass is acknowledged for systems support.

References

[1] Haralick, R. M. and Joo, H., "2D-3D pose estimation," *Proc. 9th IEEE Int. Conf. on Pattern Recognittion,* Nov 1988, pp. 385-391.

[2] 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.

[3] Horn, B. K. P., "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. of America,* vol 4, 1987, pp. 629-642.

[4] 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).

[5] 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.

[6] Z. Gigus and J. Malik. "Computing the aspect graph for line drawings of polyhedral objects," *Proc. Computer Vision and Pattern Recognition Conf.* 654-661.

[7] 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).

[8] J.J. Koenderink and A.J. van Doorn. "The Singularities of the Visual Mapping", *Biological Cybernetics* 24:51-59 (1976).

[9] H. Plantinga and C. Dyer. "Visibility, occlusion and the Aspect Graph," *Int. Journal of Computer Vision.*

[10] J. Ponce and D. Kriegman. "Computing exact aspect graphs of curved objects: parametric surfaces" *Proc. 8th National Conf. on Artificial Intelligence,* 1074-1079.

[11] J. Stewman and K. Bowyer. "Aspect graphs for planar-faced convex objects," *Proc. IEEE Workshop on Computer Vision,* 123-130.

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

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.

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

[21] 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.

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

[23] Cipolla, R. and Blake, A., " Surface Shape from the deformation of Apparent Contours", *Int. J. Comput. Vision*, 9(2):83-112, 1992.

[24] 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.

[25] Linnainma, S., D. Harwood and L.S. Davis, "Pose determination of a three-dimensional object using traingle pairs," *IEEE Transactions on Pattern Analysis and Machine Vision,* Vol 10, No 5, pp 634-647, 1988.

[26] Dhome, M., M. Richetin, J.T. Lapreste and G. Rives, "Determination of the attitude of 3-D objects from a single perspective view," *IEEE Transactions on Pattern Analysis and Machine Vision,* Vol 11, No 12, 1989.

[27] 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.

[28] 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