V. Kůrková et al. (Eds.): ITAT 2014 with selected papers from Znalosti 2014, CEUR Workshop Proceedings Vol. 1214, pp. 28–33
http://ceur-ws.org/Vol-1214, Series ISSN 1613-0073, © 2014 P. Fejfar, D. Obdržálek

Smart and Easy Object Tracking

Petr Fejfar, David Obdržálek
Department of Theoretical Computer Science and Mathematical Logic
Faculty of Mathematics and Physics, Charles University
Malostranské náměstí 25, 118 00 Praha 1, Czech Republic

Abstract. *In this work, we present a system which is able to track objects over time even in situations the objects are not apart enough to get separated input data. The system processes distance data repeatedly acquired from a distance sensor. More specifically, it tracks balls rolling on the floor using laser rangefinder measurements. The noisy data is first filtered, then processed for ball detection and consecutively paired with long-term data of objects movement. Finally, Kalman filter helps to bridge the drop-outs of object position information caused by interactions between the balls and occlusions.*

Based on the tracked movements and smart predictions, the system is able to cope well with two principally different and poorly distinguishable situations: when two balls pass close to each other without touching and when two balls collide and bounce away.

The system has been successfully implemented on a real robot equipped with a short-distance IR laser rangefinder.

1 Introduction

In robotics, only few systems can work without sensing and observing the real-world environment in which they perform their actions. Moreover, simple sensing is usually not enough. Sensed low-level data needs to be processed further to obtain high-level object data so that the control system has enough information about its working environment to be able to perform its high-level task.

Among the methods for such data manipulation, object recognition, matching and tracking is often used. Low-level sensor data is processed so that higher-level object information or description is retrieved. Then, the recognized objects are matched against the recognized set of objects and consecutively such objects are tracked over time. However, in not so rare situations, the objects are very similar which makes the matching and tracking challenging. Moreover, when there are more similar objects in the scene close to each other, they can get easily mismatched and exchanged during the matching process which leads to wrong high-level data being processed at high control levels. The situation could get even worse when the objects move and get close to each other or when they even interact. Therefore, good detection and matching is highly important.

2 Moving Object Tracking

There are many systems which use video stream or still pictures as basic input data. Object recognition, matching and tracking is done using many various methods.

For example, Cheriyadat, Bhaduri and Radke use in [1] the following approach: the objects are searched based on their movement between the time snapshots when two pictures were taken. Input images are interpreted as 2D matrices where each pixel is represented by a real number. The two directly following pictures (the matrices) are subtracted and a resulting matrix represents the change of position of the objects between the two image acquisition times. The highest numbers can be found at the edges of moving objects. The process continues by finding corners using the Shi-Tomasi-Kanade corner detector [2] and Rosten-Drummond corner detector [3]. Then, the path along which the corners travel is found using the Kanade-Lucas-Tomasi (KLT) algorithm [4]. The authors define the metrics of the “movement similarity” of the corners. Then they look for corners which appear to move similarly according to this metrics and gather them in sets. Such corner sets are then announced as the result of moving objects detection.

There are many different object tracking algorithms which take video images as input data. Most of them not surprisingly work under specific conditions and suffer from different weaknesses. Some authors therefore suggest using more different algorithms and then compiling their results. To illustrate, Siebel and Maybank propose in [5] to track people using combination of tracking moving areas, searching human body silhouettes and searching head silhouettes. The hypotheses of moving people which result from those different algorithms are then compiled and filtered which results in a robust algorithm. They can correctly detect two people holding conversation: while they could be detected as one bigger moving region, the body silhouette-searching algorithm correctly detects two distinct bodies.

Another approach to people tracking was presented by Cui, Song, Zhao, Zha, and Shibasaki in [6]. The authors use rangefinder sensors placed close to the ground at ankle level. They aggregate measurements from multiple sensors placed around the area and check the ends of individual rangefinder emitted rays. Such ends denote an obstacle. The points at those ends are filtered using the Parzen windows algorithm [7] and the local maxima are considered to be people legs. The authors focus on the leg which is still during the walk and pair the legs belonging to one person. Finally, based on the leg movements they output the person position.

