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.

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, tracking strategies enable mobile sensors to monitor moving targets in cluttered environments. In home care settings, a tracking robot can follow elderly people around and alert caregivers of emergencies or even help elderly or disabled people as personal porters. Let us take a specific scenario of an automated personal shopping assistant following an elderly person in a shopping mall or keeping an eye on young kids while their parents are shopping. The shopping mall is an example of a highly dynamic environment with people walking around and can create obstruction and occlusion in addition to the static environment for the tracking robot. While the layout of the environment might be available in some cases, exact maps for localizing the robot are hardly provided. Moreover, the target can be completely unpredictable in moving from one shop to another. In such scenarios an online tracking strategy that can plan its actions based on local information is necessary.

Target tracking has to deal with *motion constraints* in the presence of obstacles, as well as *visibility constraints* posed by occlusions in a cluttered environment [2]. Additionally, the physical realization of a tracking system requires consideration of *hardware constraints* viz, limited sensor range, field of view (FoV), noisy sensor readings, uncertain target identification, inaccurate odometry etc, at the algorithmic level as well as at implementation level. In this work, we build upon our earlier work on target tracking using local information [2, 1] for purely geometric environments and realize it on real system in a real environment.

Tracking strategies differ greatly, depending on whether the environment is known in advance. If both the environment and the target trajectory are completely known, optimal tracking strategies can be computed by dynamic programming [6] or by piecing together certain canonical curves [4], though usually at a high computational cost. If only the environment is known, one can preprocess the environment by decomposing it into cells separated by critical curves [8]. Often, neither the environment nor the target trajectory is known in advance. One approach in this case is to move the robot so as to minimize an objective function that depends on the shortest distance for the target to escape from the visibility region of the robot's sensor, abbreviated as SDE [5, 7, 9]. The concept of vantage time [2, 1] provides a more systematic way to integrate various factors contributing to the escape risk and derive a risk function offering better tracking performance. The use of local velocity estimation to predict target motion further improves the performance of the tracking strategy.

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 real hardware. In this paper, we show that the limitations of the sensors, e.g field of view (FoV) can be modelled into the objective function itself, while safe obstacle avoidance is better modelled as constraints. We also 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 just a simple laser scanner. Experiments also show that it outperforms standard visual servo and SDE based trackers. We found our system to work well in indoor office environments and in highly dynamic public environments like the school cafeteria for reasonable target behavior.

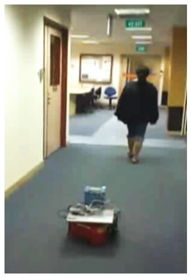
2 Problem Formulation

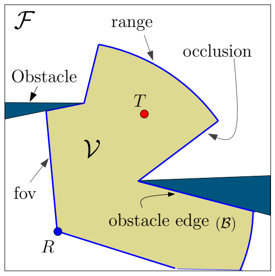
This work focuses on tracking an unpredictable target in an office like unknown environment (Figure 1a) in the presence of people dynamically creating occlusions and obstacles.

We use the standard straight-line visibility model for the robot's sensor. In Figure 1b, let $\mathcal{F}$ denote the free space. The target is *visible* to the robot if the line of sight between them is free of obstacles, and the distance between them is smaller than D_{max}, the maximum sensor range. The *visibility set* $\mathcal{V}(x)$ of the robot at position x consists of all the points at which the target is visible:

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

where $d(x,q)$ denotes the distance between x and q, and $ang(x,q)$, the orientation of q w.r.t. x. If the robot must monitor the target from a minimum distance away, we can impose the additional constraint $d(x,q) \geq D_{min}$. In the following, the visibility set is always taken w.r.t. the current robot position, and so we omit the argument x.





(a) (b)

Fig. 1. (a) The tracker (P3-DX) robot mounted with a laser (SICK lms200), following a person in an office environment (b) The shaded region depicts $\mathcal{V}$ bounded by different escape edges.

