Experiments With Driving Modes for Urban Robots

M. Hebert, R. MacLachlan, P. Chang
The Robotics Institute
Carnegie Mellon University
Pittsburgh PA 15213

ABSTRACT

In this paper, we describe experiments on semi-autonomous control of a small urban robot. Three driving modes allow semi-autonomous control of the robot through video imagery, or by using partial maps of the environment.
Performance is analyzed in terms of maximum speed, terrain roughness, environmental conditions, and ease of control. We concentrate the discussion on a driving mode based on visual servoing. In this mode, a template designated in an image is tracked as the robot moves toward the destination designated by the operator. Particular attention is given to the robustness of the tracking with respect to template selection, computational resources, occlusions, and rough motion. The discussion of algorithm performance is based on experiments conducted at Ft. Sam Houston, TX, on Jul. 5-9 1999. In addition to the driving modes themselves, the performance and practicality of an omnidirectional imaging sensor is discussed. In particular, we discuss the typical imaging artifacts due to ambient lighting.

1. INTRODUCTION

Research in mobile robotics has primarily focused thus far on either small robots for operation in mild indoor environments, or on larger systems, i.e., non-portable, for operation in unstructured terrain. Recent advances in computing hardware, sensing, and algorithms have enabled the development of systems that are small and affordable, yet able to operate in difficult environments and to conduct complex missions. In particular, DARPA initiated recently the Tactical Mobile Robot Robotics (TMR) program with the objective of developing small, backpackable robots capable of semi-autonomous operation in unstructured urban environments [2]. In paper we describe components of one such system being developed by a team composed of JPL, CMU, USC, Oakridge Natl. Lab. and IS Robotics. In this system, the user controls the robot by designating goals, either in an image of in a map, which the robot attempts to reach with little or no interaction with the user. The user has the option to use different designation modalities which correspond to different driving modes of the robot. There are two main challenges in designing those driving modes. First of all, they must be non-distracting in that the user must be required to input only minimal information; the details of the actual control of the robot must be handle transparently in an autonomous fashion. Second, the on-board driving system must be robust to the typical level of shock, vibration, and rapid change of attitude which are typical of a lightweight robot operating at high speed in an unstructured terrain.
We briefly describe below the three main semi-autonomous driving modes that were selected and the experimental environment that was used for evaluating the system. In the next section, we describe in detail one of the driving mode and its performance and limitations based on a recent series of experiments. The other components of the system such as obstacle avoidance and operator control interface are outside the scope of the paper.
The first driving mode is based on visual servoing. In that mode, the user selects a destination template in the current image, that is, an image taken from the current position of the robot. Given that template, the robot attempts to servo its path to the designated goal by tracking the selected template in the subsequent images. The tracking algorithm must be robust enough to deal with the disturbances caused by the terrain and by other entities competing for control of the robot, e.g., obstacle avoidance. It must be able to seamlessly recover from events in which the target feature is lost.
The second driving mode is “visual waypoint navigation”, in which the operator designates successive destination points in an image taken on-board the robot. The waypoints selected in the image are converted to physical points in the world by backprojection from the camera, assuming that the terrain is flat. Once those waypoints are sent to the robot, the on-board processor attempts to correct their positions based on its motion sensors as the robot drives toward the current estimate of the closest waypoint. As the robot drives closer to the waypoints, its estimate of their actual locations is refined. Eventually, the robot follows a path that brings it as close to the waypoints as possible. This mode is particularly effective when the operator can easily identify an approximate path to an area of interest in the images sent back from the robot. This driving mode is an extension of the system originally introduced by Kay [6].
For those two image-based driving modes, the operator must be able to select destinations any direction without being concerned with knowing the current orienation of the robot with respect to the desired direction. Also, the on-board tracking software must be able to retain target lock no matter what the orientation of the robot is. For those two reasons, it is highly

convenient to use an imaging sensor that can provide a full 360^o panorama of the robot's environment at all times. Such an omnidirectional camera is briefly described in the next section.

The third mode of driving uses a map of the environment that is incrementally built from the obstacle detection sensors. Given a destination specified by the user, the robot re-computes an optimal path to the goal at every time step, based on the new map information gathered from the sensors during that time step. The basic planning algorithm is based on technology originally developed for UGVs, as described in [10]. A challenge specific to small urban robots is the high level of uncertainty on robot position due to the unreliability of GPS in urban environments, and the notoriously inaccurate dead-reckoning on a skid-steer robot. Extensions to the planner are underway to deal with uncertain maps. The second aspect of the development of planning algorithms for urban robots is the integration of reasoning about threats in the environment and areas of risk.

