Presented at the 1994 IEEE International Conference on Systems, Man, and Cybernetics, Oct. 2-5, San Antonio, TX

The Minnesota Robotic Visual Tracker: A Flexible Testbed for Vision-Guided Robotic Research

Scott A. Brandt Christopher E. Smith Nikolaos P. Papanikolopoulos
Artificial Intelligence, Robotics, and Vision Laboratory
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455

Abstract

Most early research in robotic visual tracking, especially prior to 1990, separated the vision processing and robot control aspects of the system. Recent attempts to solve the problem close the control loop by incorporating the output of the vision processing as an input to the control subsystem. The Controlled Active Vision framework describes one such approach wherein dynamic target, camera, and environmental factors are incorporated via adaptive controllers that utilize the Sum-of-Squared Differences (SSD) optical flow measurements as an input to the control loop. This paper describes recent work at the University of Minnesota's Artificial Intelligence, Robotics, and Vision Laboratory in developing the Minnesota Robotic Visual Tracker (MRVT), a Controlled Active Vision robotic testbed. In addition, enhancements to the basic SSD algorithm are presented that produce order-of-magnitude improvements over previously reported results.

1 Introduction

Fast and simultaneously accurate tracking is critical for many real-time and near-real-time robotic and image processing applications. While much work has been done on the image processing and image understanding aspects of this problem, relatively little work has been done to truly integrate the vision processing with robot control. Typical approaches implement the "look-and-move" paradigm [1][2][11][14][22], in which the vision and control portions of the system are separate. A static image is processed, the manipulator is moved, another image is captured and processed, etc. In addition, many approaches use detailed *a priori* knowledge of the camera model, the environment, and the target to supplement the vision processing [4][13].

Recent solutions to the problem involve closing the control loop by incorporating the output of the vision processing as an input to the control subsystem [8][17]. One example of this type of approach is the Controlled Active Vision framework [17]. Instead of heavily relying on *a priori* information, this framework provides the necessary flexibility to operate under dynamic conditions where many environmental and target-related factors are unknown and possibly changing. The Controlled Active Vision framework is based upon adaptive controllers that utilize Sum-of-Squared Differences (SSD) optical flow measurements [3] as an input to the control loop. The SSD algorithm is used to measure the displacements of feature points in a sequence of images where the displacements may be induced by manipulator motion, target motion, or both. These measured displacements are then used as an input into the robot controllers, thus closing the control loop.

This paper describes the University of Minnesota Robotic Visual Tracker (MRVT), a complete robotic visual servoing testbed, and our implementation of a Controlled Active Vision system. The MRVT provides a flexible environment for the implementation of visual servoing tasks such as target tracking and recovery of depth (see [18] for a discussion of depth recovery). In developing the system, we enhanced and significantly extended the vision subsystem over similar previous implementations, achieving order-of-magnitude improvements in processing speed and system performance over previously reported results.

In this paper, we first briefly discuss previous work in the area. Then, we discuss the computation of the visual measurements used in the tracking system, including an enhanced SSD surface construction strategy and several optimizations. Finally, we describe the MRVT hardware and software architecture and present results from tracking experiments using the MRVT.

2 Previous Work

Research into robotic visual tracking began in the early 1980's. Initial research efforts in this area assumed a static camera (as compared with one mounted on the robot itself) [2][10] and performed only simulations [23]. Almost no work took into account the dynamics of the robot in computing the control signal. Since then, various researchers have expanded the problem to incorporate robot dynamics, moving camera models, or both, and several have focused on using vision information in the dynamic feedback loop [6][7][21]. Koivo and Houshangi [12] presented an adaptive method of visual servoing using a static sensor. Hunt and Sanderson [11] described robotic visual tracking techniques applicable to slow-moving objects wherein the centroid of the object is determined. Feddema [8][9] concentrated on the control aspects of the problem. Papanikolopoulos [16][17] developed the Controlled Active Vision approach that forms the basis for the work described in the rest of this paper.

