

Fast and Robust Tracking of Multiple Moving Objects with a Laser Range Finder

Boris Kluge Christian Köhler Erwin Prassler
{kluge, koehler, prassler}@faw.uni-ulm.de

Research Institute for Applied Knowledge Processing,
Helmholtzstr. 16, 89081 Ulm, Germany

Abstract

In this paper we focus on the task of tracking multiple moving objects in rapidly changing, dynamic environments. Objects are extracted from laser range finder images and correspondences between successive images are established by network optimization techniques. The approach is implemented on a robotic wheelchair, used in two applications and evaluated experimentally.

Keywords: robot sensing, object tracking, dynamic environment, laser range finder, network flows, combinatorial optimization.

1 Introduction

Today most existing robot systems are not designed to cope with rapidly changing, dynamic environments. Usually they are programmed once and then perform the same motion many thousand times. If there is an unforeseen situation, like a human entering the working cell of an assembling robot or people crossing a mobile robot's desired path, at best the robot stops and possibly tries to evade this obstacle in order not to hurt anybody. On the other hand, such a robot might be easily obstructed by a person playing jokes on it. So this lack of awareness is a serious obstacle to a wide spread use of such systems, and a basic requirement to this awareness is continuous observation of objects surrounding the robot.

1.1 Related Work

Tracking human motion with computer vision is an active field of research [Moe99], most approaches using video image sequences. Range images are used for example in intelligent vehicles or driver assistance systems [MA98, SB98]. An object tracking system using

a laser range finder has been implemented at our institute previously [PSSS98].

Matching multiple moving point objects by computing minimum weight matchings in bipartite graphs is mentioned in [AMO93] as an application example. Our network optimization approach is inspired by this application example and the relationship between matchings and flows in bipartite graphs [AMO93, Law76].

1.2 Overview

Tracking of objects around the robot is performed by repeated execution of the following steps. At the beginning of each cycle a scan image of the environment is taken. This scan is segmented (section 2.1) and further split into point sets representing objects (section 2.2). As our matching approach is based on graph theory some notions are reviewed (section 3.1) before the algorithm itself is presented (section 3.2). Next we present detection of deliberate obstructions by humans (section 4.1) and accompanying a guiding person (section 4.2) as applications. Results of our experiments are shown (section 5) and discussed (section 6) before concluding the paper (section 7).

2 Finding Objects

Objects are extracted from each laser scan image by two heuristic steps. At first the scan is segmented into densely sampled parts. In the second step these parts are split into subsequences describing "almost convex" objects.

2.1 Scan Segmentation

A scan image is given by a sequence of samples representing the distances from the range finder to closest

opaque obstacles in the horizontal half plane in front of the robot. The samples in this sequence are angularly ordered counterclockwise with respect to the position of the range finder.

Consider two adjacent samples in this sequence. Intuitively, if they are taken from the same obstacle, their distances to the range finder will be similar. On the other hand, if one of these samples is taken from a different obstacle which is located in front of or behind the other sample's obstacle, we will encounter a significant difference in those two distances. Thus it is plausible to use such distance gaps as split positions for the sequence of samples in order to find distinct objects. A threshold value is chosen in advance for the maximum allowed distance gap.

2.2 Object Extraction

The subsequences of sample points yielded by the preceding step are divided further, as we would otherwise encounter problems with objects situated close to other objects, for example humans leaning against or walking close to a wall or other humans. We assume that most objects of interest in the robot's environment are either almost convex or can be decomposed into almost convex sub objects in a reasonable way.