An interesting combination was also presented in [8]: the authors extend the rangefinder-based system by using an additional video camera. By positioning the camera at other place than the rangefinders, they can also address situations when the rangefinder is occluded and does not provide relevant source data. In such situations, the camera may see the objects because of different observation angle.

3 Motivation

For our work, we have selected one of the sub-tasks needed for implementation of a robot for the SICK Robot Day 2012 Competition [9]. During the contest, three robots compete in a circular arena, collecting coloured balls. Each robot is assigned one colour – green, yellow or white. Its task is to collect balls of that colour and bring them to a target area of the same colour. The arena is about 15m in diameter and 10 balls of each colour are randomly spread there (see Fig. 1). The three robots have to operate at the same time. At the start of the match, the balls are still. However, due to their number and as three robots independently run there, the balls easily become moving and bouncing. It is therefore impossible to operate statically using an image of the scene taken at the match start.

For the operation, the robot may use nearly any type of sensor. A combination of a colour camera and a laser rangefinder was selected for ball detection and tracking[1]. The camera used is a standard “webcam”-type camera. The rangefinder (SICK LMS 100) provides distance measurements within 0.5-20m in the angle of 270° with configurable step of 0.5° or 0.25°. The scanner is mounted so that we acquire data at the level of approximately the ball semidiameter (example of data is shown at Fig. 2). From the camera, the control system finds out the ball colour. However, due to the significantly varying illumination both at different places and directions on the scene (standard gymnasium with full-height windows over the long side) as well as during the day (as the sun proceeds), it is quite difficult to obtain proper colours of objects visible by the camera. Therefore, the idea was to get the colour whenever possible and connect it with object tracking which is due to the nature of a laser rangefinder not colour-aware. For object tracking as such, the colour is not important. Therefore it is not considered in the following text.

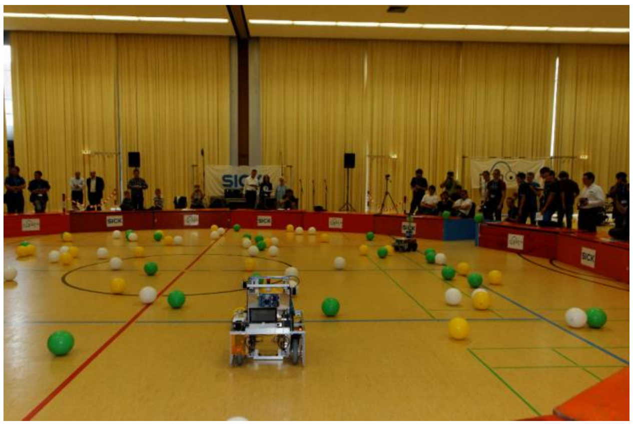
Fig. 1. SICK Robot Days 2012 circular playing arena. Green team starting place (marked by two green balls) can be seen at far right; white and yellow starting places are not visible from this position as all the starting places are located 120° apart around the arena.

[1] Other sensors were used on the robot too, but they are irrelevant to the topic of this paper.

4 Solution

For the ball tracking, the following algorithm was proposed and implemented:

1. Data is read from the laser rangefinder
2. Data noise is filtered
3. Positions of individual balls are calculated
4. Calculated positions are matched with tracked balls
5. Balls are tracked over long-time

This algorithm is repeatedly run throughout the whole robot run.

In the following sections, individual steps will be described more in detail.

4.1 Data Acquisition

As mentioned above, the data was acquired using the SICK LMS 100 laser rangefinder. Example of a scan is shown at Fig. 2. The principle of measurement is in measuring the time of flight of a laser beam. Full set of data was available 20 times per second which was more than enough for our task. However, the data was subject to noise and disturbances caused by the nature of the measurement method. For example, the laser can easily get reflected on a shiny surface or can interact with dust in the air. Also, the distance data may vary with the different properties of the object from which the laser mirrors. Last, but not least, due to the speed of the signal, time measuring is crucial and any inaccuracy in time measuring directly influences the calculated distance. This can be easily seen as non-linear room edge at Fig. 2.