There is no explicit representation of the environment. $\mathcal{V}$ is an open set and its boundary, $\partial\mathcal{V}$, actually encodes the locally sensed environment. $\partial\mathcal{V}$ constitutes of points on the obstacles boundaries, *obstacle edges* ($\mathcal{B}$), and points in free space boundaries, *escape edges* ($\mathcal{G}$). Escape edges are of three major types, *occlusion edges*, *field of view (FoV) edges* and *range* edges, Figure 1b. $\mathcal{B}$ can be extracted quite simply from the sensor readings that are not the sensor limits.

$$\mathcal{B}(x) = \{q \in \partial\mathcal{V} \mid d(x,q) \neq D_{max} \text{ and } ang(x,q) \neq \theta_{max}, \theta_{min}\}$$

The robot's motion is modeled with a simple discrete-time transition equation. Let $\mathbf{x}(t)$ denote the position of the robot at time t. If it chooses a velocity $v(t)$ at time t, its new position $\mathbf{x}(t+\Delta t)$ after a fixed time interval Δt is given by

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + v(t)\Delta t$$

Here, we implicitly assume that sensing occurs every Δt time. This discrete model is effective as long as Δt remains small.

The target's motion is modeled similarly. We assume that the velocity bound of the robot V is not lower than the target's maximum velocity V'. Otherwise, the target can easily escape by running straight ahead with maximum velocity, and the tracking problem is uninteresting. Although the target can move in any direction, for a non-evasive target, it's heading can give a fair indication about its next step. We locally predict the target's future velocity by Gaussian distribution, $\mathcal{N}(mean, var)$, on its current velocity, $\mathbf{v}'$ (t).

$$\mathbf{v}'(t+\Delta t) = \mathcal{N}(\mathbf{v}'(t), \sigma)$$

The variance σ gives a measure of confidence in estimating the target velocity. In the worst case, the target can be assumed to take on any direction with equal probability resulting in a uniform distribution. 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, $\mathcal{V}$, for as long as possible. However the boundary of the visibility, has a number of *escape edges*, $\{\mathcal{G}_i\}$ in $\mathcal{F}$, through which the target may escape. In order to keep the target in view the robot must guard $\{\mathcal{G}_i\}$ against the target. The robot's visibility, $\mathcal{V}$ and so $\{\mathcal{G}_i\}$ at any moment is dependent on the robot's position $(\mathbf{x})$ w.r.t. its local environment and so the robot can actively manipulate $\{\mathcal{G}_i\}$, by its actions $(\mathbf{v})$. Let us first consider a single escape edge, $\mathcal{G}_i$. The ability of the robot to manipulate $\mathcal{G}_i$ effectively is important in maintaining the target in view. Let us denote this manipulation ability 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 depends on the position of the target, $\mathbf{x}'$ and its velocity, $\mathbf{v}'$ w.r.t. to all the escape edges, $\{\mathcal{G}_i\}$. Also, the risk of the target's escape decreases with increased ability of the robot's manipulation of each $\mathcal{G}_i$. If the robot can move $\mathcal{G}_i$ away from the target faster than the target's velocity towards $\mathcal{G}_i$, the risk of the target's escape is low. We can then formulate a risk function $(\varPhi)$ that takes these parameters into account and gives a measure of the risk of losing the target through all possible escape edges, $\{\mathcal{G}_i\}$ in $\mathcal{V}$ and the robot chooses an action $\mathbf{v}^\star$, to minimize this risk :

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

While $\varPhi$ 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 $\mathcal{G}_i$. As the target's behavior is unknown, it can escape through any of the escape edges. We approximate the total risk $\varPhi$, by the expected risk for all the gaps.

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

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. These constraints may either be on the desired robot positions, *obstacle avoidance, stealth* or on the robot's actions, *kinematic, dynamic constraints.* These constraints can be handled by defining 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 becomes choosing an action $(\mathbf{v}^\star)$, that minimizes $\varPhi$ while satisfying $\mathcal{L}$ (x) in the time step Δt,

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

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 each $\mathcal{G}_i$, called *vantage zone* $\mathcal{D}$, such that,

$$\mathcal{D} = \{q : q \in \mathcal{V}; dist(q, \mathcal{G}_i) \leq dist(\mathbf{x}, \mathcal{G}_i)\}$$

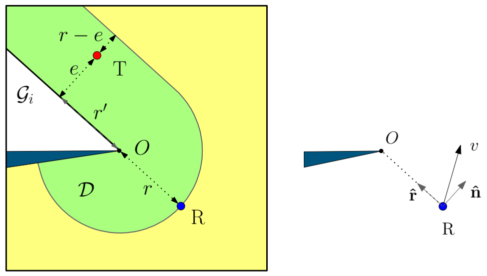


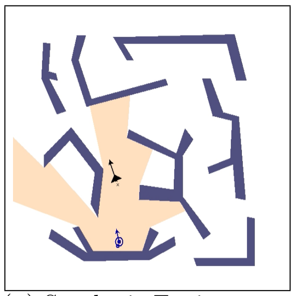
Fig. 2. Calculating $t_{r.v}$ for occlusion edges

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. Substituting the parameters from Figure 2,

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

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 details are addressed in [2].

