Tracking Multiple Moving Objects for Real-Time Robot Navigation

E. Prassler, J. Scholz
Research Institute for
Applied Knowledge Processing (FAW)
P.O. Box 2060, D-89010 Ulm
Germany
Phone: +49 731 501-621
Fax: +49 731 501-999
{prassler,scholz}@faw.uni-ulm.de

A. Elfes
Automation Institute
Informatics Technology Center (CTI)
P.O. Box 6162,13089-500 Campinas, SP,
Brazil
Phone: +55 19 746-6115
Fax: +55 19 746-6167
elfes@ia.cti.br

Abstract

This paper proposes a method for detecting and tracking the motion of a large number of dynamic objects in crowded environments, such as concourses in railway stations or airports, shopping malls, or convention centers. With this motion information, a mobile vehicle is able to navigate autonomously among moving obstacles, operating at higher speeds and using more informed locomotion strategies that perform better than simple reactive manoeuvering strategies. Unlike many of the methods for motion detection and tracking discussed in the literature, our approach is not based on visual imagery but uses 2D range data obtained using a laser rangefinder. The direct availability of range information contributes to the real-time performance of our approach, which is a primary goal of the project, since the purpose of the vehicle is the transport of humans in crowded areas. Motion detection and tracking of dynamic objects is done by constructing a sequence of temporal lattice maps. These capture the time-varying nature of the environment, and are denoted as time-stamp maps. A time-stamp map is a projection of range information obtained over a short interval of time (a scan) onto a two-dimensional grid, where each cell which coincides with a specific range value is assigned a time stamp. Based on this representation, we devised two very simple algorithms for motion detection and motion tracking. The approach is very efficient, with a complete cycle involving both motion detection and tracking taking 6 ms on a Pentium 166Mhz. The system has been demonstrated on an intelligent wheelchair operating in railway stations and convention centers during rush hour.

Keywords: motion detection, real-time motion tracking, multiple moving objects, range images, temporal maps

Submitted to the Journal of Autonomous Robots, Special Issue on Perception for Mobile Agents, expected publication date mid 1999.

1 Introduction

We consider the problem of observing a crowded spatial scene and detecting and tracking the motion of any moving object in the scene such as people, animals, or vehicles. No information is available in advance about the environment and its dynamics. It is not known *a priori* how many objects are present in the scene, nor is there any previous information about the shape of the objects and their dynamics, including their intended motion direction, velocity, acceleration, turning behaviour or inertia. Examples of such environments are shopping malls or concourses in airports or railway stations, where typically tens or hundreds of objects move around. For an autonomous agent to travel through such an environment, both static and moving objects which may affect its own navigation must be identified, and the motion of dynamic objects must be tracked to allow the identification and avoidance of potential collisions.

The motion detection and tracking problem becomes even more difficult when we assume that the motion sensor is not stationary but also moves around while observing the environment. This leads to a situation where apparent changes in the agent's perception of the environment may not only be due to the actual motion of objects, but also to a change in the point of view of the sensor. Distinguishing between actual moving objects and previously unknown stationary objects that are being seen from a new viewpoint is not a trivial problem, but is essencial when it comes to planning collision-free locomotion through crowded, dynamic environments.

We are studying the problem of motion detection and tracking of multiple dynamic objects in the context of the MAid research project[14]. This project focusses on the development of a guidance system for an intelligent wheelchair. This guidance system should enable the wheelchair to manoeuver autonomously in a shopping mall or a railway station and cross an area with many moving people, and is particularly targeted towards disabled persons who have lost most of their motion control capabilities and are therefore physically unable to effectively operate an electrical wheelchair. This application imposes several design requirements on our system, including in particular the ability to perform real-time obstacle avoidance combined with motion detection and tracking, and the use of off-the-shelf computer and sensor components to achieve this real-time performance.

2 Related Work

A large body of work in real-time tracking of moving objects exists in the field of computer vision. The approaches can typically be subdivided into *model-based* methods, which require a substantially precise object model, and *motion-based* methods. The latter group can be further divided into *optical flow* and *motion energy* methods.

Model-based object tracking methods allow us to track the object's translational and rotational motion [16], but can only be applied if a geometric description of the objects to be tracked is available. Motion-based methods use information about motion in the image to infer a corresponding motion in the real world. A well-established approach to determining image motion is the computation of the *optical flow* [9]. Since this is a computationally expensive and often ill-posed problem, many variants of optical flow have been explored. An example is SSD-optical flow [13], which requires the depth information, or distance to the object, to be known.

Motion energy methods in general use the image derivative to detect regions of activity. In most cases, image subtraction is used to approximate the derivative, and is combined with additional information like the movement of edge locations [10, 11] or dynamic background recovery [4].

Problems arise when the camera itself moves and the apparent image motion induced by the camera motion must be separated from that induced by the motion of the objects in the field of view of the camera. While this can be accomplished for pure rotational movements of the camera [11], translational movements are much more difficult to handle. A recent approach by Nordlund and

Uhlin [12] addresses this scenario, but is limited to one moving object in the scene. The approach is also computationally very expensive.

A different class of approaches to tracking moving objects employs the data provided by range sensors instead of video images. Much of the work in this area deals with target tracking in sonar or radar scans [2]. Kalman filtering is a widely used technique for these applications.

