Motion Strategies for People Tracking in Cluttered and Dynamic Environments

Tirthankar Bandyopadhyay[1], David Hsu[2], and Marcelo H. Ang Jr.[1]

[1] Department of Mechanical Engineering, tirthankar@nus.edu.sg; mpeangh@nus.edu.sg
[2] Department of Computer Science, dyhsu@comp.nus.edu.sg
National University of Singapore, Singapore.

Summary. The key contribution of this work is the construction of a robot tracking system that follows an unpredictable target (e.g people) in cluttered, unknown and dynamic environments. To do so, mobility constraints, sensor limitations, and uncertainty have been addressed at both the algorithmic and implementation levels. The work formulates target tracking as minimizing an objective function that models the risk of the target's escape and derives this risk function from basic principles. Various practical hardware limitations – e.g., sensor's field of view, safe navigation – are taken into account either as part of the risk function or as constraints during risk minimization. The resulting tracker demonstrates substantially improved performance, compared with other local online strategies. The tracker was able to track a person walking around in indoor office environments as well as in a highly dynamic and public environment like the school cafeteria successfully.

1 Introduction

Target tracking is an important task for autonomous robots. The goal of this work is to construct motion strategies for a robot equipped with visual sensors so that it can track a moving target despite obstruction by obstacles. Target tracking has many applications in security and surveillance systems, home care settings, computer assisted visualizations, etc. Let us take a specific scenario of an automated personal shopping assistant following an elderly person in a shopping mall, or of keeping an eye on young kids while their parents shop. The shopping mall is complex environment. Moreover, people walking around add to the visual occlusions and motion obstructions, thereby creating a highly cluttered and dynamic environment. While the layout of the environment might be available in some cases, exact maps for localizing the robot are hardly provided. On top of that, the target can be completely unpredictable in moving from one shop to another. In such scenarios, an online tracking strategy that uses only local information becomes necessary.

Tracking strategies differ greatly, depending on the information available. For known environment and target motion, optimal motion can be precomputed [5, 3, 6], though usually at a high computational cost. For unknown

environment and unpredictable targets, [4, 7] move the robot to minimize a distance based objective function. The concept of vantage time [2, 1] systematically integrates various local factors contributing to the target's perceived risk of escape and derives a risk function that is minimized to attain better tracking performance. While our earlier work [2, 1] introduced the vantage tracker, it was done in simulation with perfect sensing, localization, omnidirectional visibility and motion, making it difficult to implement directly on a real hardware. This paper describes the enhanced algorithm to handle such practical requirements, and the realization of such a tracking system.

2 Problem Formulation

In order to identify and track a person, we use visibility based sensors, based on the standard straight line-of-sight visibility model. In the free space $\mathcal{F}$, the *visibility set* $\mathcal{V}(x)$ is given by,

$$\mathcal{V}(x) = \{q \in \mathcal{F} \mid \overline{xq} \subset \mathcal{F} \text{ and } d(x,q) \leq D_{max} \text{ and } \theta_{min} \leq ang(x,q) \leq \theta_{max}\}$$

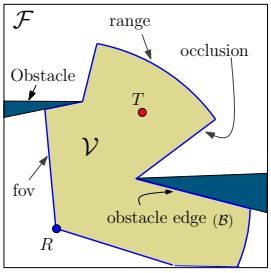


Fig. 1. The shaded region depicts $\mathcal{V}$ bounded by different escape edges

where $d(x,q)$ denotes the distance between x and q, while $ang(x,q)$ is the orientation of q w.r.t. x. Information about the local environment is encoded into the boundary ($\partial\mathcal{V}$), of the visibility polygon ($\mathcal{V}$).

Both the robot and the target's motion, are modeled with a simple discrete-time transition equation. As the target behavior is unknown, its velocity ($\mathbf{v}'$) is modeled by a gaussian around its current heading : $\mathbf{v}'(t+\Delta t) = \mathcal{N}(\mathbf{v}'(t), \sigma)$. The variance σ gives a measure of confidence in estimating the target velocity. Although we use a Gaussian distribution to model the uncertainty in the target behavior, the approach remains valid for any other velocity prediction method, even non-parametric ones.