Practical Limitations


(*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.

The earlier work [2] computes the best motion of the robot in a synthetic environment (Figure 3*a*), with omni-directional visibility, perfect sensing, perfect target identification among other things. The algorithm showed good results in simulation [2], but it is not implementable for the tracker in scene Figure 3*b*, directly.

Most visibility sensors available have limitations on the range of sensing and the field of view. Although omni-directional visibility systems can be specifically created for the robot, they are relatively more expensive and not as robust as a simple off-the-shelf camera or a laser. Handling the range and FoV limitations becomes an important requirement. 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. In reality however, uncertain and noisy sensor data makes target identification and localization unreliable. The tracking algorithm has to handle scenarios of false detection and no detection. For the synthetic environment the tracking algorithm does not have to deal with the physical characteristics of the robot. The robot was treated as a point object and it could move arbitrarily close to obstacles. In reality the robot has a finite size and may have kinematic and dynamic constraints that need to be considered. Safe robot navigation is necessary not only for the sake of the robot but also to prevent damaging the environment. In fact safety becomes a critical issue when introducing the robot in human environments.

Handling Visibility Limits

As mentioned earlier, apart from environmental occlusions, limits of the visibility sensors also pose a risk from which the target can escape. In general there are two kinds of sensor limitations : limits on the maximum and minimum sensing range *range edges* and limits on the field of view *FoV edges*. Unlike occlusion edges, these edges are special because they cannot be eliminated by robot itself. The robot needs external sensors or additional robots to be able to eliminate them. In the absence of the ability to eliminate these edges the best strategy is to delay the target's escape through these edges. 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,

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

Field of View

Most of the visibility sensors like lasers or camera have a limited field of view (FoV). This requires the tracking strategy to keep the target within the boundaries of FoV. 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.

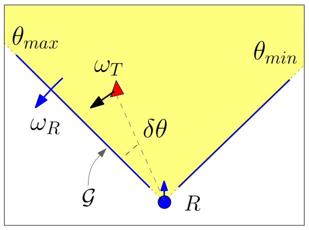


Fig. 4. FoV limits

For guarding against FoV constraints, we manipulate the *FoV edges* ($\mathcal{G}$ in Figure 4) away from the target. The only way to move $\mathcal{G}$ 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 $\omega_{eff} = \omega_T - \omega_R$,

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

Max/Min Range Limits

As with FoV edges, the robot cannot eliminate range edges as they are induced by its own sensor limitations. In Figure 5, D_{max} and D_{min} show the visibility range limits. In a similar approach as above, for *range edges* we use t_{esc} as the risk. 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,

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

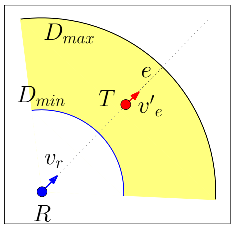


Fig. 5. Range Limits

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. The robot chooses actions based on the local gradient, where $v_{eff} = v'_e - v_r$ The analysis for D_{min} is identical in which case, 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, makes the robot move towards the target. This is exactly the visual servo behavior. This shows that visual servo is a special case of vantage tracking when there are no occlusions. The actions for FoV and range edges can be integrated directly into the formulation, (Equation 2).

Obstacle Avoidance

Real navigation has to take care of the robot's size. Local reactive planning techniques are fast and can deal with unknown and dynamic environments, they have their own drawbacks. Applying obstacle avoidance to a tracking mission in a behavior based framework can cause chattering effect when the behaviors switch from obstacle avoidance to tracking and back. Using a weighted sum approach this shattering can be prevented but then blind mixing of obstacle avoidance motion to the tracking motion can create a final motion path, undesirable to both obstacle avoidance and tracking.

On the other hand, more sophisticated motion strategies fuse the obstacle avoidance into the higher level mission by defining the configuration free space due to the obstacles and then planning the mission in this space. Although such a strategy can generate optimal motion strategies, the problem with these techniques is that they require the obstacle geometries to be provided. For a local sensing then the geometrical shapes of the sensed obstacles have to be extracted first. This computation can be very expensive especially in the case of a cluttered environment.