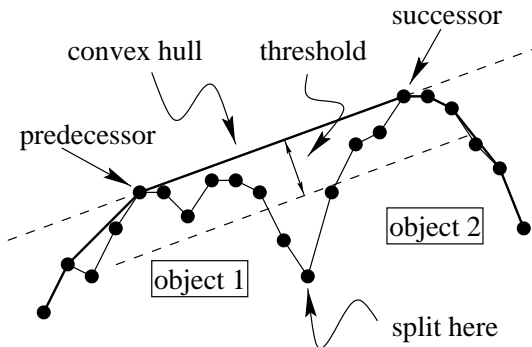


Figure 1: Using the convex hull to find split points

Thus our approach is to compute the visible part of the convex hull for each of the given subsequences (see figure 1). For each sample point its distance to the line defined by the next preceding and succeeding sample points on the hull is computed. If there is a point whose distance exceeds a threshold value (i.e. the shape is not even “almost convex”), the sequence is divided at a point with a maximum distance, and this split procedure is applied again in a recursive manner.

Note that the visible part of the convex hull can be computed efficiently in linear time, since we have an angular ordering of the points around the range

finder. The algorithm is an adaptation of Graham's scan [PS88].

3 Object Correspondence

As we intend to track objects in a dynamic environment, we have to compare information about objects from successive scans. This is done by a combination of graph algorithms. Below we will briefly review some notions in graph theory before presenting the algorithm.

3.1 Notions in Graph Theory

A (*directed*) graph is an ordered pair (V, E) with vertices $V \neq \emptyset$ and edges $E \subseteq V \times V$. An edge $e = (u, v)$ is said to have *source node* u , *target node* v , and to be *incident* to u and v . The *subgraph* of $G = (V, E)$ induced by $V' \subseteq V$ is the graph $G' = (V', E \cap (V' \times V'))$. A graph $G = (V, E)$ is *bipartite* if there is a partitioning of the nodes $V = S \cup T$, $S \cap T = \emptyset$, such that $E \subseteq (S \times T) \cup (T \times S)$. A *matching* M in a graph $G = (V, E)$ is a subset of its edges $M \subseteq E$, such that every node $u \in V$ is incident to at most one edge $e \in M$ of the matching. A matching M is *perfect* if for each node $v \in V$ there is an edge $e \in M$ such that e is incident to v .

Let $pred(v) = \{u \in V \mid (u, v) \in E\}$ the set of *predecessors* of v and $succ(v) = \{w \in V \mid (v, w) \in E\}$ the set of *successors* of v . Let $e : V \rightarrow \mathbb{R}$ a node label, the *excess* of a node. Let $u : E \rightarrow \mathbb{R}^{\geq 0}$ and $c : E \rightarrow \mathbb{R}^{\geq 0}$ edge labels, the *capacity bound* and the *cost* of an edge. Now we can ask for an edge label $f : E \rightarrow \mathbb{R}^{\geq 0}$, a *flow*, which complies with the mass balance condition

$$e(v) = \sum_{w \in succ(v)} f((v, w)) - \sum_{u \in pred(v)} f((u, v))$$

for each node $v \in V$ and the capacity bound

$$f(e) \leq u(e)$$

for each edge $e \in E$. Such a label f is called a *feasible flow* in G . Often only a special case is considered, where there is exactly one node s with $e(s) > 0$, the *source node*, and exactly one node t with $e(t) < 0$, the *sink node*. A well known problem is the question for a *maximum flow* in this special case, that is, how large may the amount of excess $e(s) = -e(t)$ be such that there is still a feasible flow. If there is any feasible flow in a given labeled graph, we can search for a feasible flow f^* with minimal cost

$$\sum_{e \in E} f^*(e) \cdot c(e) = \min_{f \text{ is a feasible flow}} \left(\sum_{e \in E} f(e) \cdot c(e) \right)$$

among all feasible flows f .

There are several algorithms which solve these problems efficiently. Implementations are available for example via the LEDA library [MN99]. For more details on graph theory and network optimization see [AMO93, Law76, Nol76].

3.2 Matching Objects

From the previous and the current scan two sets of objects $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_m\}$ are given. The goal is to find for objects $u_i \in U$ from the previous scan corresponding objects $v_j \in V$ from the current scan. This can be seen as a matching in the bipartite graph $(U \cup V, U \times V)$.