In our work, we have chosen to sidestep many of the problems that arise in motion detection and tracking using visual imagery by handling range data directly, as provided by a 2D (planar) laser range scanner. Additionally, since human purposive locomotion cannot be easily characterized, at the observational time scales and resolutions we are interested in, using dynamic models, we have also chosen not to use a Kalman filter approach, as we would not be able to take full advantage of the power of the Kalman filter. Instead, we have chosen to approach our motion detection and tracking problem by performing spatio-temporal matching and clustering of individual range points in consecutive range images using a *nearest–neighbor* (NN) metric [15].

3 Fast Motion Detection and Tracking

The fundamental approach to identifying temporal changes in the surrounding environment of an autonomous agent is to consider a temporal sequence of spatial observations or snapshots and to determine how these observations differ from each other. A discrepancy between two temporally subsequent snapshots is a strong indication of a potential change in the environment. This discrepancy can be caused either by new information that has been obtained about unknown, stationary objects due to the egomotion of the observer, by the inter-observation motion of an already discovered dynamic object, or by a combination of the two previous causes. To discriminate between these cases, our approach performs sensor motion estimation together with motion detection and tracking. Since this in itself is not enough to inequivocally identify moving objects, we accumulate sensor-based information in a spatio-temporal model called the time-stamp map to allow robust motion detection and tracking.

In outline, our approach to motion detection and tracking is based on the following components:

- Range-based sensing of the environment of operation
- Estimation of sensor motion (or robot egomotion)
- Representation of time-varying information using time-stamp lattice maps
- Motion detection and tracking using spatio-temporal clustering applied to the time-stamp maps

We proceed to discuss each of the components of the approach.

3.1 Sensors for Fast Motion Detection

When considering the problem of navigation in crowded dynamic environments, we evaluated a variety of sensors for our task: imaging cameras, laser rangefinders, sonar, infrared, and tactile sensors. Not all of these sensors are equally useful for detecing and tracking motion. Infrared and tactile sensors can be excluded because of their limited range. Sonar has a very limited angular resolution and in our case provides sensor images which are inadequate to reliably track a number of distinct moving objects. This reduced the selection of an appropriate type of motion tracking sensor to a camera or a laser rangefinding system.

Although there is quite a body of literature on motion detection in sequences of video images, we chose to use a laser rangefinder. Our choice is motivated as follows. Standard methods for motion detection in video images such as *optical flow* or *motion energy* allow us to infer motion in

the image plane. It is not straightforward, however, to compute from this the motion parameters of a moving object in 3D space, including its motion direction, velocity, and acceleration. Stereo vision, which allows us to compute the position of an object in three-dimensional space, is currently still computationally too expensive and not robust enough for rapidly changing environments. A laser rangefinder has the significant advantage of providing range imagery directly, and for our application has shown itself to facilitate the development of an easy, robust and computationally inexpensive approach to motion detection and extraction for multiple moving objects.

The sensor which we have selected for our application is a SICK PLS 200 laser rangefinder. The device is a time-of-flight scanner. It has a maximum range of $d = 50\,\mathrm{m}$ with an accuracy of $\sigma_d \approx 50\,\mathrm{mm}$. The sensor has an angular range of $\pm 90\,\mathrm{deg}$ (relative to heading) with a resolution of $0.5\,\mathrm{deg}$. Consequently, a full scan comprises 361 range measurements. The device is currently operated at a frequency of 3 Hz, providing three range scans per second.

3.2 Estimation of Sensor Motion

The range data provided by the laser rangefinder are registered to the local frame of reference attached to the sensor. To be able to compare two temporally subsequent local range images, and consequently to compute a differential image, we have first to transform the scans into the same frame of reference. For this, it is necessary to know precisely the motion executed by the sensor between two scans: what distance it has moved from one viewpoint to the next (translational information), and how far it has turned (rotational information). In stationary environments, the information about sensor translation and rotation may be inferred from the sensor images themselves by some type of matching algorithm. In time-varying environments such an approach becomes far more difficult to apply.

In the research presented here we are not primarily interested in building high-precision maps that cover extensive geographical areas, but are rather concerned with identifying local spatial changes that occur in spatially and temporally limited windows of observation. This means that we need to have good registration accuracy among scans that are spatially and temporally close, but do not require precise registration among sensor observations that are spatially or temporally distant. Consequently, long-term position estimation drift is not a primary concern, and this has led us to opt for obtaining the sensor translation and rotation information from the odometric data provided by the vehicle itself. The current position and orientation of the vehicle, and consequently of the sensor, are obtained from the *dead-reckoning* system that our experimental wheelchair system is equipped with. This dead-reckoning system enables the wheelchair to keep track of its position and orientation over a limited travel distance with reasonable accuracy. In our experiments, we measured a positional error of $\sigma_{x,y} \approx 15\,\mathrm{cm}$ after a traversal of $10\,\mathrm{m}$.

The information from the dead-reckoning system is used to label each range scan with a stamp describing the sensor pose (position and orientation) at the observation position. With this information, it is straightforward to transform local range images from earlier scans into the current frame of reference, or vice-versa.

3.3 Representation of Time-Varying Environments

The next question to consider is what kind of representation of the sensor information is appropriate for motion detection and estimation.