3 Visual Measurements

The visual measurements are based upon a simple camera model and a measure of optical flow in a sequence of images. They are combined with search-specific optimizations and a dynamic pyramiding technique in order to optimize the performance of the system both in terms of the speed of the vision subsystem and the maximum tracking rate achieved.

3.1 Optical Flow Calculations

We use a matching-based technique known as the Sum-of-Squared Differences (SSD) optical flow [3]. For the point $\mathbf{p}(k-1) = (x(k-1), y(k-1))^T$ in the image $(k-1)$ where k denotes the kth image in a sequence of images, we want to find the point $\mathbf{p}(k) = (x(k-1)+u, y(k-1)+v)^T$. This point $\mathbf{p}(k)$ is the new position of the projection of the feature point **P** in image (k). We assume that the intensity values in the neighborhood F of **p** remain relatively constant over the sequence k. We also assume that for a given k, $\mathbf{p}(k)$ can be found in an area A about $\mathbf{p}(k-1)$ and that the velocities are normalized by time T to get the displacements. Thus, for the point $\mathbf{p}(k-1)$, the SSD algorithm selects the displacement $\Delta\mathbf{x} = (u, v)^T$ that minimizes the SSD measure

$$SSD(\mathbf{p}(k-1), \Delta\mathbf{x}) = \sum_{m,\, n \in F} [I_{k-1}x(k-1) + m, y(k-1) + n - I_k(x(k-1) + m + u, y(k-1) + n + v)]^2 \quad (1)$$

where $u,v \in A$, F is the neighborhood of **p**, m and n are indices for pixels in F, and I_{k-1} and I_k are the intensity functions in images $(k-1)$ and (k).

Feature points are selected using the SSD measure combined with an auto-correlation technique to produce an SSD surface corresponding to an auto-correlation in the neighborhood A [3] [17].

The size of the window F must be carefully selected to ensure proper system performance. Too small an F may provide insufficient information for accurate tracking while too large an F increases the associated computational overhead and enhances the background. An algorithm based upon the SSD technique may also fail due to too large a latency in the system or displacements resulting from motions of the object that are too large for the method to accurately capture. The search optimizations and dynamic pyramiding techniques are introduced to counter these latter concerns.

3.2 Search Optimizations

The primary source of latency in a vision system that uses the SSD measure is the time needed to identify $(u, v)^T$ in equation (1). To find the true minimum, the SSD measure must be calculated over each possible $(u, v)^T$. The time required to produce an SSD surface and to find the minimum can be greatly reduced by employing several schemes that, when combined, reduce the search time significantly.

The first optimization used is loop short-circuiting. During the search for the minimum on the SSD surface (the search for $u_{\min}$ and $v_{\min}$), each SSD measure must be calculated according to equation (1). This requires nested loops for the m and n indices. During the execution of these loops, the SSD measure is calculated as the running sum of the squared pixel value differences. If the running sum is checked against the current SSD minimum as a condition on these loops, the execution of the loops can be short-circuited as soon as the running sum exceeds the current minimum. This optimization has a worst-case performance equivalent to the original algorithm plus the time required for the additional condition tests. On average, this optimization decreases execution time by a factor of two.

The second optimization is based upon the heuristic that the best place to begin the search for the minimum is at the point where the minimum was last found on the surface and to expand the search radially from that point. This heuristic works well when the disturbances being measured are relatively regular. In the case of target tracking, this corresponds to targets that have locally smooth velocity and acceleration curves. If a target's motion does not exhibit such relatively smooth curves, then the target is untrackable due to the inherent latency in the video equipment and the vision processing system. Furthermore, the manipulator trajectory required to track such a target would likely be mechanically unachievable by the system.