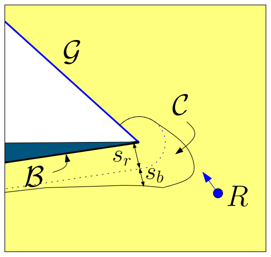


Fig. 6. Obstacle Dilation

We propose a local obstacle avoidance mechanism with a small look-ahead. We do not need to extract the geometric features and using the robot velocity gives an implicit look-ahead by preventing motions that may lead to future collisions. We then use these extended obstacles to constrain the robot motion.

We approximate the robot's size by the radius (s_r) of its bounding circle. Also, depending on the robot's current velocity, there is a finite braking distance (s_b) based on the max deceleration of the robot. If the robot is closer than this distance to the obstacle, the collision is imminent.

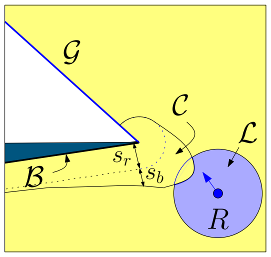


Fig. 7. Obstacle Dilation

Based on this braking distance, s_b and the robot's dimension, s_r, we define 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 the $\mathcal{C}$ 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 $\mathcal{R}$, feasible region then becomes $\mathcal{L} = \mathcal{R} - \mathcal{C}$. As an example, assuming omni-directional motion ability of the robot, Figure 7, $\mathcal{R}$ is a disk of radius, $V\Delta t$, and the darker shaded region shown is $\mathcal{L}$.

Local target recovery

Our algorithm tries to keep the target in view for as long as possible. But many times it is impossible to guarantee this. In case the target steps out of view, the robot can try to retrieve it. This brings the problem into the domain of target searching in an unknown environment. In our case, we try a simple two step strategy of recovering the target, Figure 8. Let the possible position of escape be P and r' be the distance OP. If the target escapes with a high r' that means that it is easier to recover by swinging, On the otherhand when r' is small swinging does not help and the robot should move more towards the occlusion point, O of the escaped edge. Based on this observation we propose a simple recovery velocity as

$$\mathbf{v} = \frac{r\hat{\mathbf{r}} + r'\hat{\mathbf{n}}}{\sqrt{r^2 + {r'}^2}} V$$

We execute this strategy for a fixed number of steps. In case the robot fails to recover the target, the robot moves to eliminate the gap, $\mathcal{G}_i$. Range and FoV edges are dealt with similarly.

However, there might be cases where after reaching the last known target position, the robot either does not find any target or finds multiple targets and does not know which of them is the target to track. At that moment we stop the algorithm. Advanced strategies can be applied in such cases like using bayesian techniques to find the most probable target or choosing the next most probable target's exit.

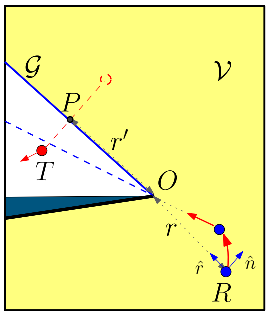


Fig. 8. Local target recovery strategy

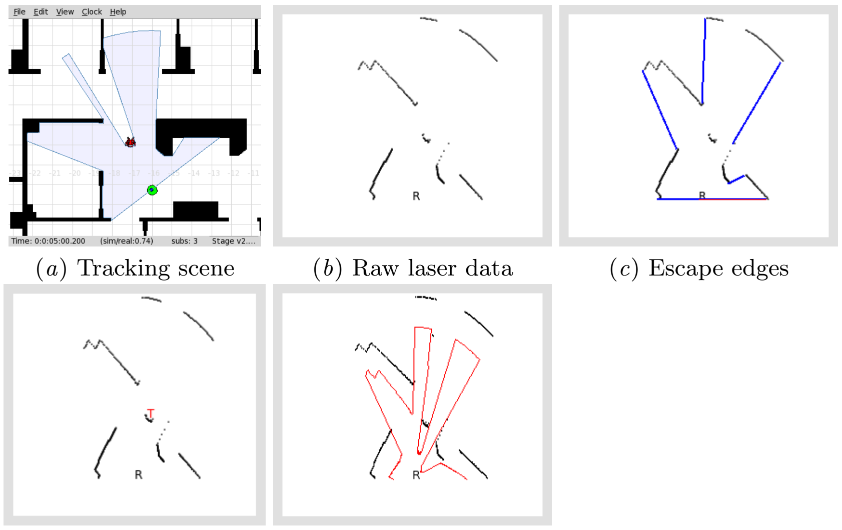
4 Hardware Implementation

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 [3] 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 9a. The green circle is the robot and the red object is the target. Figure 9(b-e), are shown in the local frame of the robot with the origin at R.