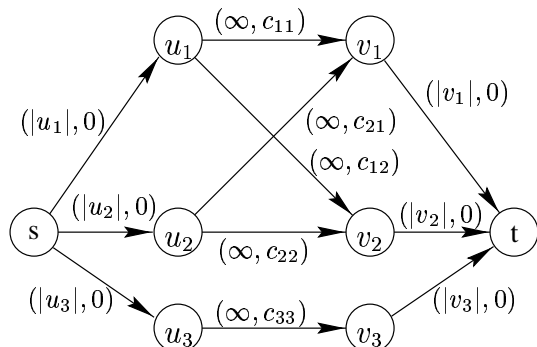


Figure 2: Graph G for minimum cost maximum flow

To find a matching representing plausible assignments between objects at successive points in time in the real world, we start by computing a maximum flow with minimal cost in a graph $G = (\{s, t\} \cup U \cup V, E)$ from source s to sink t as illustrated by figure 2. Each edge is labeled by its capacity and costs, where $|u|$ denotes the shape length of object u , and c_{ij} denotes the dissimilarity of objects u_i and v_j as described below. We have

$$E = \{(s, u) \mid u \in U\} \cup \{(v, t) \mid v \in V\} \cup \{(u, v) \mid P(u, v)\}$$

for some predicate $P : U \times V \rightarrow \{true, false\}$ of reasonably low computational complexity. We could use the constant predicate *true* here (i.e. accept all edges), but for practical reasons P is chosen such that $P(u, v)$ is true if the distance between the centers of gravity of u and v does not exceed a threshold value. This is equivalent to the assumption of a speed limit for objects in our environment. The finite scanning frequency does not allow tracking of arbitrary fast objects anyway.

Thus the size of the graph is reduced without imposing a further restriction, resulting in faster flow computations. Finally an object matching is deduced by retaining only edges conveying large amounts of this minimum cost maximum flow, i.e. we compute a maximum weight matching. Details on each step are given below.

Capacity Labels. For each object $u_i \in U$ we compute the length $|u_i|$ of the polygonal chain induced by its scan points. This length is taken as capacity label $u(e)$ for the edge $e = (s, u_i)$. Analogously we determine $u(e) = |v_j|$ for $e = (v_j, t) \in V \times \{t\}$. Finally we assign infinite capacity to edges $(u_i, v_j) \in U \times V$. Computing all capacity labels requires $\mathcal{O}(|S| + |U| \cdot |V|)$ computational steps, where $|S|$ denotes the size of a laser scan image. Intuitively we try to assign as much object shape length as possible from one scan to the next by computing a maximum flow. This is reasonable if we assume small changes of this length for each object between two successive scans.

Cost Labels. Edges (s, u_i) incident to the source node and edges (v_j, t) incident to the target node are assigned zero costs. So now consider edges (u_i, v_j) incident only to nodes representing objects. These edges will be assigned costs that reflect the similarities in shape and position of these objects in the real world, rendering less plausible object matchings more expensive than plausible ones. Our approach to compute these cost labels is to roughly estimate the work needed to transform one object into the other. Note that for a resting point of mass the work that is necessary to move it by a certain distance in a constant period of time is proportional to the square of this distance.

For each object u_i a constant number of evenly spread sample points $U_i = \{u_1^i, \dots, u_N^i\} \subseteq \mathbb{R}^2$ are taken from its shape as an approximation of it. Analogously for each object v_j points $V_j = \{v_1^j, \dots, v_N^j\}$ are taken from its shape. For each edge $(u_i, v_j) \in U \times V$ of graph G let $H_{ij} = (U_i \cup V_j, U_i \times V_j)$ a bipartite graph with an edge label $w_{ij} : U_i \times V_j \rightarrow \mathbb{R}^{\geq 0}$ which is defined by $w_{ij}((u_k^i, v_l^j)) = (d_2(u_k^i, v_l^j))^2$, where d_2 denotes the Euclidean distance in the plane. Since we do not want to make an assumption about the maintenance of the order of points on an object shape between successive scans, we follow a least effort approach and compute a minimum weight perfect matching M_{ij}^* in H_{ij} . The total weight $c_{ij} = \sum_{e \in M_{ij}^*} w_{ij}(e)$ of this matching is taken as a rough estimate for the necessary work and is assigned to the according edge $e = (u_i, v_j)$ of graph G as the value of its cost label $c(e)$.