The rest of the paper in devoted to describing the implementation of the visual servoing mode and its performance based on experiments recently conducted. The robot used in those experiments is the IS Robotics Urban Robot (Urbie) which is equipped with the electronic control box developed by JPL, which includes two Pentium 166MHz PC-104 boards for on-board computation. The maximum driving speed is 0.8m/s. In the system described here, all the image processing tasks reside on one processor, and the control and command arbitration tasks reside on the other processor. In this paper, we concentrate on the algorithms used for image-based, semi-autonomous control of the robot; a complete description of the overall system may be found in ??.

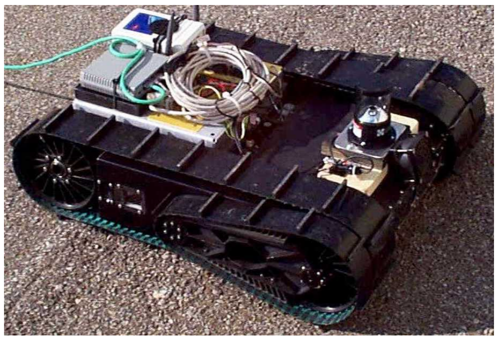


Figure 1 The ISR Urbie robot used in the experiments described in this paper – the CMU development version is shown here.

The results reported in this paper, including systems parameters and performance, were collected for the most part during a series of experiments at Ft. Sam Houston, TX, which is the demonstration site for the TMR project. Figure 2 shows two aerial views of the main building on this site. The typical paths used for testing semi-autonomous driving are also indicated in Figure 2; the paths were designed to simulate a mission in which the robot approaches the building prior to entry. This environment forced the system to be exposed to all the challenges of the urban environment, including large variation in lighting, harsh sunlight, obstacles such as curbs and rocks.

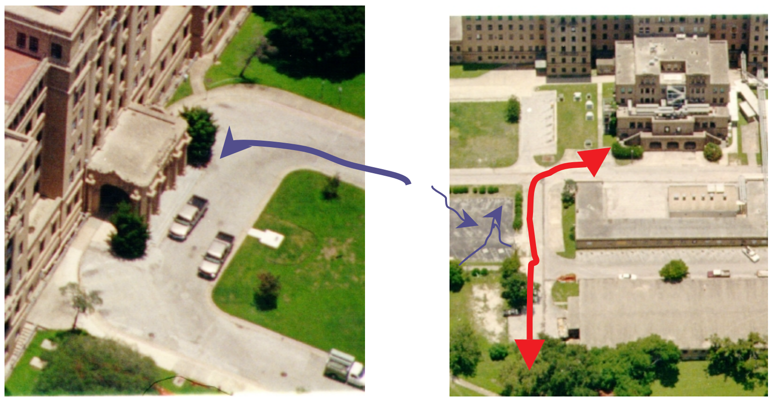


Figure 2 Typical areas used for experimentation at the Ft. Sam Houston test site.

2. VISUAL SERVOING OVERVIEW

Visual servoing has been used for mobile robot navigation in earlier systems using single or multiple templates [9,12]. The approach described here is based on the standard techniques for template tracking, with an emphasis on robustness to robot motion and taking advantage of the geometry of the omnidirectional camera.

Because the tracker should be able to track any target chosen by the user without any assumption about its appearance, only template based tracking methods are considered. These tracking methods are used to compare the template over a certain region of the input image to find the best match position and update this region as the new target. There is always a trade-off between robustness and speed of the overall tracking performance. In order to be more robust, the tracker needs to search over a larger region which causes the tracking time to increase. This trade-off also applies to the size of the template. Normally the sizes of the template and the search region are chosen empirically according to the specific purposes of the tasks.

We use a conventional SSD-based tracking algorithm to do the tracking over the omni-directional image. All affine motion parameters are computed using the SSD tracker. The disadvantage of SSD is that it only allows small variations between two neighboring templates which makes it prone to losing the target when the target moves too fast or when an occlusion happens. As a result, quantitative experiments must be carried out to identify the maximum motion speed that can be tracked, and strategies must be designed to cope with tracking at high speed and occlusion.

Given a template selected by the user in a virtual image in direction (φ,θ), a standard affine tracker is applied as long as the template remains inside the virtual image. The standard tracking, as described in [3] and [5], for example, assumes that the transformation between two templates at times t and $t+\Delta t$ is a small affine transformation represented by a matrix ΔA.

After linearization in ΔA, the motion parameters are recovered by least squares estimation. A major issue with this type of tracking is that it is limited to small motions (in fact, strictly speaking, a single iteration of the linearization/least-square estimate is limited to one pixel motion.) Even using multiple iterations of the estimator is insufficient for typical robot motion on rough terrain – all the results shown in this paper were obtained with ten iterations of the tracker.

In order to address this problem, we use a first step of normalized correlation search prior to applying the affine tracker. Although the correlation search cannot recover rotational and scaling distortions, it does provide an estimate of the translation component of the motion. This estimate is used for bootstrapping the iterations of the affine tracker. In practice, typically templates are 40x40 pixels and the correlation search is applied for translations in x and y of +/- 16 pixels. This allows the system to handle rotational motions of up to +/-5^o per frame. This step is, in practice, critical in order to apply the tracker to motion on rough terrain.