Visibility polygon

The output of the sensor is a radial scan of range values (Figure 9b). 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


File Edit View Clock Help
Time: 0:0:05:00.200 (sim/real:0.74) subs: 3 Stage v2....
R
T
(a) Tracking scene (b) Raw laser data (c) Escape edges

(d) Target detection (e) Obstacle Dilation

Fig. 9. Snapshots of the implementation at various stages.

change in adjacent data points. These changes represent the occlusion edges and their location can be extracted from the corresponding data points. Two additional edges are added at the orientation of the min/max angular limits to form the FoV limits. This is shown in Figure 9*c*.

Target identification

The thresholding groups the data points into clusters (Figure 9*b*). 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, and focus on the subsequent target matching. In our experience, we find that a simple nearest neighbor match gives reliable target identification for reasonable target behaviors, even in a crowded environment. We perform the identification as follows.

The clusters are filtered based on average human leg size with some tolerance to give a set of potential target clusters. These potential clusters can be the chair or table legs, pipes or other human legs. Given the target's maximum speed, we can estimate a bound on the target's future position in time Δt, $V' \ \Delta t$. Within this bound, we choose the cluster closest to the target's previous position. False negatives are handled by increasing this bound by a fixed number of steps before declaring the target lost. In fact, this even helps in cases of momentary occlusion when someone walks between the target and the robot.

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. The cluster based target detection is more reliable than shape based feature detection (*e.g.* arc based leg detection) since leg features may partially occlude, eclipse or fuse with each other in the process of walking.

We found that maintaining a list of the non-target clusters in addition to the target using a simple nearest neighbor match, decreases the rate of false positives. Of course, algorithms like EKF, MHT can make the target detection much more robust.

Obstacle avoidance

The clusters computed earlier create motion obstructions in addition to the visibility occlusions. 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 9*e*.

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 [5]). 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 reliability of the target's velocity 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. This improves 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 in a to show why the vantage tracker is able to perform better.

Dynamic Environment Expt

In Figure 11, 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 11*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. This makes the performance of the tracker independent of the dynamic nature of the environment. Later the box stops and the target starts to move (Figure 11*c*) and the tracker is able to successfully follow the target till the end (Figure 11*d*).

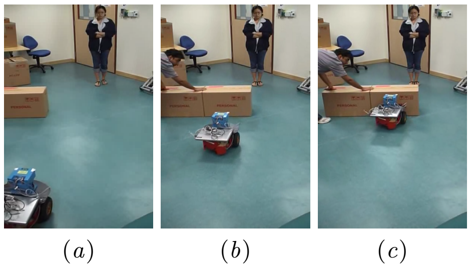


Fig. 10. Visual Servo : Since the robot does not take into account the environment information, it moves straight ahead towards the target (b) and loses the target to the occluding box (c).

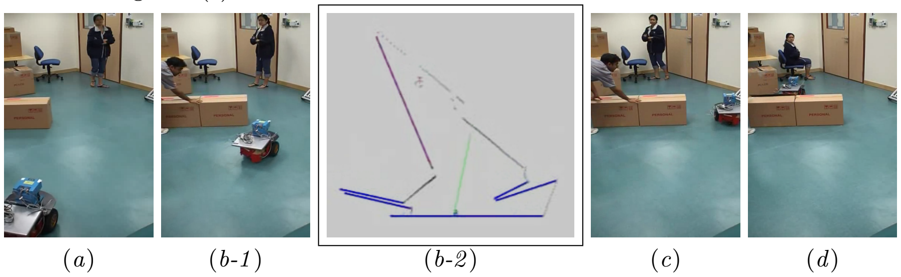
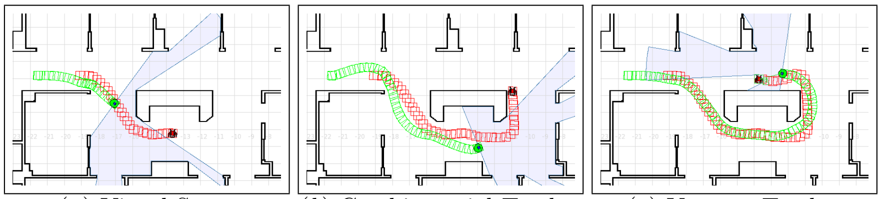


Fig. 11. Vantage tracker : (*b-2*) shows the robot's local perception of the environment. The target is marked by T, the blue lines are the occlusion edges, red line is the most critical occlusion and the green segment starting from R denotes the robot's motion decision. The robot sees the target too close to the occlusion and swings out.