These matchings are computed by a function from the LEDA library which requires $\mathcal{O}(n(m + n \log n))$ computational steps for a graph with n vertices and m edges. But these two parameters are constants in our context. Thus computing all cost labels requires at most $\mathcal{O}(|S| + |U| \cdot |V|)$ computational steps.

Flow Computation. Using these labels a maximum flow from source s to sink t with minimal costs is computed. This flow can be computed within at most $\mathcal{O}(|S| + (|U| + |V|)^3 \log((|U| + |V|) \cdot C))$ computational steps where C denotes the maximum edge costs, if the FIFO preflow push algorithm is used for the maximum flow and the cost scaling algorithm is used for minimum cost flow computation [AMO93].

Object Matching. The computed flow gives an idea of the motion in the environment of the robot but does not yet induce a unique matching. There may be objects that split and rejoin (consider a human and his arm) and thus induce a flow from one node to two successors or reversely. As the length of the shape of an object varies there may be a small flow reflecting these changes as well. Thus it is a good idea to focus attention on edges with a large amount of flow. Consequently the final step in our approach is to compute a matching of maximum weight in the bipartite subgraph of G induced by the two object sets U and V , using the flow labels computed in the previous step as weight labels. This last step is accomplished within at most $\mathcal{O}((|U| + |V|)^3)$ computational steps. We finally have unique assignments between objects from two successive scans by this matching.

4 Applications

Fast and robust tracking of moving objects is a versatile ability, which we use in two applications: detection of obstructions and motion coordination with a guiding person. In these examples the motion of the robot is controlled by a reactive system [FS98, PSS98] that is able to avoid collisions with static and dynamic obstacles.

4.1 Obstruction Detection

A goal beyond the scope of this paper is to enable a mobile system to recognize certain situations in its environment. As a first situation we address deliberate obstructions by humans, people who attempt to stop the mobile system. This situation has a certain importance to real world applications of mobile robots,

since systems like these may attract passers-by to experiment on how the system reacts and controls its motion.

Our approach is based on a supervised area in front of the robot and three counters assigned to each tracked object. The supervised area represents the space that the robot needs for its next motion steps. We call an object *blocking* if it is situated inside this area but does not move at a sufficient speed in the robot's desired direction. The counters represent the number of entrances of an object into the supervised area, the duration of the current blocking state and the overall duration of blocking states. If a threshold value is exceeded, the corresponding object is considered obstructing. Following a dynamic window approach, these counters forget blockings after a period of time. Note that a passively blocking object is evaded by the control scheme, i.e. its relative position leaves the supervised area. Therefore this counter approach in conjunction with the used control scheme detects deliberate obstructions fairly well.

4.2 Motion Coordination

Objects in the environment are not always opponents. In our second application one object is accepted as a guide that the robot has to accompany. This idea is inspired by our implementation of the tracking system on a robotic wheelchair. Typical applications are a walk in a park, shopping mall or pedestrian area together with a disabled person. There is no need for continuous steering maneuvers, it is sufficient to indicate the guiding person.

This ability to accompany a moving object is realized by a modification to the underlying velocity obstacle approach [FS98], for details see [PBKS01]. However it should be clear that this application demands for robust tracking of the guide which is provided by the presented approach.

5 Experiments

The described system is implemented on a robotic wheelchair equipped with a SICK laser range finder and a sonar ring [PSS98]. Computations are performed on an on-board PC (Intel Pentium II, 333 MHz, 64 MB RAM). The laser range finder is used to observe the environment, whereas the sonar ring helps to avoid collisions with obstacles that are invisible to the range finder.