Several reasons countermand the direct utilization of the raw range data. Sensor range measurements are affected by sensor accuracy and noise, as well as by target detection uncertainty[6], in particular by what are called type I and II errors in the radar literature[3]). Even with a static sensor operating in a static environment and with no changes in measurement conditions, these operational characteristics of the sensor lead to variations among successive scans. The overall

sensor uncertainty affects all further operations on the range data, and specifically disturbs the reliable detection and estimation of target motion.

Furthermore, if we work directly with the raw range data, what we have to deal with are essentially noisy point sets in the 2D (horizontal) plane. Algorithms for correlating and matching point sets are computationally expensive; minimizing the Hausdorff distance for two point sets, for example, has a complexity worse than $O(nm^4)$ [1], where n and m are the number of points in the two point sets to be matched. With such computational complexity, the algorithm may not be adequate for a real-time application.

These considerations have led us to employ an alternative representation for range imagery of time-varying environments which seems to be more suitable for motion detection and estimation. Our representation is an adaptation of the Occupancy Grid and Inference Grid models[7, 8]; it involves the projection of the range scans on a two-dimensional rectangular lattice, and the annotation of each cell with the time-stamps of the measurements that fall into it. In the lattice, each grid element corresponds to a small spatial region in the robot world.

While investigating the performance of Occupancy Grid-based mapping procedures such as suggested in [5], we noticed that a significant amount of time is spent in mapping free space. The further away the observed objects are, the larger is the free space between the sensor and the objects to be updated in the map. Initialization of the Occupancy Grid, for grids with typical sizes of several tens of thousands of cells, can also be an expensive, even though one-time, operation. While mapping large areas of free space can be important for in a variety of spatial mapping and navigation scenarios, it is less important in the context of this paper, with its emphasis on short-term detection and tracking of moving objects.

In the representation discussed in this paper, we only update the information in those map cells which were observed as occupied by a given scan executed at time t, while all other cells in the grid remain untouched. We call this representation, which simultaneously encodes *occupancy* and *time of observation*, a *time-stamp map.*

Compared to the incorporation of a range image into an Occupancy Grid, the generation of a time-stamp map is a simpler and computationally less demanding procedure. Mapping of a range measurement involves only a single step: the cell that corresponds to the target detection event given by a single range measurement executed at time t is labelled using this time stamp t. The stamp means that the cell was observed as being *occupied at time t*. No other cells are updated during this operation; in particular, we do not mark as *free* any cells that lie between the sensor position and the cell corresponding tof the range measurement.

The time-varying characteristics of the environment are captured by the sequence TSM_t, TSM_{t-1}, ..., TSM_{t-n} of those time-stamp maps. An example of such a sequence is shown in Fig. 1 a) - c). The pictures show three snapshots of a simple time-varying environment, with one moving and one stationary object in a time-stamp map representation. The age of the observation is indicated using different gray levels, with darker regions indicating more recent observations. Note that the maps are already aligned, so that they have the same orientation. A translation by the corresponding position offsets transforms the maps into the same frame of reference. The aligned maps are shown in Fig. 1 d).

In the experimental work discussed later in the paper, the incorporation of a full range scan (361 measurements) into a 200×200 time-stamp map takes 1.5 ms on a Pentium 166Mhz.

3.4 Fast Motion Detection

In the work described here, we have implemented a very simple approach to motion detection using a sequence of time-stamp maps. We consider the set of cells in TSM_t which carry a time stamp t (*occupied at time t*) and test whether the corresponding cells in TSM_{t-1} were occupied too, i.e., carry a time stamp $t-1$. If corresponding cells in TSM_t, TSM_{t-1} carry time stamps t and $t-1$, respectively, then we interpret this as evidence that the spatial region circumscribed by