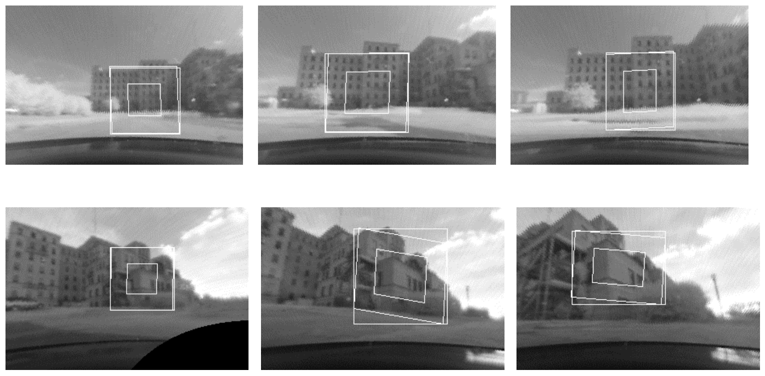


Figure 3 Two examples (top and bottom) of tracking targets for semi-autonomous driving at Ft. Sam Houston. The inner box shows the template; the outer box shows the region of interest extracted from the virtual image.

As mentioned above, it is highly convenient to use an imaging sensor that can provide a full 360^o panorama of the robot's environment at all times. For that purpose, an omnidirectional camera is used for all the tracking tasks in lieu of conventional cameras. This type of camera allows for continuous tracking in any arbitrary direction without mechanical control of camera heading and without any field view constraints. We now briefly describe the prototype camera used in the current system. We limit the description to the points that are necessary to understand the operation of the visual servoing algorithm; we refer the reader to the appropriate publications of the camera designers' for additional details.

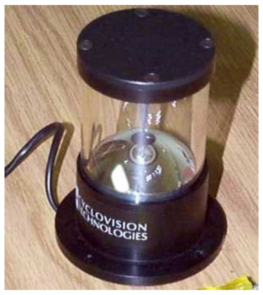
The omnidirectional cameras used in this work were developed at Columbia University by Shree Nayar and later by Cyclovision Inc. The version of the camera used in the experiments reported in this paper is the folded mirror camera described in [8]. The camera is shown in Figure 4 (a). We summarize in this section the geometry of the camera and the basic approach used for generating virtual perspective views from the camera. Since the details are given in the camera's designers' publications, we limit ourselves to the essential features necessary to understand the tracking algorithms.

The camera uses a parabolic mirror to take a 360 degree view which is projected in a disc. Details of the camera geometry are given in [6]. For our purposes, the geometry shown in Figure 4(b) is sufficient to understand its use in visual servoing and target designation. Rays from the scene pass through the focal point of the parabolic mirror. Their intersection with a parabola centered at the focal point is projected on the image plane to form the disc image. Given any arbitrary image plane *P*, the image that would be formed on *P* by the scene can be computed by a simple transformation of the points in the disc image *D*. We denote by $F_p(x_p)$ the transformation that associates a point x_s in the disc image to a point x_p in the reconstructed virtual perspective image. The exact definition of *F* and its properties are described in [7].

In the rest of the paper, we will use the following conventions: A virtual perspective view is defined by the viewing direction *V* at the center of the image and by a focal length *f*. *V* is defined by two angles: the azimuth angle θ, which specifies the orientation of the image with respect to the heading of the robot, and the altitude φ, which specifies the vertical angle of the viewing direction with respect to the horizontal plane of the robot . There remains one degree of freedom in order to fully specify the virtual image, that is, the rotation about the *V* direction. This rotation is constrained by always forcing the *x* axis (columns) of the virtual image to be parallel to the disc image plane.

Given a virtual image direction (φ,θ) and a new disc image, the new virtual image is computed by calculating $x_s = F_p(x_p)$ for each pixel x_p. Since the computation of $F_p(x_p)$ involves a few non-linear operations, it is more efficient to first compute a lookup table, denoted by $\mathrm{LUT}(\varphi,\theta)$, such that $x_s = \mathrm{LUT}(\varphi,\theta)[x_p]$ for all pixels in the virtual image. This implementation detail will become important when we discuss tracking across virtual views.

In all the examples shown in the paper, the virtual perspective images are 200x300 with $f = 175$ pixels, corresponding to a 90^o field of view.



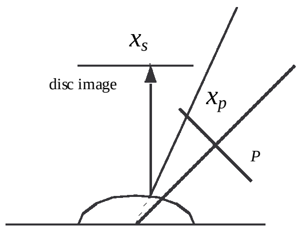


Figure 4 (a) The Cyclovision omnicam used in the experiments; (b) Simplified geometry of the camera.