The tracking system has been tested in our lab environment and in the railway station of Ulm. The number of objects extracted from a scan typically ranges

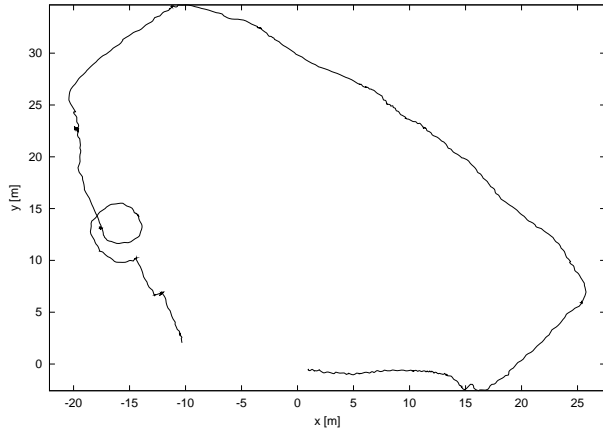


Figure 3: Outdoor trajectory

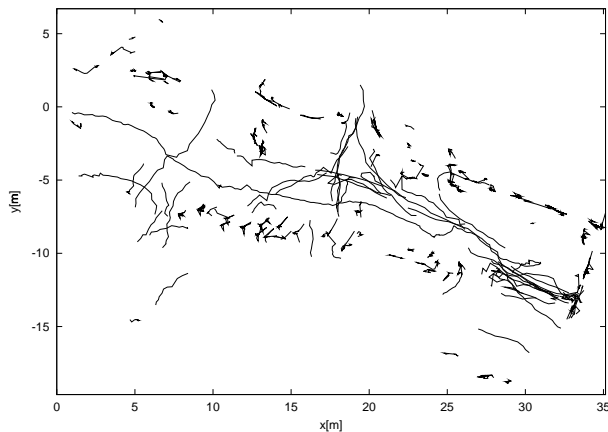


Figure 4: Trajectories of objects tracked in the railway station of Ulm

from ten to twenty, allowing cycle times of about 50 milliseconds using the hardware mentioned above. However the performance of the serial communication link to the range finder imposes a restriction to three cycles per second.

Figure 3 shows the trajectory of a guide walking outside on the parking place in front of our lab. The guide has been tracked and accompanied for 1073 cycles (more than five minutes), until he finally became occluded to the range finder. The wheelchair moved counterclockwise. The small loop is caused by the guide walking around the wheelchair.

Figure 4 shows trajectories of numerous objects tracked in the railway station of Ulm. The wheelchair moved from $(0, 0)$ to about $(30, -10)$ accompanying a guide. Pedestrians' trajectories crossing the path of the robot or moving parallel can be seen as well as static obstacles, apparently moving as their centers of

gravity slowly moved due to the change of perspective and dead reckoning errors. This scene has been observed for 247 cycles (82 seconds).

6 Discussion

Unfortunately the tracking system still loses tracked objects occasionally. One obvious cause is occlusion. It is evident that invisible objects cannot be tracked by any system. But consider an object occluded by another object passing between the range finder and the first object. Such an event cancelled the tracking shown in figure 3, where the guide was hidden for exactly one scan. Hence a future system should be enabled to cope at least with short occlusions.

But tracked objects get lost occasionally even if they are still visible. This might happen for example if new objects appear and old objects disappear simultaneously, as the visual field of the range finder is limited. To illustrate this, imagine a linear arrangement of three objects. Now delete the leftmost object and insert an object next to the rightmost. A flow computed as described above induces a false assignment, that is a shift to the right. This problem is partially dealt with by the restriction to a local search for correspondents as presented in section 3.2. It might be further improved if we do not assign any old objects to new objects that become visible by a change of perspective due to the robot's motion.