4.2 Noise Filtering

To filter the noise, a simple median filter is used. Originally, floating average was used, but the median filter better preserves the edges.

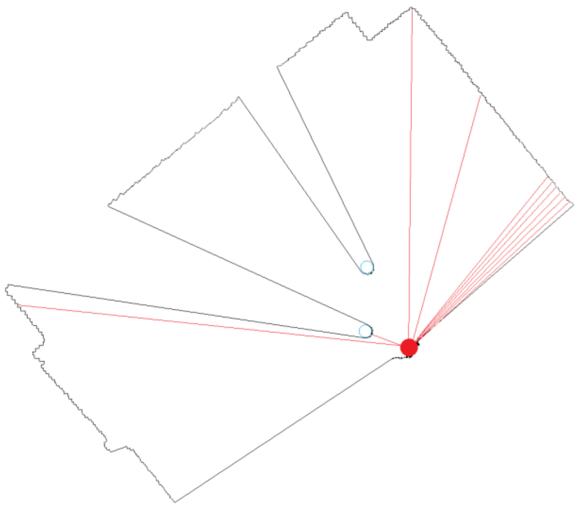
Fig. 2. Example of SICK LMS 100 data. Red dot denotes the sensor position, red lines show individual range scans (only a selected subset shown). Two blue circles are the balls with their scan “shadows” clearly visible. Black boundaries mark the distance measured at the specific angle.

4.3 Ball Position Calculation

In the scan, the ball makes “a shadow.” The distance measured directly to the side of the ball and to the ball itself differs a lot. Interpreting the scan as a one-dimensional picture, the boundary between the ball and the background is represented by an edge. Therefore, the balls are detected using the edge detection algorithm. Every shadow which is detected on the scan is then checked for the expected size – from the distance data, we know the distance to this object and knowing the ball diameter we can estimate its width in the data. All objects which do not correspond to this simple check are discarded. They can result from the noise (if the object is too narrow) or they can represent an opponent robot (if it is too wide as the robots are much bigger than the balls).

From the edges (shown as blue lines at Fig.3), the expected ball centre is calculated (shown as a cross). Usually the shortest ray in the middle between the boundaries prolonged by a ball semidiameter is a good guess (see also Fig. 4).

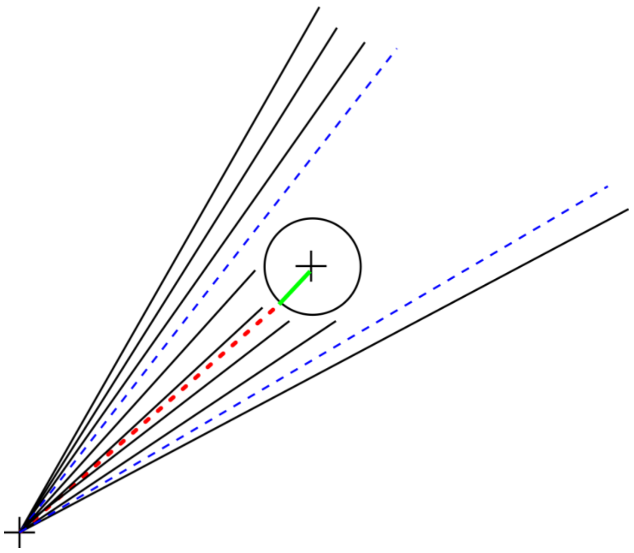
Fig. 4. Ball centre detection. Black lines are the individual scans. Blue dashed lines are the ball boundaries. Red dashed line is in the middle between them with the length of the shortest ray between the two boundaries. Green line is its prolongation by the ball semidiameter and leads to the centre of the ball (black cross).

4.4 Ball Matching

For this part of the algorithm, the inputs are a set of actually detected balls and a set of long-time maintained set of tracked balls. The output will be three sets: balls paired with measurements, solitary measurements and solitary balls.