3. ROBUSTNESS ISSUES AND PERFORMANCE

The baseline tracking algorithm described in the previous section is sufficient for operating in relatively mild conditions, i.e., relatively low speed, short distances, and smooth motion. In practical environments, however, a number of robustness issues need to be addressed. A first issue is that as the robot travel for long distances, the original image template may grow in size or change in shape substantially. In order for the tracker to operate over long distance, we need to develop a strategy for *updating target size*. A second issue is that the tracker described so far operates in a single virtual perspective image constructed from the raw omnicam image. As the vehicle turns or pitches, the image template may move outside of the field of view of the current virtual view. This requires an algorithm for *changing virtual views* without substantial degradation of tracking performance during the transition between views. Another critical robustness issue is the ability to recover quickly in case the tracker loses the target. Although, the strategy of combining correlation search and affine tracking described in the previous section does support fairly large motion in the image, it cannot handle cases in which the robot turns or pitches rapidly. Those situations cannot be avoided since the robot may drive over substantially large obstacles. Therefore, additional algorithms for fast *target recovery* need to be integrated in the system. We describe below one such recovery algorithm which was shown experimentally to be fast enough for recovering targets without affecting the robot's speed in typical cases such as driving over, or climbing down from obstacles such as curbs. Finally, because of the potentially harsh lighting conditions in outdoor environments, a number of undesirable imaging artifacts may adversely affect the performance of the tracking. Although those artifacts are caused by physical effects that cannot be fully eliminated, efficient algorithms for reducing imaging artifacts are included in the tracking system. We show below how those algorithms make smooth operation of the tracker still possible under difficult ambient lighting conditions.

In the rest of this section, we describe in detail the algorithms used for addressing those issues and we illustrate the performance of those algorithms by using the results of the Ft. Sam Houston experiments.

3.1. Updating Target Size

Since we are interested in applications in which the robot may travel over a substantial distance toward the designated target, the appearance of the target may change drastically over time from the template that was originally selected by the user. There are two distinct affects that contribute to this problem: the change of template size over time, and the change of template shape over time. The problem is complicated by the fact that template initialization is an expensive operation compared to template tracking. Therefore, continuously updating the template based on the most recent image is not recommended.

Instead, we address those problems by occasionally reinitializing the template based on the analysis of size and shape history over time, thus compromising between optimality of the template and computational efficiency. It should be noted that, although the issues addressed in this section are not fundamental algorithmic issues, they are critical to proper operation of the driving system.

The first problem is that the size of the template increases as the robot approaches it. This is not a problem when the robot is "far" from the target since the increase in size is relatively modest. However, the increase in template becomes much more rapid as the robot comes closer to the target. Rapid changes in size lead to two problems. First, if the template is not periodically re-initialized, the image becomes substantially different from the initially template and the track may be lost. Second, if the template is frequently re-initialized to the appropriate size, the extra computation cost of matching larger template may be too high.

To illustrate this point, Figure 5(a)-(b) show the result of tracking a target over a travel distance of 15m. As expected, the template grows slowly initially, but then reaches a size large enough that it makes real-time tracking difficult to achieve.

A solution to this problem is to periodically reinitialize the template based on its change in size. If X and Y are the current sizes of the template in the column and row directions, respectively, and (X_o,Y_o) is the size of the template had the last time it was initialized, then the template is unchanged as long as $(X - X_o)/X_o < SR_{max}$ and $(Y - Y_o)/Y_o < SR_{max}$. SR_{max} is the maximum allowed relative change in size. If $(X - X_o)/X_o < SR_{max}$, then the template is reduced symetrically along the columns to X_o; a similar algorithm is applied in the Y direction. This approach maintains the template at a manageable size while ensuring that the center position of the template remains at the location originally selected by the user. Figure 5(c) shows the template after update using this algorithm. Although the template should have grown from 40 to more than 100 pixels if the size change were not controlled, its size remains close to the original size, while still being centered at the correct location.

The implementation makes this algorithm fully reversible in that the same algorithm addresses the size reduction issues. In addition to the relative threshold SR_{max}, an absolute threshold SA_{max} controls the maximum allowed size change of the template with respect to the initially selected template. This threshold is introduced to avoid limit situations in which the robot is too close to the target for the dynamic size adjustment to be meaningful. A complete history of the size changes is kept in order to evaluate the absolute size change.

Another problem that occurs in updating the template is that the template may be substantially deformed over time. In fact, the transformation between the original template and the current image template may be arbitrarily large depending on the motion of the robot. Although, in theory, the tracker should be able to handle such deformations by appropriate warping of the template, in practice, large warps of the template lead to poor performance. In practice, the scales S_x, S_y, skew k and rotation α are extracted from the transformation A obtained by compounding the individual transformation δA between frames. The template is reinitialized if any of those of those parameters is greater than a threshold, S_x^{max}, S_y^{max}, S_k^{max}, S_α^{max} (more precisely, in the case of scale, the template is updated when $|S_x - 1| > S_x^{max}$.)