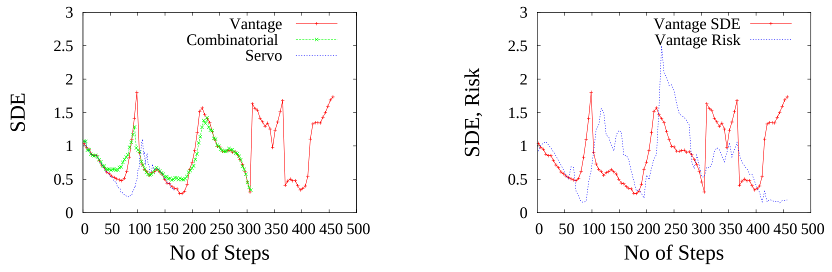
Performance Comparison : Qualitative Study

In (Figure 12 & 13),We compare the performance of the different algorithms : the visual servo, combinatorial tracker (maximizing the shortest distance to escape, SDE) [5] , 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, rather than on real hardware.



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

Fig. 12. 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. 13. (*a*) Plotting SDE for various algorithms, (*b*) Plotting SDE and the risk value for the vantage tracker.

In Figure 13(*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 12*a*,13*a*) and then the combinatorial tracker around #300 (Figure 12*b*,13*a*) where as the vantage tracker manages to continue till the end (Figure 12*c*,13*a*). 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 12*b* 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 than the combinatorial tracker at that time step and is better positioned to handle the occlusion edge. Figure 13*b* plots the vantage risk value over the same path and this plot shows why the vantage tracker performs better. The risk plot 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 14, and compare the them for the percentage of the number of steps in which target was visible. For longer runs, local target recovery modes were activated. Visual servo tracker tried to regain the target by moving directly towards the last position seen, while the combinatorial and the vantage tracker both followed the local recovery algorithm mentioned earlier in this paper.

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.

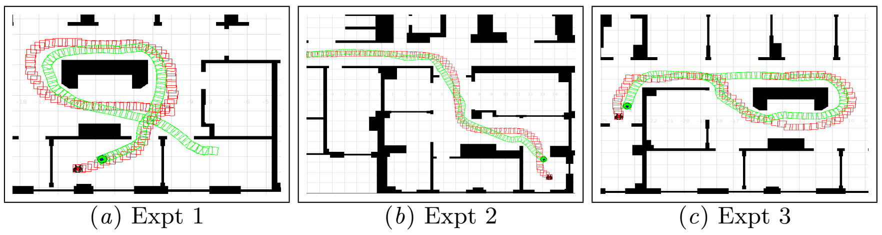

(*a*) Expt 1 (*b*) Expt 2 (*c*) Expt 3

Fig. 14. The simulation experiment paths taken by the target (red) and the path of the vantage tracker (green).

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

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 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.

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.

Acknowledgments The authors would like to thank Prof. Cezary Zelinsky (Warsaw University of Technology, Poland) for initial discussions that led to the physical realization of the tracking algorithm, which was first implemented in his laboratory. The help of Mayank Chandiwal (Indian Institute of Technology) in running extensive experiments is also gratefully acknowledged.

References

1. T. Bandyopadhyay, M. Ang Jr., and D. Hsu. Motion planning for 3-d target tracking among obstacles. In *Proc. Int. Symp. on Robotics Research*. Springer, 2007. To appear.
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. T. Collett, B. MacDonald, and B. Gerkey. Player 2.0: Toward a practical robot programming framework. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005).*, 2005.
4. 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.
5. 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.
6. 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.
7. C.-Y. Lee. *Real-Time Target Tracking in an Indoor Envrionment.* PhD thesis, Dept. of Aeronautics & Astronautics, Stanford University, Stanford, CA, USA, 2002.
8. 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.
9. 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.