2.1 Local Greedy Approach

The objective of the robot is to keep the target inside the robot's visibility, $\mathcal{V}$, by manipulating the *escape edges*, $\{\mathcal{G}_i\}$ away from the target. Let us denote the manipulation ability of a single escape edge, $\mathcal{G}_i$, by the symbol, $\Delta\mathcal{G}_i$. $\Delta\mathcal{G}_i$ is a function of the robot position, $\mathbf{x}$, and its actions, $\mathbf{v}$: $\Delta\mathcal{G}_i(\mathbf{x}, \mathbf{v})$. The risk of losing the target, on the other hand, depends on : (a) the target position ($\mathbf{x}'$), (b) the relative target velocity ($\mathbf{v}'$) w.r.t. to $\{\mathcal{G}_i\}$, and finally (c) the robot's ability to manipulate the edges, $\Delta\mathcal{G}_i$. We can then formulate a risk function (Φ) and choose the robot action, $\mathbf{v}^\star$, to minimize Φ:

$$\begin{aligned} Risk &= \Phi(\mathbf{x}', \mathbf{v}', \{\mathcal{G}_i\}, \{\Delta\mathcal{G}_i(\mathbf{x}, \mathbf{v})\}) \\ \mathbf{v}^\star &= \mathbf{arg\ min}\ \ \Phi(\mathbf{x}', \mathbf{v}', \{\mathcal{G}_i\}, \{\Delta\mathcal{G}_i(\mathbf{x}, \mathbf{v})\}) \end{aligned} \tag{1}$$

While Φ is the risk of losing the target through any escape edge in the entire $\mathcal{V}$, we can assign a risk φ_i, of losing the target to each escape edge, $\mathcal{G}_i$. We approximate the total risk Φ, by the expected risk for all the gaps.

$$\Phi \approx E[\varphi_i] = \sum_i p_i \varphi_i(\mathbf{x}', \mathbf{v}', \mathcal{G}_i, \Delta\mathcal{G}_i(\mathbf{x}, \mathbf{v})), \qquad \mathbf{v}^\star \approx \sum_i p_i v_i^\star \tag{2}$$

where p_i is the probability of the target's escape through $\mathcal{G}_i$. p_i is computed based on the target's current velocity, $\mathbf{v}'$. The details can be found in [2].

However, in choosing $\mathbf{v}^\star$, the robot has to satisfy many constraints on the desired robot positions, *obstacle avoidance, stealth* or on the robot's actions, *kinematic, dynamic constraints*. We define a *feasible region*, $\mathcal{L}$ (x), that satisfies all the constraints in the position domain, $\mathcal{C}_i(x) : \mathcal{L}(x) = \bigcap_i \mathcal{C}_i(x)$. The local greedy optimization then chooses an action ($\mathbf{v}^\star$), that minimizes Φ while satisfying $\mathcal{L}$ (x) in the time step Δt,

$$\mathbf{v}^\star = \mathbf{arg\ min}\ \ \Phi(\mathbf{x}', \mathbf{v}', \{\mathcal{G}_i\}, \{\Delta\mathcal{G}_i(\mathbf{x}, \mathbf{v})\}) \qquad s.t \quad \mathbf{v}^\star \Delta t \in \mathcal{L}(\mathbf{x}) \tag{3}$$

3 Tracking Algorithm

In the previous work [2], a local greedy algorithm based on *relative vantage* was proposed. Relative vantage refers to the ability of the robot to eliminate $\mathcal{G}_i$ before the target can escape through it. We introduce a region around $\mathcal{G}_i$, called *vantage zone*, $\mathcal{D} = \{q : q \in \mathcal{V}; dist(q, \mathcal{G}_i) \leq dist(\mathbf{x}, \mathcal{G}_i)\}$

The objective of the robot is to keep the targets away from $\mathcal{D}$ and accordingly, we can take the measure of time taken to move the target outside $\mathcal{D}$, $t_{r.v}$, as the risk value. From Figure 2,