Under this heuristic, the search pattern in the (k) image is altered to begin at the point on the SSD surface where the minimum was located for the $(k-1)$ image. The search pattern then spirals out from this point, searching over the extent of u and v. This is in contrast with the typical indexed search pattern where the search begins at the desired feature location and indices are incremented in a row-major scan fashion. Figure 1 contrasts a traditional row-major scan with the spiral scan where the center position in each corresponds to the position where the minimum was last found. This search strategy may also be combined with a predictive controller to begin the search for the SSD minimum at the position that the controller indicates is the most likely location of the minimum. Since the structure that implements the spiral search pattern contains no more overhead than the loop structure of the traditional search, worst-case performance is identical. When combined with loop short-circuiting, this optimization produces execution times 8-10 times better than initial results.

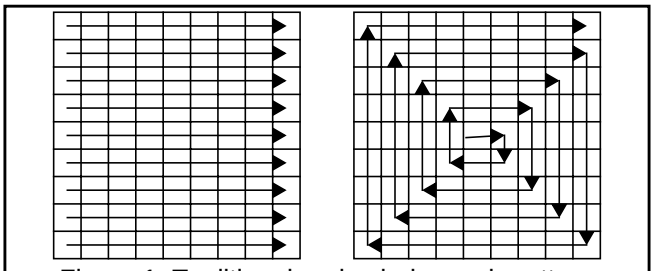

Figure 1: Traditional and spiral search pattern

The third optimization arose from the observation that search times for feature points varied significantly, by as much as 100%, depending upon the shape/orientation of the feature. In determining the cause of the problem and exploring possible solutions, we realized that by applying the spiral image traversal pattern to the calculation of the SSD measure we could simultaneously fix the problem and achieve additional performance improvements. Spiraling the calculation of the SSD measure yields best-case performance that is independent from the orientation of the target by changing the order of the SSD calculations to no longer favor one portion of the image over another. In the traditional calculation pattern (a row-major traversal of the feature region) information in the upper portion of the region is used before that in the lower portion of the region, thus skewing the timing in favor of those images where the target or the foreground portion of the feature being tracked appears in the upper half of the region. Additional speed gains are achieved because the area of greatest change in the SSD measure calculations typically occurs near the center of the feature window, which generally coincides with an edge or a corner of the target, resulting in fewer calculations before the loop is terminated in non-matching cases. Speed gains from this optimization are approximately 40%.

When combined, these three optimizations (the loop short-circuiting, the spiral search pattern, and the spiral SSD calculation) interact cooperatively to find the minimum of the SSD surface as much as 17 times faster on average than the unmodified search.

Experimentally, the search times for the unmodified algorithm averaged 136 msec over 5000 frames under a variety of relative feature point motions. The modified algorithm with the search-loop short-circuiting alone averaged 60-72 msec search times over several thousand frames with arbitrary relative feature point motion. The combined short-circuit/spiral search algorithm produced search times that averaged 13 msec under similar tests and the combined short-circuit, dual-spiral algorithm produced search times that averaged 8 msec. Together these optimizations allow the vision system to track three to four features at RS-170 video rates (33 msec per frame) without video under-sampling.

3.3 Dynamic Pyramiding

Dynamic pyramiding is a heuristic technique that attempts to resolve the conflict between accurate positioning of a manipulator and high-speed tracking when the displacements of the feature points are large. Previous applications have typically depended upon one preset level of pyramiding (via subsampling) to enhance either the top tracking speed of the manipulator or the positioning accuracy of the end-effector above the target [17].

In contrast, dynamic pyramiding uses multiple levels of pyramiding (see Figure 2). The level of the pyramiding is selected based upon the observed displacements of the target's feature points. If the displacements are small relative to the search area, the pyramiding level is reduced; if the displacements are large compared to the search area, the pyramiding level is increased. This results in a system that enhances the tracking speed when required, but is always biased in favor of the maximum accuracy achievable. The dynamic level switching thus allows the tracker to effectively adjust to accelerations and decelerations of the target (when the feature locations diverge from the expected) and then to increase accuracy when the tracking adapts to the new, higher or lower speed or when the target is at rest.