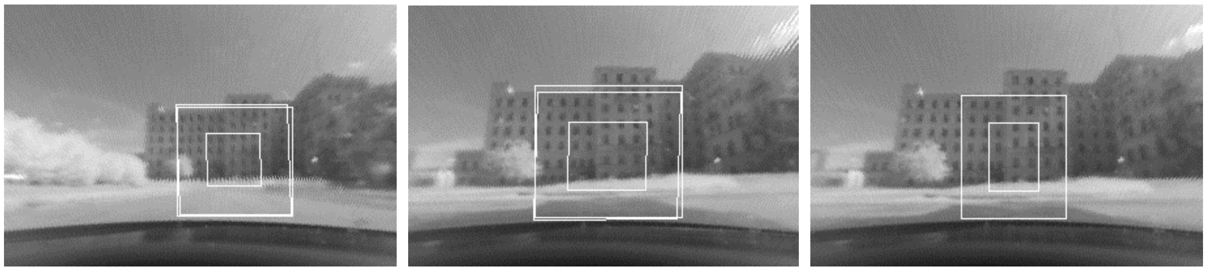


Figure 5 Template update example: (a) Initial target; (b) Target reaches maximum change in scale in the horizontal direction; (c) Target is re-scaled horizontally.

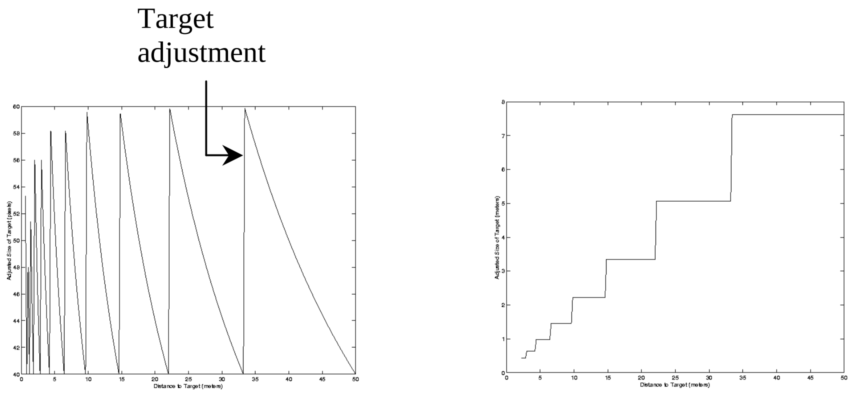


Figure 6 (a) Size of template in pixels, taking into account the periodic updates as function of distance traveled, assuming 40x40 pixels initial target at 50m; (b) Physical size of template, taking into account the periodic updates as function of distance traveled, assuming 40x40 pixels initial target at 50m.

3.2. Changing Virtual View

Since the tracking algorithm assumes an affine transformation between two neighboring templates and this holds only for perspective image planes, we cannot apply the standard tracking directly on the disc image. However the tracking can take advantage of the special spherical geometry by "pointing" the image in the appropriate direction (Figure 7.) When the user selects an object in 3D space, a perspective image plane P_1 along this viewing direction is constructed. On this perspective image plane, the affine transform assumption for the tracking holds, so we can use the standard tracker to estimate the affine transform which gives us six parameters δA[1]. We can keep using this tracker to track the object on this perspective plane, but when the object moves far away from the optical axis, which is also the original viewing direction, the template will be skewed by a larger amount. Eventually, the template will move outside of the field of view of the current virtual image. In that case, a new viewing direction is computed as the direction of the current center of the template. Since changing the direction of the virtual images amounts to a rotation around the virtual optical center of the images, the template T_2 in the new image is related to the template T_1 in the current virtual image by a homography H: $T_2 = H(T_1)$. The homography is given by: $H = M^{-1}R_2^{-1}R_1^{-1}\mathrm{M}$, where R_i is the rotation matrix corresponding to the viewing direction , and M is the projection matrix defined by f and by the sizes of the perspective image in x and y. After applying the homography transformation, the template is re-initialized and the tracking proceeds as before in the new virtual image.

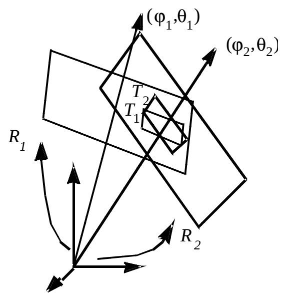


Figure 7 Geometry of the virtual view transfer.