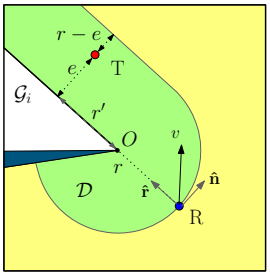
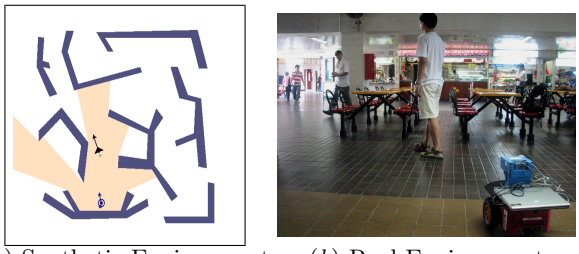


Fig. 2. Calculating $t_{r.v}$ for occlusion edges

$$\varphi_g = \frac{dist(T, \mathcal{D})}{rel.vel(T, \mathcal{D})} \approx \frac{r - e}{v_{\text{eff}}}, \qquad v_i^\star = \frac{\varphi_g}{v_{\text{eff}}} \left(\frac{r'}{r} \hat{\mathbf{n}} + \hat{\mathbf{r}} \right)$$

where $v_{\text{eff}} = v_r + v_n(r'/r) - v'_{\text{e}}$ is the effective velocity in the direction along the shortest path from the target to $\mathcal{G}_i$. The algorithm showed improved tracking performance in simulation. The details are addressed in [2]. However, the simulation assumed omni-directional visibility, perfect sensing, perfect target identification among other things, (Figure 3*a*).

In real life (Figure 3*b*), most visibility sensors available have limitations on the range of sensing and the field of view which have to addressed explicitly. Another important issue in implementing the tracking system, is the identification of the target feature from sensor data. In simulation, this problem is bypassed as the simulator can be queried for the exact position of the target. However, uncertain and noisy sensor data in real life scenarios, makes target identification and localization unreliable. The tracking algorithm has to handle such cases of false detection and no detection. Also, the robot has a



(a) Synthetic Environment (b) Real Environment

Fig. 3. Adapting the algorithm from a synthetic environment (a) to real environment (b) requires addressing the problem of hardware limitations and uncertainty.

finite size and may have kinematic and dynamic constraints that need to be considered. Safe robot navigation is necessary not only for the robot itself, but also to prevent damaging the environment. In fact, safety becomes a critical issue when introducing the robot in human environments.

Handling Visibility Limits

In general, there are two kinds of sensory limitations : limits on the maximum and minimum sensing range, *range edges*, and limits on the field of view, *FoV edges*. As these edges cannot be eliminated by the robot itself, the best strategy would be to delay the target's escape. A measure of this delay, the time taken by the target to escape through a gap ($\mathcal{G}$), is called *escape time* (t_{esc}). The robot then chooses actions that maximizes t_{esc}, where we define t_{esc} by,

$$t_{esc} = \frac{dist(T, \mathcal{G})}{rel.vel(T, \mathcal{G})}$$

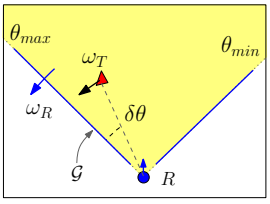


Fig. 4. FoV limits

Field of View Limits : The FoV can be modeled as an annular sector with the robot at the center, with the visibility spanning from θ_{min} to θ_{max}, Figure 4. The only way to manipulate $\mathcal{G}$ is by rotating the visibility sensor towards the target. In case the visibility sensor has an additional degree of freedom over the robot, *e.g.* a pan mechanism, the angular velocity of the panning, ω_R is the action of the robot. On the other hand, if the sensor is attached rigidly to the robot base, the turning of the robot itself acts to rotate $\mathcal{G}$. In that case we treat ω_R as the rotation of the robot.