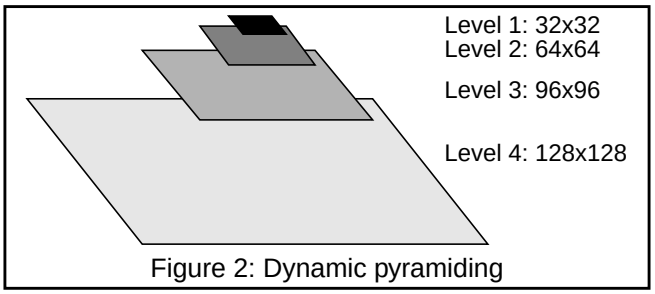

Figure 2: Dynamic pyramiding

During the search process, the SSD measurements are centered upon particular pixels in the pyramided search area A. The size of the search area and which pixel positions are selected ($\Delta \boldsymbol{x} = (u, v)^{\mathrm{T}}$ in equation (1)) is dependent upon which of the four levels of pyramiding is currently active. The lowest level selects every pixel in a square 32×32 pixel region of the current frame. Subsequent levels increase the search and feature areas to cover a larger percentage of the image, but subsample the images to achieve the same number of calculations (and consequently the same processing time). Therefore, the second pyramid level searches every other pixel in a 64×64 pixel region, the third pyramid level searches every third pixel in a 96×96 pixel region, etc. The current implementation uses a total of four pyramid levels.

4 The Minnesota Robotic Visual Tracker

The MRVT, shown in Figure 3, is a multi-architectural system that consists of two main parts: the Robot/Control Subsystem (RCS) and the Vision Processing Subsystem (VPS).

The RCS is comprised of a PUMA 560 manipulator, its associated Unimate Computer/Controller, and a VME-based Single Board Computer (SBC). The manipulator's trajectory is controlled via the Unimate controller's Alter line and

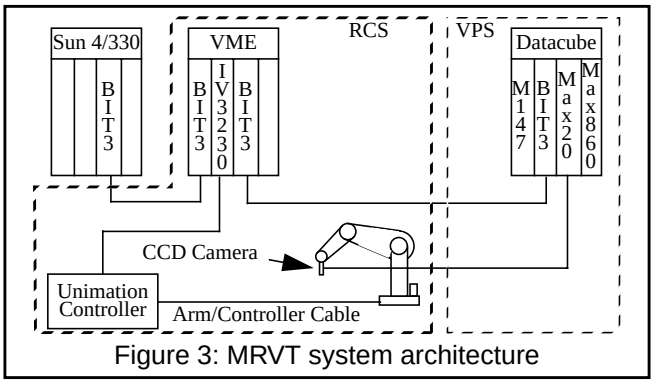

Figure 3: MRVT system architecture

requires path control updates once every 28 msec. Those updates are provided by an Ironics 68030 VME SBC running Carnegie Mellon University's CHIMERA real-time robotic operating system [20]. A Sun SparcStation 330 serves as the CHIMERA host and shares its VME bus with the Ironics SBC via BIT-3 VME-to-VME bus extenders. BIT-3 extenders are also used to allow shared-memory communications between the RCS and the VPS.

The VPS receives input from a Panasonic GP-KS102 miniature camera that is mounted parallel to the end-effector of the PUMA and provides a video signal to the Datacube system for processing. The Datacube system is the main component of the VPS and consists of a Motorola MVME-147 SBC running OS-9, a Datacube MaxVideo20 video processor, a Datacube Max860 vector processor, and a BIT-3 VME-to-VME bus extender. The bus extender allows the VPS and the RCS to communicate via shared memory, eliminating the need for slower and more complex serial communication. The VPS performs the optical flow, calculates the desired control input, and supplies the input vector via shared memory to the Ironics processor for inclusion as an input into the control software. The image processing is performed under a pipeline programming model using Datacube's ImageFlow libraries. The calculations required to produce the desired control input are executed on the 147 SBC and transferred to the RCS via the BIT-3 shared memory using a simple handshake technique for synchronization.