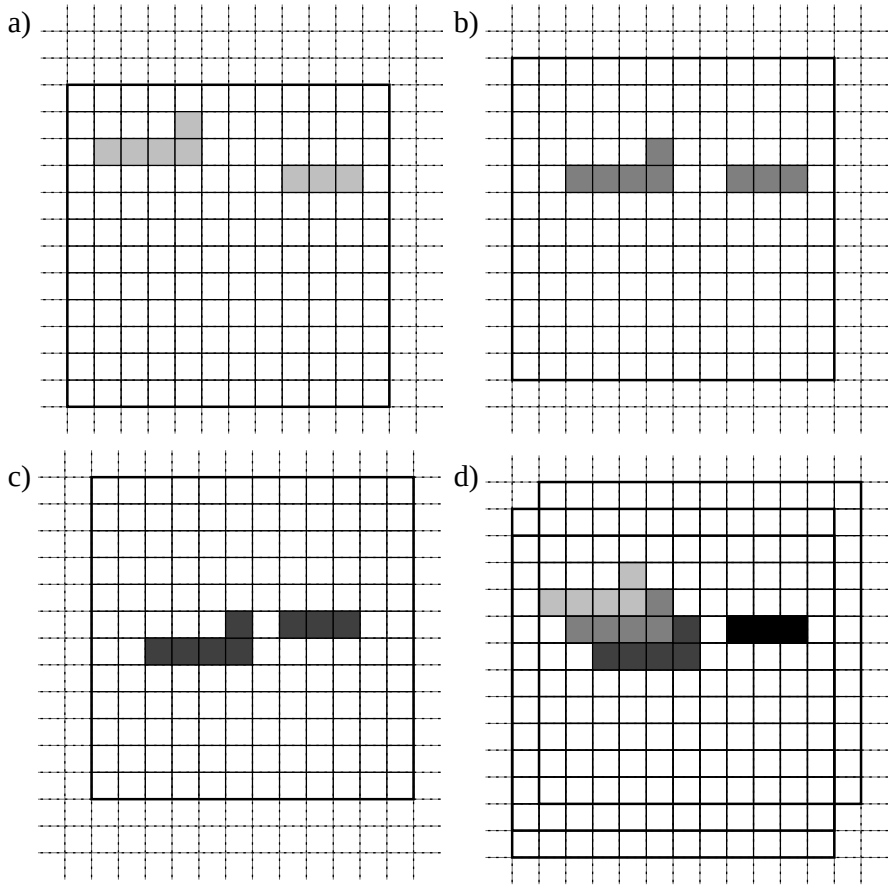


Figure 1: A sequence of time-stamp maps describing a simple environment. The age of the observation is indicated using different gray levels. More recent observations correspond to darker gray levels in the object contours.

these cells has been occupied by a stationary object. If, however, the cells in TSM_{t-1} carry a time stamp different from $t-1$ or no time stamp at all, then the occupation of the cells in TSM_t must be due to a moving object.

Note that here we consider only the two most recent maps, TSM_t and TSM_{t-1}, for detecting moving objects. This limits our motion detection resolution, since objects that move very slowly as compared to the sensor scan rate will not be detected as moving, and is known in the computer vision context as the aperture problem.

Furthermore, we do not consider all cells in TSM_t, but confine our search to those cells which carry a stamp *occupied at time t*. To make the search for moving objects as efficient as possible, any cell *occupied at time t* is additionally entered into an ordered list. This list can be updated while a range image is being mapped into a time-stamp map, and allows a more efficient access to the occupied cells in the maps. Also, to keep false moving target detection events at a low rate, we do not track single cells that are apparently moving, but group ensembles of coherently moving cells into distinct objects. This clustering corresponds to a kind of space-time filtering operation, and is implemented using a simple algorithm which investigates the adjacency of cells. The aperture problem also becomes evident here, since objects that move at very high rates as compared to the data acquisition rate of the sensor will show up as "blips" in the map and may be dismissed as noise.

The above motion detection algorithm is described in pseudo-code notation in Table 1.

As we pointed out earlier, the time-stamp representation of a time-varying environment is more

```
procedure detectMotion;
  for each cell ensemble cs_{x,t} describing an object x in TSM_t
    for each cell c_{i,t} in cs_{x,t}
      for each corresponding cell c_{i,t-1}, ..., c_{i,t-k}, ..., c_{i,t-n} in
            TSM_{t-1}, ..., TSM_{t-k}, ..., TSM_{t-n}
        if c_{i,t-k} carries a time stamp t - k
          then c_i is occupied by a stationary object
          else c_i is occupied by a moving object
  if majority of cells c_{i,t} in cs_{x,t} is moving
    then cell ensemble cs_{x,t} is moving
    else cell ensemble cs_{x,t} is stationary
```

Table 1: A motion detection algorithm based on a sequence of time-stamp maps.

efficient for motion detection than commonly used Occcupancy Grid representations. In particular, the time-stamp representation allows us to use a sequence of maps in a round-robin mode, without having to clear and initialize the map which is being updated with the new sensor image. Outdated time stamps which originate from the mapping of previous sensor images do not have to be deleted but are simply overwritten. This seems to be an inappropriate procedure because the map which is supposed to receive a new sensor image may be polluted by outdated information. This approach is, however, not only efficient — as we avoid an expensive initialization operation — but is also correct. Cells which are marked by an outdated time stamp are simply considered as free space, which has the same effect as assigning some default value.

The time-stamp representation has one disadvantage, though. As such, it does not distinguish between the discovery of a new stationary part of the environment and the observation of a moving object. This means that, if we discover a new stationary object in an area which was formerly occluded, the above algorithm falsely reports a motion in that area. This is not a severe problem, however, since at the next time step the new object is already recognized as stationary. In fact, this effect is closely linked to the window effect that exists whenever motion estimation is performed.

It should be further mentioned that the motion detection algorithm described above only infers the motion of the observable part of a moving object, in other words, of that part of the object which is moving along the direction of motion. This is due to the fact that we do not attempt to reconstruct a full spatial description of a moving object based on multiple views of the object in motion. For our application, this is not a cause of concern, since it neither affects the tracking of a moving object (as we will see below), nor does it affect the computation of a collision-free path.

3.5 Motion Tracking

After sets of cells have been identified as being part of a moving object (the motion detection problem), we need to track the object over time, thereby determining its trajectory through space (the motion tracking problem).

As we mentioned earlier, the target tracking methods that are used in aerospace, marine and terrestrial applications typically employ some model of the dynamics of the moving target(s). This entails knowing target parameters such as object mass, acceleration and braking capabilities, kinematic constraints, and so forth.

In contrast, consider the task of detecting and tracking walking people in a crowded train station

concourse. Obviously, none of the parameters necessary to set up an analytic dynamical model of the moving objects in the scene are available. Although kinematic and dynamic models of human walking motions and gaits have been developed, there are no analytical models of human purposive locomotion, which would allow us to make inferences about the motion of a person over longer distances and periods of time.