As we deal with the angular motion, it is natural to derive t_{esc} using these rotational parameters. Based on the target's motion, we can approximate its angular velocity, ω_T towards $\mathcal{G}$. This gives $rel.vel(T, \mathcal{G}) = \omega_T - \omega_R$, and $dist(T, \mathcal{G}) = \delta\theta$. Treating t_{esc} as the risk, φ_f, with $\omega_{eff} = \omega_T - \omega_R$,

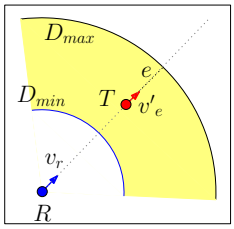


Fig. 5. Range Limits

$$\varphi_f = \frac{\delta\theta}{\omega_{eff}}, \qquad v^{\star}{}_{FoV} = \frac{\varphi_f}{w_{eff}}\hat{\omega}_R \tag{4}$$

Max/Min Range Limits : In Figure 5, D_{max} and D_{min} show the visibility range limits. In this case, e is the distance towards the nearest point in the range edges. From Figure 5 we can calculate the φ_d as,

$$\varphi_d = \frac{e}{v_{eff}}, \qquad v^{\star}{}_{range} = \frac{\varphi_d}{v_{eff}} \hat{v}_r \tag{5}$$

where v'_e is the velocity component of the target towards the escape edge D_{max}, and v_r is the robot's velocity in pushing the range edge away from the target, and $v_{eff} = v'_e - v_r$. The analysis for D_{min} is identical where, the robot would actually back away from the target to guard D_{min}.

An interesting thing to note, is that the behavior generated by the range edges alone is exactly the visual servo behavior. This shows that visual servo is a special case of vantage tracking when there are no occlusions.

Obstacle Avoidance

Although, purely low-level reactive obstacle avoidance techniques, can handle dynamic and unknown environment, they may sometimes move the robot contrary to the required tracking direction. On the other hand, planning in the configuration space may be too computationally expensive in a cluttered and dynamic environment. We propose a local obstacle avoidance method with a small look-ahead. The robot's velocity is used to enlarge the obstacle edges. These extended obstacles then constrain the planned robot motion.

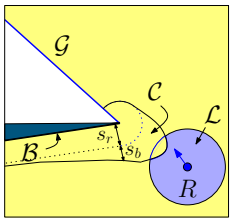


Fig. 6. Obstacle Dilation

First, we approximate the robot's size by the radius (s_r) of its bounding circle. Then, we compute the finite braking distance, s_b, using the maximum decceleration and the robot's current velocity. This braking distance, s_b and the robot's dimension, s_r, defines a collision region $\mathcal{C}$ (x), around the obstacle edge, $\mathcal{B}$, Figure 6,

$$\mathcal{C}(x) = \{q \in \mathcal{V} : d(q, \mathcal{B}) \leq (s_r + s_b)\} \tag{6}$$

The robot can actively change the shape of $\mathcal{C}$ by changing its speed and heading. For safe navigation, the robot must avoid $\mathcal{C}$. If we denote the *reachable* region of the robot in Δt, as $\mathcal{R}$, the feasible region becomes $\mathcal{L} = \mathcal{R} - \mathcal{C}$. As an example, assuming omni-directional motion ability of the robot, Figure 6, $\mathcal{R}$ is a disk of radius, $V\Delta t$, and the darker shaded region shown is $\mathcal{L}$.

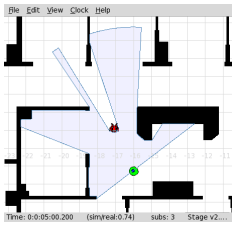


Fig. 7. Tracking scene

4 Hardware Implementation

The tracking algorithm is implemented on a Pioneer P3-DX differential drive robot. A SICK-lms200 is mounted on the robot. The laser returns 361 readings on a field of view of 180deg at the resolution of 0.5deg. The maximum range of the robot is 8m. The control algorithm

runs on a Pentium M Processor @1.5GHz laptop running Player server v-2.0.5 on linux. The algorithm runs at 10Hz. The target is a person walking around the lab corridors in the presence of other people. For illustration purposes, a snapshot of the algorithm running in the stage simulator is shown in Figure 7. The green circle is the robot and the red object is the target.

Visibility Polygon : The output of the sensor is a radial scan of range values. The data points with max-range readings form the range edges. We detect the continuity of the remaining data point to its neighbors by thresholding the range value changes in adjacent data points. These changes represent the occlusion edges and their location can be extracted from the corresponding data points. This is shown by blue lines in Figure 8. Two additional edges are added at the orientation of the min/max angular limits to form the FoV limits.