5 Visual Tracking Results

We conducted multiple experimental runs for the tracking of objects that exhibited unknown 2-D translational motion with a coarse estimate of the depth of the objects. The targets for these runs included books, batons, and a computer-generated target displayed on a large video monitor. During the experiments, we tested the system using targets with linear and curved trajectories.

The first experiments were conducted using a book and a baton (see Figure 4) as targets in order to test the performance of the system both with and without the dynamic pyramiding. These initial experiments served to confirm the feasibility of performing real-time pyramid level switching and to collect data on the performance of the system under the pyramiding and search optimizing schemes. Figure 5

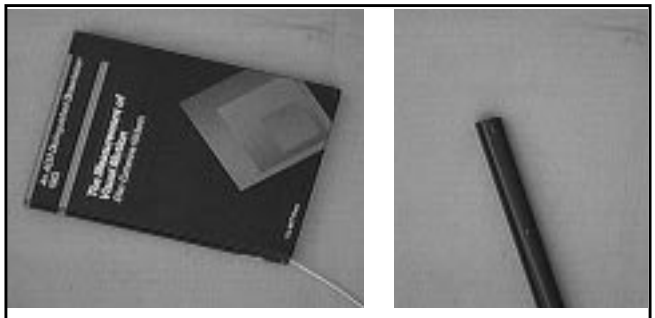

Figure 4: Example targets

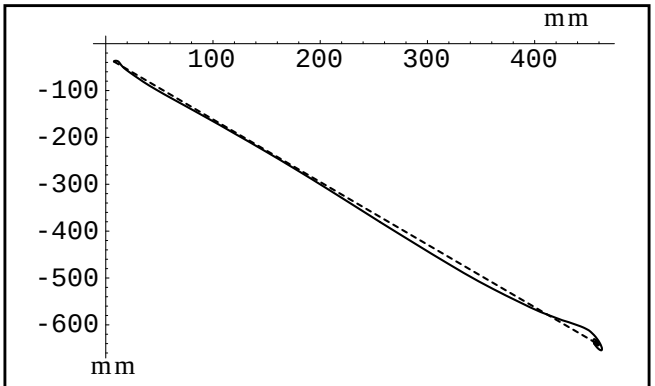

Figure 5: Target and manipulator paths

shows the target trajectory and the manipulator path from one of these experiments. With the pyramiding and search optimizations, the system was able to track the corner of a book moving along a linear trajectory at 80 cm/sec (see Figure 6). In contrast, the maximum speed reported by Papanikolopoulos [17] was 7 cm/sec, representing an order of magnitude increase in tracking speed. The dashed line is the target trajectory and the solid line represents the manipulator trajectory. The start of the experiment is in the upper left corner and the end is where the target and manipulator trajectories come together in the lower right. The oscillation at the beginning is due to latency in the system and the switch to higher pyramiding levels as the system compensates for the acceleration of the target and the oscillations at the end are produced when the target slows to a stop and the pyramiding drops down to the lowest level. This results in the manipulator centering above the feature accurately due to the higher resolution available at the lowest pyramid level. The tracking inaccuracies in the middle of the plot are due to inaccuracies caused by the pyramiding, which loses data at the higher levels. This results in poorer, but faster, tracking at higher speeds and accelerations.

Once system performance met our minimal requirements, testing proceeded to a computer-generated target where the trajectory and speed could be accurately controlled. For these experiments, the setup of the PUMA and the video monitor to display the targets is shown in Figure 7. The tar-