Consequently, our approach to motion tracking is to collect information about its past motion and to extrapolate this past motion into the near future, if necessary. For this purpose, we consider the sequence of recent sensor scans and extract information about the motion direction, velocity, and acceleration of the moving objects. This motion history is obtained from the spatial changes that occur in the spatio-temporal maps of these sensor images.

As we saw earlier, in our approach to motion detection it is sufficient to compare only two time-stamp maps, provided that the dynamic objects move at adequate speed. However, computing the motion history of the various dynamic objects in the scene requires the use of more extended sequences of sensor images. We start by assuming that the cells which describe distinct objects have already been grouped into cell ensembles, and that these cell ensembles and the corresponding objects have been classified either as *moving* or as *stationary* by the motion detection algorithm described above.

The first step in computing the motion history of a moving object is to identify the same object in a sequence of scans or maps. Once we have solved this correspondence problem, it is relatively straightforward to estimate the motion direction and the velocity of a dynamic object from its observed positions. To determine the correspondence between multiple observations of the same object in successive scans, we use a nearest-neighbor criterion applied to the time-stamp maps. The nearest-neighbor criterion is defined over the Euclidean distance between the centers of gravity of cell ensembles representing distinct objects. For each cell ensemble representing an object at time t, we determine the nearest cell ensemble in terms of the Euclidean distance in the map describing the environment at the preceding time step $t-1$. Obviously, this operation requires the objects to be represented in the same frame of reference.

If the distance to the nearest neighbor is smaller than a given threshold, we assume that both cell ensembles describe the same object. The threshold depends on whether the considered objects and cell ensembles are stationary or moving. To establish the correspondence between two cell ensembles describing a stationary object, we choose a rather small threshold, since we expect the cell ensembles to have very similar shapes and to occupy the same space. To determine the correspondence between cell ensembles describing a moving object, the threshold used is correspondingly larger. In the experimental results discussed below, we have used a threshold of 30 cm for stationary objects, and a threshold of 1 m for moving objects, since this is approximately the distance which a person moving at fast walking speed covers between two sensor scans. A complete cycle involving both detecting and tracking any moving objects takes approximately 6 ms on a Pentium 166MHz.

A description of the above algorithm in pseudo-code notation is given in Table 2.

Two considerations concerning the performance of this approach are in order. Firstly, computing the correspondence between the centers of gravity of cell ensembles is a very simple and efficient, but not extremely reliable, method for object tracking. Assume, for example, that the view of an object has changed between two observations due to occlusion effects. If the parts of an object which are perceived at time t differ considerably from the parts which are perceived at time $t-1$, considering the centers of gravity is not sufficient and our tracking algorithm may fail to establish the correct correspondence. The consequence is that the track that should correspond to a single moving object may be broken up in two or more tracks. For real-time efficiency reasons, we accept this potential limitation of our approach, and consciously refrain from using shape matching methods. These could improve the target tracking reliability, but are also more time consuming.

Secondly, we note that the use of thresholds is an additional source of potential errors. Irrespective of the threshold values chosen, there will always be cases where the parameter choices are

```
procedure findCorrespondence;
  for each object o_{i,t} in TSM_t
     for each object o_{j,t-1} in TSM_{t-1}
        CorrespondenceTable[i,j] = corresponding(o_{i,t}, o_{j,t-1});

function corresponding(o_{i,t}, o_{j,t-1});
  if o_{i,t} is stationary and o_{j,t-1} is stationary
     then ϑ = ϑ_s;  (threshold for stationary objects)
     else  ϑ = ϑ_m; (threshold for moving objects)
  if d(o_{i,t}, o_{j,t-1}) < ϑ
        and not_exists o_{k,t} :   d(o_{k,t}, o_{j,t-1}) < d(o_{i,t}, o_{j,t-1})
        and not_exists o_{l,t-1} : d(o_{i,t}, o_{l,t-1}) < d(o_{i,t}, o_{j,t-1})
     then return true;
     else  return false;
```

Table 2: An algorithm for tracking moving objects in a crowded environment.

inappropriate.

More specifically, we distinguish between two types of errors which may occur while computing the correspondence between objects in successive sensor images. First, a correspondence may have been computed between two cell ensembles which do not correspond to each other, and so do not refer to the same actual object. Second, there may be cell ensembles in the world time-stamp map at time t for which no corresponding cell ensembles are found in the time-stamp map for time $t-1$.

The first type of error occurs if two cell ensembles which actually correspond to each other do not satisfy the nearest-neighbor criterion; that is, the cell ensemble which would actually correspond to the ensemble being considered is farther away than a cell ensemble which represents a physically different object. The probability of this target crossover error depends on the velocity of the objects and on the density of the objects in the environment: the faster the objects and the more crowded the environment, the higher the likelyhood that such a false association may occur. Again, the probability of such false correspondences may be reduced by more sophisticated shape matching methods, but cannot be eliminated due to the limited shape information provided by a 2D range image.

The second type of error may have several reasons. One possibility is that the considered cell ensemble describes an object which has just entered the visibility range of the sensor at time t, and which was therefore not visible at time $t-1$. In this case, we will naturally not find the corresponding cell ensemble in the time-stamp map computed at time $t-1$. Another possibility has to do with the object under consideration moving at high speed, so that its distance to its previous position at time $t-1$ exceeds the threshold discussed earlier. This is the aperture problem referred to above, and the corresponding error rate can be diminished by increasing the value for this maximum distance. However, raising this threshold also raises the risk of false associations or target crossover errors. Finally, a third possibility is a situation where the nearest-neighbor criterion has previously led to a false association, and where subsequently the cell ensemble at time $t-1$ which would actually correspond to the one being considered has already been erroneously associated with a different object.