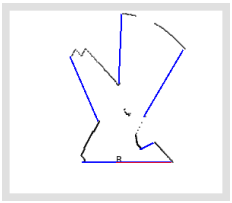


Fig. 8. Visibility Polygon

Target identification : The thresholding groups the data points into clusters (Figure 9). These clusters represent physical objects in the sensing range, *e.g.* walls, furniture, other people, *etc.* In fact, one of the clusters is the target. We start with a known target. Given the target's maximum speed, we can estimate a bound on the target's future position in time Δt: $V'\ \Delta t$. We then perform a nearest neighbor match within this bound. Clearly, this method will fail if $V'\ \Delta t$ is too large. However, the practical success of this simple technique can be found in : (a) small Δt, as the algorithm runs at a high frequency of about 10Hz, (b) target speed being slow enough, average human walking speed 1m/s (giving the tolerance level of about 10cm) and (c) low false negative rate for cluster detection. More sophisticated target detection algorithms using EKF, MHT or JPDAF, etc can improve the tracking robustness.

Fig. 9. Target Detection

Obstacle avoidance : We utilize an implementation of applying (Equation 6) to the data points directly. This saves us considerable computation in extracting the actual shape of the cluster to compute $\mathcal{C}$. To achieve this we perform a radial transformation in to move each data point towards the robot by ($s_r\ + s_b$). s_b is estimated based on the relative velocity and the maximum relative acceleration towards the data point. The result is shown in Figure 10.

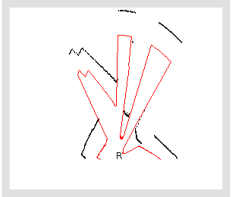


Fig. 10. Obstacle dilation

Robot Motion control : The optimal velocity $\mathbf{v}^\star$, generated from the algorithm does not take into account the non-holonomicity of the robot base. We apply a simple low level control on the robot velocity that tries to achieve $\mathbf{v}^\star$ (similar to [4]). From the structure of the risk function we see that Φ, is locally smooth. Due to this, $\mathbf{v}^\star$ also changes slowly and the controller is stable.

Uncertainty in sensing and execution : As the algorithm uses only local geometric information available to the robot's visual sensors, it does not require a global map and thus bypasses the difficulty of localization with respect to a global map. Noisy sensor data and the uncertainty in robot's control can affect the performance of the target's relative localization and especially the target's velocity estimate. In the hardware experiments, we found that the reliability of the target's velocity estimate was quite poor. Still, the tracker was able to successfully track the target by focusing more on the worst case scenarios. Moreover, uncertainty in sensing and motion control does not accumulate, because the robot's action is computed using sensor data acquired in the current step only improving the reliability of tracking.

5 Experimental Results

We run three types of experiments to validate our approach. In the first, we show the improved performance of the vantage tracker to the visual servo in a real dynamic environment. Subsequently we run control experiments on the algorithms to show and analyze the improved performance of vantage tracker.

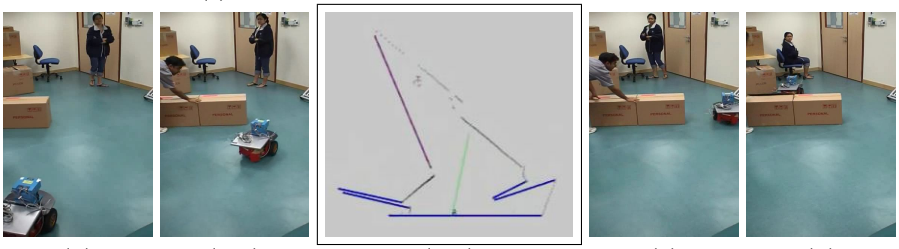
Dynamic Environment Expt : In Figure 12, a box is pushed between the target and the robot to occlude the target. Since, the vantage tracker actively tries to avoid future possible occlusions, it is able to adapt to the changing environment (Figure 12*b-1*). A point to note is that the vantage tracker does not model the motion of the environment but just re-plans its motion at a high frequency, making the tracker independent of the dynamic nature of the environment. Later, when the box stops and the target starts to move (Figure 12c), the tracker is able to successfully follow the target (Figure 12d).