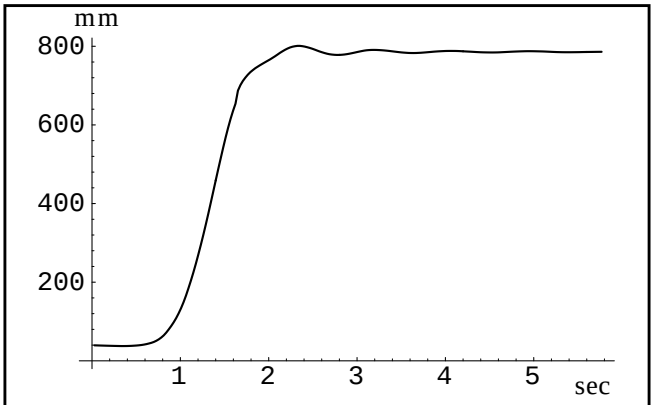

Figure 6: Distance vs. time

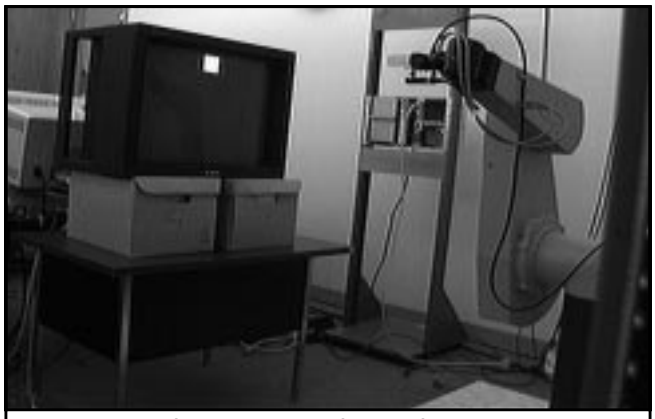

Figure 7: Experimental setup

get (the white box) traced either a square or a circle on the video monitor while the PUMA tracked the target.

The first experiments traced a square and demonstrated robust tracking in spite of the target exhibiting infinite accelerations (at initial start-up) and discontinuities in the velocity curves (at the corners of the square). The results in Figure 8 show the oscillatory nature of the manipulator trajectory at the points where the velocity curve of the target is discontinuous. The speed of the target in these experiments approached 15 cm/sec.

The next set of experiments was performed using a target that traced a circular path on the video monitor. This path produces a smoothly varying velocity curve while retaining the infinite acceleration of the target at start-up. We allowed the experiments to continue for many loops of the target in order to determine the response of the system over time. The results demonstrated an initial oscillation due to the acceleration at start-up followed by smooth tracking once the controller compensated for the extreme initial disturbances. The results of the circular tracking are shown in Figure 9. The multiple manipulator paths demonstrate the repeatability achieved using the controller, algorithms, and optimizations. The target in this experiment moved with a speed of 35 cm/sec. The oval-shaped manipulator trajectory visible in the figure is caused by start-up conditions in the system. The last three loops of the manipulator plotted in Figure 10 show the nearly circular trajectory achieved once the system has stabilized.