In spite of the potential errors discussed above and the simplicity of the approach, we have

found in our experimental work that the algorithms presented above for motion detection and target tracking have proven to be very fast and efficient, and also to work in a sufficiently reliable manner, as we will see in the following section.

4 Experimental Results

Our approach was tested and validated in two sets of experiments. The first set took place under relatively simplified and controlled lab conditions. The sensor (mounted on the wheelchair) was placed at a fixed position in an experimental area, with a size of approximately $4 \times 7\text{m}^2$. The environment did not contain any stationary objects, except for the walls. The experiments were recorded by a video camera which was synchronized with the rangefinder and the motion detection and tracking algorithm. In each experiment we asked a number of people to move along prescribed trajectories in the experimental area at comfortable walking speed.

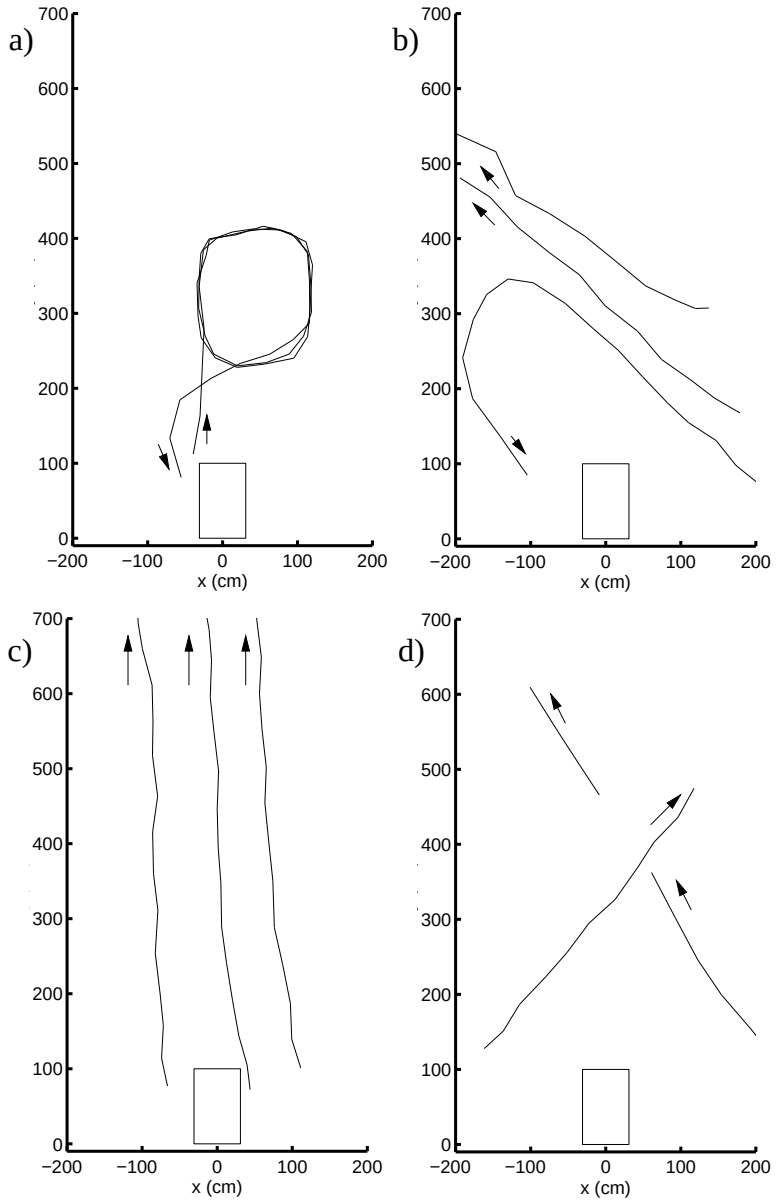


Figure 2: Motion tracking under simplified lab conditions.

The results of this first set of experiments are shown in Fig. 2 a) - d). In the first experiment, a test person walked along a given rectangular trajectory in the front area of the sensor. After several laps along this trajectory the test person headed back to its initial position. Our algorithm tracked the motion of the test person in real time without any problems. The trajectory estimated by the tracking algorithm is shown in Fig. 2 a).

In the experiment shown in Fig. 2 b) three people moved along approximately parallel straight lines away from the sensor, with one person making a sudden left turn and heading back to the sensor. The people moved at slightly different velocities, so that none of the targets was occluded during the experiment. As the figure indicates, the algorithm could easily track the motion of the three persons. In the experiment in Fig. 2 c) we tracked again the motion of three test persons which moved along more or less parallel straight lines away from the sensor. This time the people moved at a similar speed.

The last experiment in this series deserves a more detailed discussion. We had two test persons move along straight lines which subsequently crossed each other. Consequently, for a short period of time one person was occluded by the other from the viewpoint of the sensor. Our algorithm was unable to track the occluded person during this period of time. In Fig. d) this can be seen by the interruption of one of the trajectories. Our algorithm lost the person for two time steps. It detected and resumed tracking the motion after the person became visible again. Tracking moving objects whose trajectories cross each other is a very general problem (see [2]) and is not a specific problem of our algorithm. Problems of this type cannot be eliminated even by more sophisticated methods such as described in [2], which assume that there is a model of the motion of the objects which are to be tracked. As mentioned above, we have to refrain from such an assumption since those models are not available in our application.