In practice, the time required to compute the new lookup table LUT(φ_2,θ_2) may be significant compared to the tracking time. As a result, performance would be substantially degraded in situations in which the robot executes a fast point turn. A simple solution to this problem is to pre-compute the lookup tables for a predefined set of angular values, $(\Phi_k,\Theta_k)_k=1,..,N$. Given the "ideal" viewing direction (φ_2,θ_2), the closest direction (Φ_k,Θ_k) is selected out of the set of all pre-computed directions, and the corresponding lookup table LUT(Φ_k,Θ_k) is used instead of LUT(φ_2,θ_2). The advantage of this approach is that the cost of changing virtual views is essentially reduced to the cost of reinitializing the template. The costs of computing and applying the homography and of switching lookup tables are negligible.
The price to pay for this time saving is that the new direction (φ_2,θ_2) is no longer optimal, i.e., the center of the image is not necessarily at the center of the template. This is addressed by using a dense enough set (Φ_k,Θ_k) so that the virtual view selected from the pre-computed set will always be close to the ideal view. In practice, we use sixteen equally spaced values of Θ and three values for Φ.
Figure 8 shows one example of view transfer. In this example, the left panel shows the location of the template selected on the corner of a building as it is about to leave the field of view of the current virtual image (φ_1,θ_1). The right panel shows the same template after transfer to a new virtual view. This approach to view transfer allows us to do tracking over a 360^o field of view without mechanical panning and without interrupting the tracking cycles.

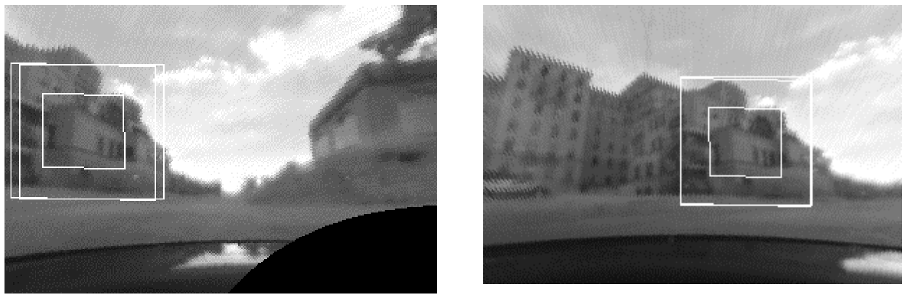


Figure 8 View transfer example: (a) Target moves out of current virtual view; (b) Target is re-initialized in closest virtual view (45^o virtual camera pan).

3.3. Target Recovery

The tracker is prone to lose the target region when there is an abrupt change in the appearance of the target, such as when the target is undergoing a rapid motion that could be not be handled by the tracker or a partial or fully occlusion happens. Thus it is critical for the tracker to detect loss of target. The tracker residual error is used for that purpose. Since the input image is normalized by its variance σ prior to processing, the residual error is also normalized too. The tracking result is rejected if the residual error is greater than one, i.e., the average error on the original images is greater than σ. In addition to an absolute limit on the residuals, a limit of the rate of change of the residual is also used, as well as a limit ρ_{min} on the best correlation value found in the search phase of the tracker. In the experiments discussed here, ρ_{min} was set to 0.8.

After a target loss is detected, a correlation-based search method is used to recapture the target. This is essentially the same correlation search method described earlier but using a larger search window. In the current implementation, motions of +-32 pixels are searched when in recovery mode. The recovery search is fast enough to be performed on the fly.
This method effectively prevents the tracking template from shifting away from the true target. In practice, the target could be lost when a full occlusion occurs, e.g. another object passes between the target and the robot. More common, however, are the cases in which the robot undergoes a sudden change of orientation due to collision with a small obstacle or because of a sudden change in terrain slope. Figure 9 shows an example in which the robot drove over an obstacle. The recovery mode is activated to relocate the target and normal tracking resume as soon as the target is located Figure 10 shows a typical example of a situation that cannot be avoided in urban terrain, in which the change in orientation – pitch in this case – of the robot is too rapid. Figure 11 shows the profile of the pitch angle as the robot drives over a log. Such angular variations were observed throughout the experiments and prompted the activation of the recovery mode at regular intervals.
In the current system, the robot stops if the target is not recovered after less than two seconds. At that point, another recovery mode which searches through a large swath of the spherical image may be activated. Since this recovery mode was not integrated at the time of this series of experiments, we do not describe it further.

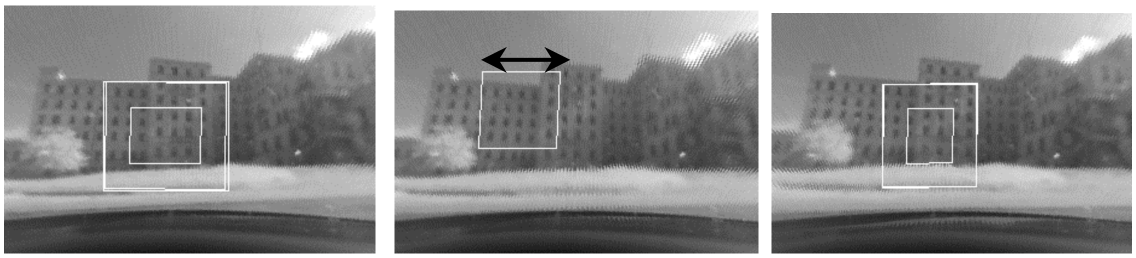


Figure 9 Recovery example: (a) In normal tracking, the search window is +/- 16 pixels around current target position; (b) In recovery search search window is increased to +/- 32 pixels. Box shows region swept by the upper left corner of target in the search region; (c) Normal tracking resumes after recovery.