Performance Comparison - Qualitative Study : In (Figure 13 & 14),We compare the performance of the different algorithms : the visual servo, combinatorial tracker (maximizing the shortest distance to escape, SDE) [4] , and our vantage tracker. In order to have controlled environment for comparison viz, identical target path and perfect target identification and localization, we run this experiment on player/stage simulator.

In Figure 14(a), we compare the performance of the trackers using the metric of the shortest distance to the nearest exit (SDE). A better performance would be indicated by larger average SDE. The plots stop as soon as the target is lost by the tracker. We see that visual servo loses the target first around step #160, (Figure 13a,14a) and then the combinatorial tracker around #300 (Figure 13b,14a) whereas the vantage tracker manages to continue till the end (Figure 13c,14a). The visual servo tracker ignores the environment due to which it loses out early. The combinatorial tracker performs better, but it focuses more on the short term goals of immediate target loss. This is seen in Figure 13b as a highly curved path due to the swinging actions. The vantage tracker balances the short term and long term goals better. Around step #300 when the combinatorial tracker loses out, the vantage tracker is much closer to the occlusion edge, and is better positioned to guard the occlusion edge.

(*a*) (*b*) (*c*)

Fig. 11. Visual Servo : Since the robot does not take into account the environment information, it moves straight ahead towards the target (b) and loses the target to the occluding box (c).

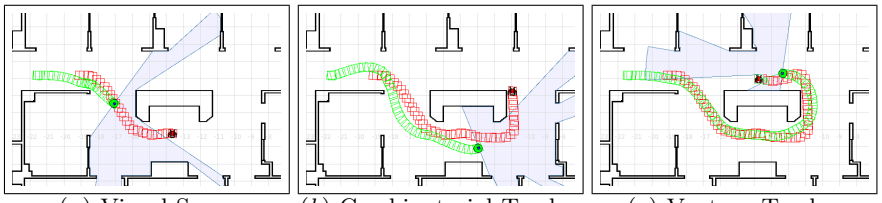


(*a*) (*b-1*) (*b-2*) (*c*) (*d*)

Fig. 12. Vantage tracker : (*b-2*) shows the robot's local perception of the environment. The target is marked by T, the blue lines are the occlusion edges, red line is the most critical occlusion and the green segment starting from R denotes the robot's motion decision. The robot sees the target too close to the occlusion and swings out.

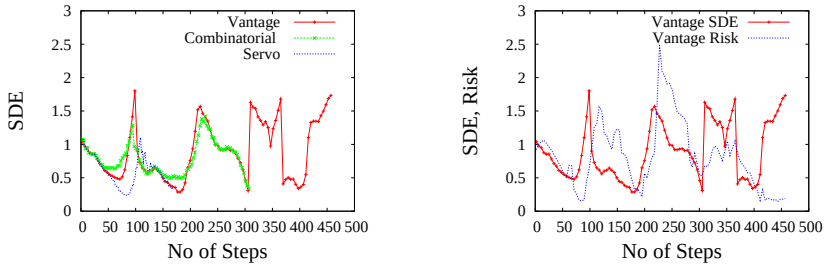
The vantage risk plot Figure 14*b*, shows peaks in the risk value when the SDE starts to fall (as the target comes closer to the occlusion edge). Anticipating this risk early allows the vantage tracker to improve its future position and keep the target safely in view.

Performance Comparison - Quantitative Study : We run the algorithms for three other target paths, shown in Figure 15, and compare the them for the percentage of the number of steps in which target was visible. When lost, the robot tries to recover the target by moving towards the target's last seen position. The results are shown in Table 1. Column 2 of the table lists the length of the target trajectory in time steps. For the each strategy, the first column lists the number of steps that the robot has the target visible as well as the number as a percentage of the total number of target steps. The next column lists the number of times that the target is lost *and* recovered with emergency actions. In case the target was not recovered even after executing emergency actions, it is marked as *lost*. The comparison in these two environments shows that the vantage tracker is (i) less likely to lose the target, (ii) has the target visible for much longer total duration, and (iii) always follows the target to the end. All these indicate better performance.