In a second series of experiments we tested the motion detection and tracking algorithms under more difficult and realistic conditions. The environment chosen for these experiments was the waiting hall of a train station during the morning rush-hour. Snapshots of the environment are shown in Fig. 3 a) - d). The pictures show the wheelchair, which in this case was manually steered through the test environment during the experiment. The sensor is mounted on the front side of the wheelchair and is not visible in the pictures. The number of people standing or moving in the waiting hall varied between 50 and 200, while the number of people in the vicinity of the wheelchair and captured by the sensor varied between 5 and 30.

The results of one of the experiments are shown in Fig. 4. The left column in the figure shows a sequence of superimposed range images. Each composite range image actually consists of six consecutive range images. Next to each composite range image we show in the right column the trajectories of the moving objects which were tracked in these six range images. Stationary objects which appear as bold point sets in the superimposed range images are not shown in the plots on the right side.

It should be noted that effects such as temporary occlusion of objects, crossing trajectories and alike, which we already faced in the lab experiments, occur far more often in the waiting hall experiment. Additionally, we observed that people often stopped suddenly in front of a counter or a display to study departure times, for example, or even in front of our wheelchair to express their sympathy with the supposedly handicapped driver of this strange vehicle. This explains why some tracks shown in the pictures in the right column appear quite erratic while others describe more or less smooth straight motions.

5 Conclusion

We presented an approach to tracking a large number of moving objects in crowded environments, such as concourses in train stations, airport terminals, or shopping malls. The approach uses the range data provided by a 2D laser rangefinder. This sensor was chosen to facilitate the



Figure 3: MAid traveling in the concourse of a railway station.

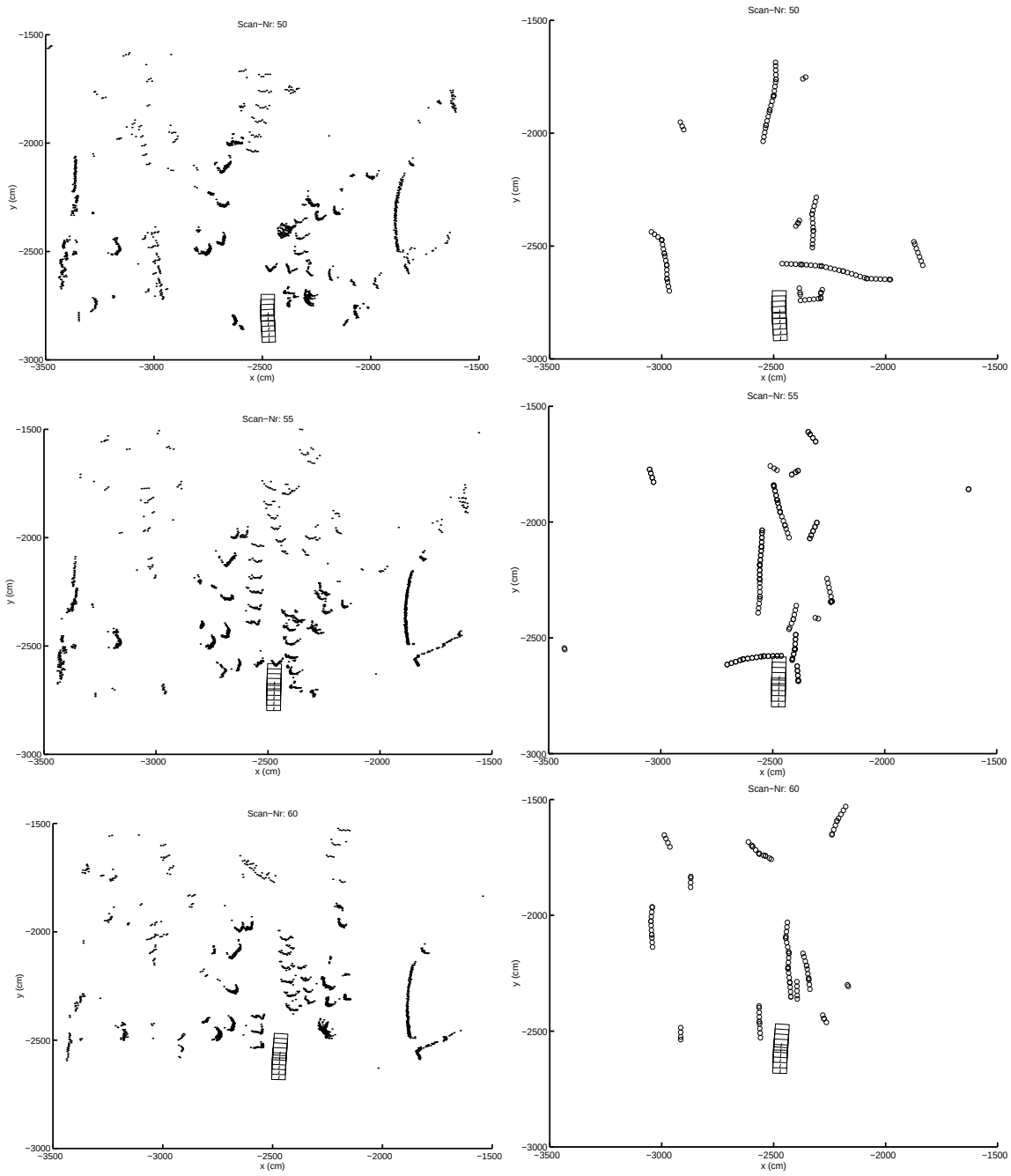


Figure 4: Tracking moving objects in the concourse of a railway station.

real-time performance of the tracking system. By using a laser rangefinger our approach differs from other work in motion detection and tracking which are based on visual information.