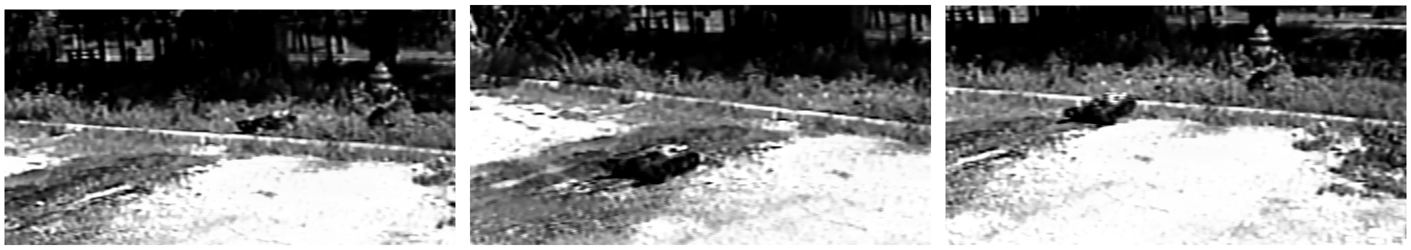


Figure 10 Going down a curb and driving over obstacles: typical examples of situations in which on-the-fly recovery is needed.

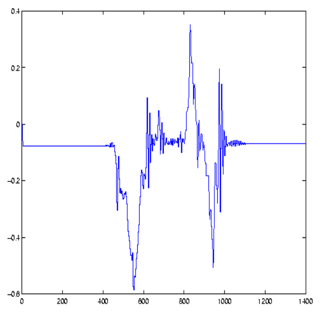


Figure 11 Profile of the pitch angle as the robot drives over an obstacle. The horizontal axis is time in ms; the vertical axis is pitch angle in radians.

3.4. Suppressing Imaging Artifacts

The discussion thus far has assumed implicitly that the camera produces accurate images of the environment. In fact, a number of artifacts may corrupt the images depending on the lighting conditions. Chief among them is the blooming effect which causes entire columns of the omnicam image to be saturated. This effect occurs whenever a light source visible in the field of view of the camera causes the illuminated CCD elements to saturate. If the saturation is sufficiently drastic, the CCD readout logic causes the illuminated elements to corrupt an entire column. This happens typically under harsh sunlight and clear sky. After warping, the blooming columns in the raw omnicam image are mapped to curves in the virtual perspective images. Although the camera is outfitted with neutral density filters in order to reduce the effect of strong illumination, the residual image artifacts are still sufficiently noticeable under extreme lighting conditions to warrant the use of additional filtering software.
Assuming that the blooming region is narrow enough, the artifact can be partially suppressed by using an interpolation algorithm. First, the algorithm identified the affected columns in the input image. Then, for every pixel, it computes an intensity value interpolated from the pixels on both sides of the affected region closest to the current pixel. Depending on the position of the pixel, the interpolation is performed along the line joining the current pixel to the center of the disc image, or along the circle passing by the current pixel in the disc image.
Figure 12 shows an example of a virtual perspective image before and after blooming reduction. In practice, this algorithm suppresses blooming sufficiently well for successful operation of the tracker under normal operating conditions.

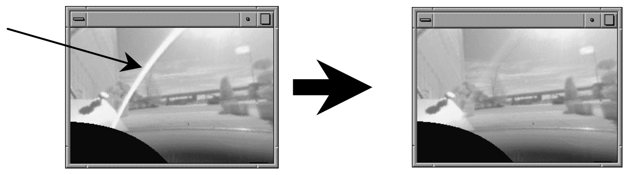


Figure 12 Effect of blooming: (a) Virtual image before processing; the white arc indicated by the arrow is caused by blooimg due to strong sunlight; (b) Same virtual image after blooming suppression.

3.5. Computational Issues

All on-board computations were carried out on a Pentium 166MHz processor. Figure 13 shows the computation time as a function of template size for one cycle of the tracker. The cycle time includes all the stages of the tracker, including virtual image filtering, view generation, correlation search, affine tracking and warping, and driving command generation. Typical target sizes are 40 to 50 pixels, in which case the computation time is close to the frame rate.