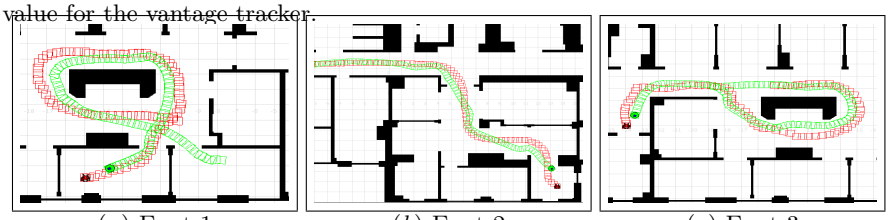
(a) Visual Servo (b) Combinatorial Tracker (c) Vantage Tracker

Fig. 13. The green robot is trying to follow the red target. The trails show their actual path. The light blue shaded region denotes the robot's visibility. Target is lost in (a) and (b), whereas in (c) the target is still in robot's view. In all cases the robot stops when the target is lost from view, there is no target recovery behavior.



(a) (b)

Fig. 14. (a) Plotting SDE for various algorithms, (b) Plotting SDE and the risk value for the vantage tracker.



(a) Expt 1 (b) Expt 2 (c) Expt 3

Fig. 15. The simulation experiment paths taken by the target (red) and the path of the vantage tracker (green).

6 Conclusion

This work deals with the problems, both theoretical and practical, to develop a human tracking system on real hardware. We model the FoV limitations as an objective function, while safe obstacle avoidance is modelled as constraints. We show that for reasonable target behaviors, a simple cluster based detection and nearest neighbor data association works well in practice. This enhanced algorithm can now handle real life conditions and we setup a robot that can track humans in semi-structured, unknown and dynamic environments using

Table 1. Performance comparison of visual servo, combinatorial and vantage trackers.

		Visual Servo		Combinatorial		Vantage	
Expt. No.	Target Steps	Visible Steps (%)	No. Times Lost	Visible Steps (%)	No. Times Lost	Visible Steps (%)	No. Times Lost
1.	1200	268 (22%)	2, lost	230 (19%)	2, lost	1015 (85%)	4
2.	1700	467 (39%)	6, lost	295 (17%)	1, lost	705 (42%)	5
3.	1200	309 (26%)	4, lost	237 (20%)	1, lost	399 (33%)	5

just a simple laser scanner. Experiments also show that it outperforms standard visual servo and SDE based trackers. The proposed algorithm worked reasonably in the school cafeteria where people walking by may introduce additional motion obstruction and visibility occlusions dynamically. All videos are available at : *http://motion.comp.nus.edu.sg/projects/follow/follow.html*

In the current implementation the target identification can be improved significantly. A lot of research has been done in robust identification of a person based on facial features etc, in computer vision. We are working towards integrating computer vision techniques to improve the target identification. This would be absolutely essential in distinguishing between probable targets in case the target is lost and the robot has to apply advanced target searching algorithms to find the target again.

References

1. T. Bandyopadhyay, M. Ang Jr., and D. Hsu. Motion planning for 3-d target tracking among obstacles. In *Proc. Int. Symp. on Rob. Research*. Springer, 2007.
2. T. Bandyopadhyay, Y. Li, M. Ang Jr., and D. Hsu. A greedy strategy for tracking a locally predictable target among obstacles. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 2342–2347, 2006.
3. A. Efrat, H. González-Baños, S. Kobourov, and L. Palaniappan. Optimal strategies to track and capture a predictable target. In *Proc. IEEE. Int. Conf. on Robotics & Automation*, pages 3789–3796, 2003.
4. H. González-Baños, C.-Y. Lee, and J.-C. Latombe. Real-time combinatorial tracking of a target moving unpredictably among obstacles. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 1683–1690, 2002.
5. S. LaValle, H. González-Baños, C. Becker, and J. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 731–736, 1997.
6. R. Murrieta, A. Sarmiento, and S. Hutchinson. A motion planning strategy to maintain visibility of a moving target at a fixed distance in a polygon. In *IEEE Int. Conf. on Robotics & Automation*, 2003.
7. R. Murrieta-Cid, H. H. González-Baños, and B. Tovar. A reactive motion planner to maintain visibility of unpredictable targets. In *Proc. IEEE. Int. Conf. on Robotics & Automation*, pages 4242–4248, 2002.