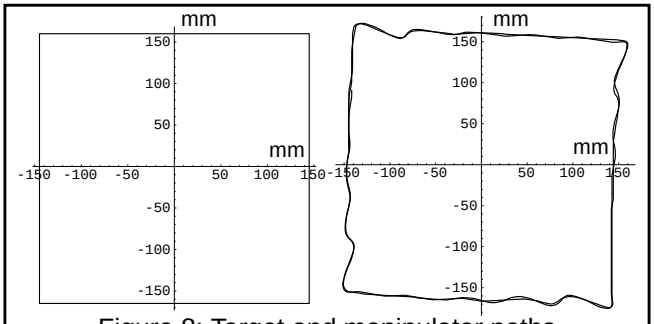

Figure 8: Target and manipulator paths

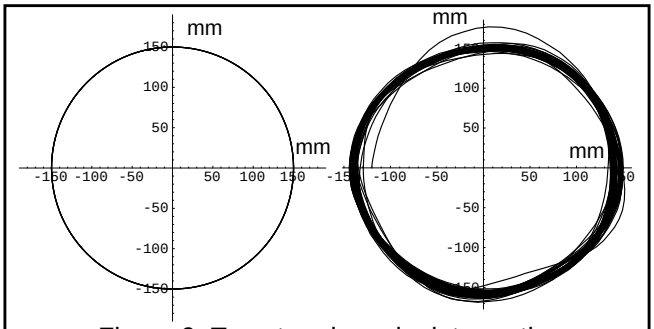

Figure 9: Target and manipulator paths

6 Future Work

There are several areas where this work can be extended to improve the performance of the system and the range of possible applications. One such area involves the enhancement of the robot controller to improve the predictive aspect of the tracking system. Better predictions of the feature locations will enable faster, more robust tracking by helping the vision processing subsystem find the minimal SSD measures sooner (thus enhancing the gains from the loop short-circuiting) and by allowing the system track at a lower pyramid level (thus improving the tracking precision). In addition, further experimentation is needed both with controllers we have implemented to track objects exhibiting 2-D translational motion with rotation and with planned controllers that track 3-D translational motion and full six degree-of-freedom motion.

Other areas for improvement exist in the vision portion of the system. One such area involves the selection of feature points. In some cases features may be lost above certain pyramid levels. Automatic evaluation of the feature points can screen feature points for this property and either disqualify these points for selection or flag them with a maximum pyramid level for use in actual tracking. A similar issue is the size of the feature window. The current implementation uses a fixed window size (an issue discussed in [15]), but variable sized windows selected on a feature by feature basis may allow for more robust tracking.

One possible improvement to the pyramiding is in the area of accuracy at higher pyramid levels. The current implementation trades accuracy for tracking speed by processing

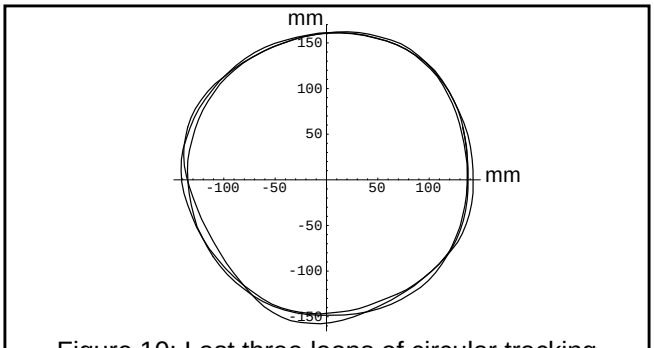

Figure 10: Last three loops of circular tracking

at a higher pyramid level when the target is moving fast or accelerating. By doing a full-resolution search in a small area around the best-match at higher pyramid levels, it should be possible to locate the target with a high degree of accuracy for very little extra computational expense.

7 Conclusion

The Controlled Active Vision [17] framework provides a robust yet flexible method for integrating vision processing with robot control. The Minnesota Robotic Visual Tracker supports the implementation of prototype applications using the Controlled Active Vision framework. One such application is the robotic visual tracker presented in this paper.

Several optimization techniques were presented that greatly enhance the performance of the tracker. Dynamic pyramiding, a technique for increasing the overall tracking speed of the system, results in order-of-magnitude speed improvements over results previously reported by Papanikolopoulos [17]. In addition, several search and SSD optimizations were presented that result in considerable improvements in the vision subsystem speed. Search spiraling and one-frame predictive filtering, combined with loop short-circuiting, allow the system to rapidly find minimal SSD measures (indicating a close match to the feature template) and limit the number of calculations. Spiraling of the SSD measure calculations results in independence of the processing speed from feature shape and/or orientation and yields further improvements in processing speed.

8 Acknowledgments

This work has been supported by the Department of Energy (Sandia National Laboratories) through Contract #AC-3752D, the Center for Transportation Studies through Contract #USDOT/DTRS 93-G-0017-1, the 3M Corporation, the Graduate School of the University of Minnesota, and the Department of Computer Science of the University of Minnesota.

9 References

[1] P.K. Allen, "Real-time motion tracking using spatio-temporal filters," *Proceedings, DARPA Image Understanding Workshop*, 659-701, 1989.