Another promising idea to improve robustness is to integrate data from different sensors than a planar laser range finder. A first approach to combine range data and video image data is to adapt cost labels in a way that "color distances" are respected as well. Thus geometric unclear assignments might be disambiguated correctly. This approach requires a change from a planar model of the scene to a prismatic one, where columns of a video image are assigned to each scan point. A future implementation may be enhanced to use full 3D range data if the object extraction step is adapted appropriately and shape lengths are replaced by surface areas as edge capacities. Color information might be incorporated in the same manner as in the 2D case. However note that there is no easy way to generalize the extraction of "almost convex" objects from 2D range data to 3D range data.

In some cases object extraction fails to properly split composed objects. If these objects are recognized separately in the previous scan, either of them is lost. But this situation may be recognized by looking at the minimum cost flow in the graph, if there is a significant flow into one node from two predecessors. This might give a hint to split the newly extracted object.

As object extraction is error-prone, one might follow the idea to compute the flow based on the scan points before extracting objects by searching for proximity groups of parallel edges carrying flow. However this might be computational infeasible, since the sizes of the graphs involved in the computations of the flows are heavily increased. Recall that the time complexity of the overall algorithm is more than cubic with respect to the number of nodes in the graph, so we might run into problems even now if scenes become very complex.

Information about the motion of an object drawn from previous scan images could be used to compute an approximation of its current position and thus direct the search for correspondents. But a first implementation of this concept regarding the motion of centers of gravity shows bad behaviour in some environments, considering walls moving as their visible parts grow.

7 Conclusion

In this paper we presented an object tracking system based on a laser range finder as its sensor and on graph algorithms for data processing. The basic idea is to represent the motion of object shapes in successive scan images as flows in bipartite graphs. By optimization (maximum flow, minimum cost, maximum weighted matching) we get plausible assignments of objects from successive scans. In our experiments the system proved to perform considerably more robust than its predecessor [PSSS98] which is based on a greedy nearest neighbor search among the objects' centers of gravity. However the presented system still has to be improved for real long-term tracking as shown in the discussion.

Acknowledgements

This work was supported by the German Department for Education and Research (BMB+F) under grant no. 01 IL 902 F6 as part of the project MORPHA.

References

- [AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.
- [FS98] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, July 1998.
- [Law76] Eugene L. Lawler. *Combinatorial optimization: networks and matroids*. Rinehart and Winston, New York, 1976.
- [MA98] Esther B. Meier and Frank Ade. Object detection and tracking in range image sequences by separation of image features. In *IEEE International Conference on Intelligent Vehicles*, pages 280–284, 1998.
- [MN99] Kurt Mehlhorn and Stefan Näher. *LEDA — A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.
- [Moe99] Thomas B. Moeslund. Computer vision-based human motion capture – a survey. Technical Report LIA 99-02, University of Aalborg, March 1999.
- [Nol76] Hartmut Noltemeier. *Graphentheorie: mit Algorithmen und Anwendungen*. de Gruyter, 1976.
- [PBKS01] E. Prassler, D. Bank, B. Kluge, and M. Strobel. Coordinating the motion of a human and a mobile robot in a populated, public environment. Submitted to Int. Conf. on Field and Service Robotics (FSR), 2001.
- [PS88] Franco P. Preparata and Michael I. Shamos. *Computational geometry : an introduction*. Springer Verlag, 1988.
- [PSS98] E. Prassler, J. Scholz, and 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*, Aachen, Germany, 1998.
- [PSSS98] E. Prassler, J. Scholz, M. Schuster, and D. Schwammkrug. Tracking a large number of moving objects in a crowded environment. In *IEEE Workshop on Perception for Mobile Agents*, Santa Barbara, June 1998.
- [SB98] Karin Sobottka and Horst Bunke. Vision-based driver assistance using range imagery. In *IEEE International Conference on Intelligent Vehicles*, pages 280–284, 1998.