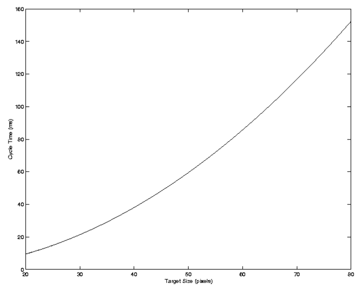


Figure 13 Computation time as function of template size as measured on the current configuration.

4. CONCLUSION

A first set of experiments shows that semi-autonomous control using image designation can be achieved. The experiments forced the system to be exposed to all the challenges that a robotic urban robot would encounter, including large variations in lighting, harsh sunlight, obstacles such as curbs and rocks. The results show that continuous tracking of distant targets can be achieved even in the presence of large disturbances caused by obstacles and terrain roughness. Moreover, the recovery of targets through large angular motions validated the use of the omnicam as a possible sensor for designation and driving. Imaging artifacts, such as blooming, that may be experienced in harsh sunlight were shown to be suppressed sufficiently for the tracking to perform.

A number of issues still remain to be addressed. First of all, the recovery algorithms described in this paper use a limited search area to search and recover targets. In practice, the offset between predicted target location and actual location may be much larger. This could happen, for example, when the robot executes a point turn much faster than any tracker could handle in the image – the robot is, in theory, capable of 100^o/s rotations. Another case is when the robot drives over a large obstacle, in which case the pitch rate may exceed the limits of the tracker. An additional mode of recovery must be used in those cases. We have developed a wide recovery algorithm which searches a large area in both horizontal and vertical angular deviations from the predicted location of the target. This recovery algorithm computes the predicted projective warp from the last known instance of the target to seed directions in increments of 20^o and searches for the warped template in a search window 20^o wide. The best location is used as the recovered template. Unlike all the other algorithms, this recovery mode requires the robot to be stopped. However, preliminary experiments show that the target can be recovered in approximately 100 ms per 20^o of angular offset between true and predicted target directions, which shows that the delay incurred by this additional recovery mode would not affect the apparent progress of the robot substantially.

A second issue is that the servoing is used in the absence of any range estimate. Although, in principle, this is a good thing since the servoing does not rely on the availability of stereo of other range sensors, it prevents the system from using a reliable termination condition. Also, range information would help validate target recovery by comparing predicted and actual range. Efforts are under way to combine the existing system with the stereo system provided by JPL.

Finally, a last issue is that the system does use any information from the robot's state estimation module, e.g., the pose from dead-reckoning. The main reason for not using the information is that dead-reckoning is poor on this type of robot, at least when using only the information from the track encoders, and is certainly not accurate enough to use as a predictive term in the tracking. However, it might be beneficial to use the state information in order to predict the performance of the tracker, e.g., by correlating the frequency of motion with tracker error. We are currently investigating such an approach for more robust tracking.

5. REFERENCES

1. P. Corke and M. Good. Dynamic effects in high-performance visual servoing. *Proc. IEEE Int. Conf. Robotics and Automation*. pp. 1838-1843, Nice, May 1992.
2. D. W. Gage. An evolutionary strategy for achieving autonomous navigation. In *Proceedings of SPIE Mobile Robots XIII, Boston MA*. November 1998.
3. G. Hager and K. Toyama. The XVision system: *A general-purpose substrate for portable real-time vision applications*. Computer Vision and Image Understanding Vol.69, No.1, pp.23-37
4. E. Huber and D. Kortenkamp. Using stereo vision to pursue moving agents with a mobile robot. *Proc. 1995 IEEE Conf. on Rob. and Automation*. pp 2340-2346, Nagoya, Japan, May 1995
5. S. Hutchinson, G. Hager, Corke, P. A. *Tutorial on visual servo control*. IEEE Transactions on Robotics and Automation, Vol. 12, No. 5, 1996, pp. 651-671
6. J. Kay, C. Thorpe. An Examination of the STRIPE vehicle teleoperation system. *Proc. IEEE/RSJ International Workshop on Intelligent Robots and System*s. Grenoble, 1997.
7. S. Nayar. Catadioptric omnidirectional camera. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*. pp 482-488. 1997.
8. S. Nayar, V. Peri. Folded catadioptric cameras. Proc. *IEEE Conf. on Computer Vision and Pattern Recognition*. 1999.
9. C. Rasmussen, G. Hager, Robot navigation using image sequences. *Proceedings of the AAAI Conference on Artificial Intelligence*. pp. 938-943, 1996
10. A. Stentz. Best information planning for unknown, uncertain, and changing domains. *Proc. AAAI-97 Workshop on On-line-Search*. 1997.
11. K. Toyama and G. Hager. Incremental focus of attention for robust visual tracking. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. pp. 189-195, 1996
12. D. Wettergreen, H. Thomas, and M. Bualat. Initial results from vision-based control of the Ames Marsokhod rover. *Proceedings IROS'97*.

6. ACKNOWLEDGEMENTS

This work was supported by the DARPA Tactical Mobile Robotics program under JPL contract 1200008.