[2] P.K. Allen, B. Yoshimi, and A. Timcenko, "Real-time visual servoing," *Proceedings of the IEEE International Conference on Robotics and Automation*, 851-856, April, 1991.

[3] P. Anandan, "Measuring visual motion from image sequences," Technical Report COINS-TR-87-21, COINS Department, University of Massachusetts, 1987.

[4] B. Bhanu, P. Symosek, J. Ming, W. Burger, H. Nasr, and J. Kim, "Qualitative target motion detection and tracking," *Proceedings of the DARPA Image Understanding Workshop*, 370-389, 1989.

[5] S. Brandt, "Enhanced robotic visual tracking under the controlled active vision framework," Master's Thesis, Department of Computer Science, University of Minnesota, 1993.

[6] A. Castano and S. Hutchinson, "Hybrid vision/position servo control of a robotic manipulator," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1264-1269, May, 1992.

[7] F. Chaumette and P. Rives, "Vision-based control for robotic tasks," *Proceedings of the IEEE International Workshop on Intelligent Motion Control*, 395-400, August 1990.

[8] J.T. Feddema and C.S.G. Lee, "Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera," *IEEE Transactions on Systems, Man, and Cybernetics*, 20(5):1172-1183, 1990.

[9] J.T. Feddema and O.R. Mitchell, "Vision guided servoing with feature-based trajectory generation," *IEEE Transactions on Robotics and Automation*, 5(5):691-699, 1989.

[10] D.B. Gennery, "Tracking known three-dimensional objects," *Proceedings of the AAAI 2nd National Conference on AI*, 13-17, 1982.

[11] A.E. Hunt and A.C. Sanderson, "Vision-based predictive tracking of a moving target," Technical Report CMU-RI-TR-82-15, Carnegie Mellon University, The Robotics Institute, January, 1982.

[12] A.J. Koivo and N. Houshangi, "Real-time vision feedback for servoing of a robotic manipulator with self-tuning controller," *IEEE Transactions on Systems, Man and Cybernetics*, 21(1):134-142, 1991.

[13] D. Lowe, "Fitting parameterized 3-D models to images," Technical Report 89-26, University of British Columbia, 1989.

[14] R.C. Luo, R.E. Mullen Jr., and D.E. Wessel, "An adaptive robotic tracking system using optical flow," *Proceedings of the IEEE International Conference on Robotics and Automation*, 568-573, 1988.

[15] M. Okutomi and T. Kanade, "A locally adaptive window for signal matching," *International Journal of Computer Vision*, 7(2):143-162, 1992.

[16] N. Papanikolopoulos, P.K. Khosla, and T. Kanade, "Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision," *IEEE Transactions on Robotics and Automation*, 9(1):14-35, 1993.

[17] N. Papanikolopoulos, "Controlled active vision," Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1992.

[18] C. Smith and N. Papanikolopoulos, "Computation of shape through controlled active exploration," *Proceedings of the IEEE International Conference on Robotics and Automation*, 2516-2521, 1994.

[19] C. Smith, S. Brandt, and N. Papanikolopoulos, "Controlled active exploration of uncalibrated environments," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 792-795, 1994.

[20] D.B. Stewart, D.E.Schmitz, and P.K. Khosla, "CHIMERA II: A real-time multiprocessing environment for sensor-based robot control," *Proceedings of the Fourth International Symposium on Intelligent Control*, 265-271, September, 1989.

[21] M.M. Trivedi, C. Chen, and S.B. Marapane, "A vision system for robotic inspection and manipulation," *Computer*, 22(6):91-97, June, 1989.

[22] D. Tsakiris, "Visual tracking strategies," Master's Thesis, Department of Electrical Engineering, University of Maryland, 1988.

[23] L.E. Weiss, A.C. Sanderson, and C.P. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE Journal of Robotics and Automation*, RA-3(5):404-417, October, 1987.