The time-varying characteristics of an environment are captured by a sequence of temporal maps, which we called time-stamp maps. A time-stamp map is a simple projection of a range image onto a two-dimensional grid, where each cell which corresponds to a specific range value is assigned a time stamp. The major advantage of this representation is that it is very efficient.

Based on this representation we have devised two very simple algorithms for motion detection and motion tracking. One complete cycle involving both motion detection and tracking takes 6 ms. The approach was successfully applied to a set of range images recorded in the waiting hall of a train station during the morning rush-hour. In the experiments we tracked between 5 and 30

moving objects concurrently and in real-time.

Our approach to motion detection and tracking does not pressuppose the existence of dynamic models of purposive human locomotion. Those models are not available for the kinds of operating environments which we have considered in our work. Given the short cycle time, however, our approach can definitely be considered "fast", and is able to deliver the required real-time capability.

Due to occlusion effects, our tracking method may temporarily loose a moving object. Occlusion effects, crossover track problems, or splitting of object tracks are, however, very general problems in motion tracking and are not specific to our approach.

Future work will address issues such as the integration of fast matching methods into the approach to disambiguate moving objects based on their shape. We also plan to extend our work by incorporating a motion prediction algorithm to cope with the problems caused by occlusion or by crossover tracks, and also to facilitate motion planning in time-varying environments.

Acknowledgment

This work was supported by the German Ministry for Education, Science, Research, and Technology (BMB+F) under Grant No. 01 IN 601 E 3 as part of project INSERVUM.

References

[1] H. Alt, B. Behrends, J. Blömer. Approximate Matching of Polygonal Shapes. In *Proc. of the 7th Ann. ACM Sypm. on Computational Geometry*, 1991.

[2] Y. Bar-Shalom, T.E. Fortmann. *Tracking and Data Association.* Academic Press, 1987.

[3] J.L. Eaves, E.K. Reedy. *Principles of Modern Radar.* Van Nostrand Reinhold, New York, 1987.

[4] Q. Cai, A. Mitiche, J. K. Aggarwal. Tracking Human Motion in an Indoor Environment. In *Proc. Int. Conf. on Image Processing*, 1995.

[5] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation.* PhD thesis, Electrical and Computer Engineering Department/Robotics Institute, Carnegie-Mellon University, 1989.

[6] A. Elfes. Dynamic Control of Robot Perception Using Stochastic Spatial Models In *Information Processing in Mobile Robots*, G. Schmidt (ed), Springer-Verlag, Berlin, 1991.

[7] A. Elfes. Multi-Source Spatial Fusion Using Bayesian Reasoning In *Data Fusion in Robotics and Machine Intelligence*, M. A. Abidi and R. C. Gonzalez (eds), Academic Press, San Diego, 1992.

[8] A. Elfes. Robot Navigation: Integrating Perception, Environmental Constraints and Task Execution Within a Probabilistic Framework In *Reasoning With Uncertainty in Robotics*, L. Dorst, M. van Lambalgen and F. Voorbraak (eds), Springer-Verlag, Berlin, 1996.

[9] B.K.P. Horn, B.G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17, pp. 185-203, 1981.

[10] Y. Mae, Y. Shirai, J. Miura, Y. Kuno. Object Tracking In Cluttered Background Based On Optical Flows And Edges. In *Proc. of the Int. Conf. on Pattern Recognition*, pp. 196-200, 1996.

[11] D. Murray, A. Basu. Motion Tracking with an Active Camera. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(5), pp. 449-459, 1994.

[12] P. Nordlund and T. Uhlin. Closing the Loop: Detection and Pursuit of a Moving Object by a Moving Observer. *Image and Vision Computing*, 14, pp. 265-275, 1996.

[13] N.P. Papanikolopoulos, P.P. Khosla, T. Kanade. Visual Tracking of a Moving Target by a Camera Mounted on a Robot: A Combination of Vision and Control. *IEEE Trans. on Robotics and Automation*, 9(1), pp. 14-35, 1993.

[14] E. Prassler, J. Scholz, M. Strobel. MAid: Mobility Assistance for Elderly and Disabled People. In *Proc. of the 24th Int. Conf. of the IEEE Industrial Electronics Soc. IECON'98* (to appear), 1998.

[15] Y. Rong Li, Y. Bar-Shalom. Tracking clutter with nearest neighbor filters: analysis and performance. *IEEE Trans. Aerospace and Electronic Systems*, 32, pp. 995-1010, 1996.

[16] P. Wunsch, G. Hirzinger. Echtzeit–Lagebestimmung dreidimensionaler Objekte aus Bildsequenzen zur Lagebestimmung eines Industrieroboters. In B. Mertsching: *Proceedings in Artificial Intelligence*, pp. 169-173, 1996.