This part of the task is an optimization task. Formally, we try to minimize the weighted pairing on a full and symmetric bipartite graph. For this, the Hungarian algorithm[2] [10] is used. We minimize the sum of distances between the ball tracked in long time and the paired ball detected in the current measurement. As the Hungarian algorithm expects the same number of objects in the two sets to be paired, we add virtual balls in case of uneven number in the two sets (see Fig. 5). These will be eliminated after the optimization and it can be proven they do not affect the optimality of the solution found.

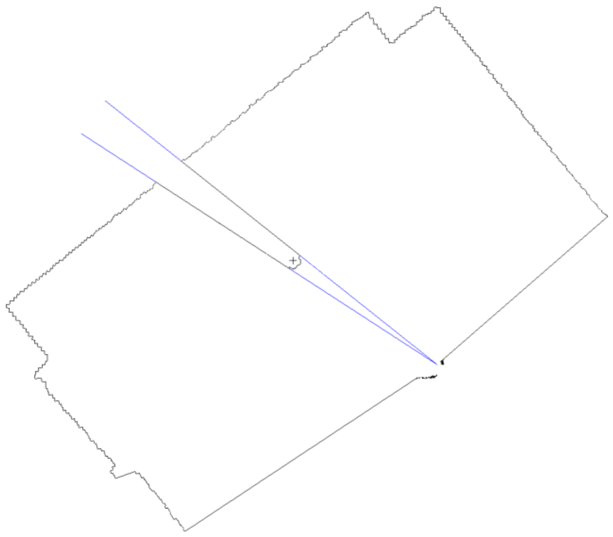
Fig. 3. Ball position calculation. Blue lines mark the boundaries (edges) of a ball, the cross denotes the resulting ball centre.

4.5 Ball Tracking

The balls are tracked over time using a basic version of the Kalman filter (described e.g. in [11]) – we want to acquire a velocity vector from a series of position measurements in time which are subject to noise and dropouts (see later). Despite the strong preconditions for Kalman filter (especially the requirement that the noise has

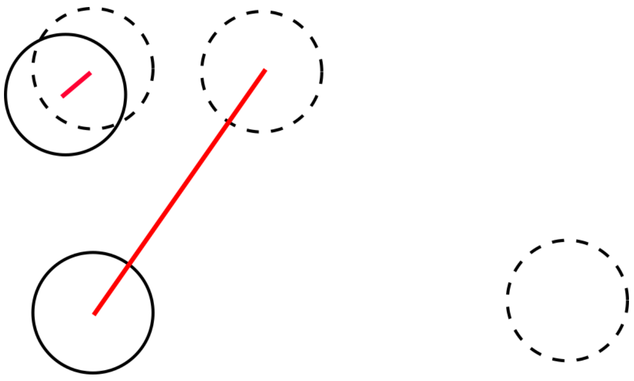
Fig. 5. Hungarian algorithm pairing.Full line circles are the predicated (tracked) balls, dashed circles are the balls detected in the new scan, red line marks the pairing (the picture shows only a subset; the detected ball at bottom right does not pair with either of the two tracked balls).

[2] Also known as Kuhn-Munkres or Munkres assignment algorithm

to be of normal distribution), the filter is so robust that it is still very well useful even when these conditions are not fully met in real life.

For our problem, the Kalman filter can overcome problems with short-time ball occlusions caused by balls travelling on trajectories which seemingly cross from the point of the observer view. In such case, one of the balls (the one further away from the observer) will not be visible for some time as it is occluded by the closer one. In the input data processing, this means that only one ball will be detected and the second one will be missing. However, thanks to the nature of the pairing algorithm and Kalman filter “prediction”, this is usually only a temporary dropout during which the occluded ball is in fact represented by that added virtual ball. Real tests have shown correct matching is maintained or re-established after the occluded ball leaves the other ball shadow.

The Kalman filter can filter the noise if the process model is linear in respect to the position and control and if the noise is of a normal distribution. The linearity usually does not impose problems for simple systems; however, the noise distribution is in real life often anything else but normal.

5 Implementation

As mentioned earlier, this work was motivated by the SICK Robot Day 2012 Competition. However, it was implemented after the event based on the observations and experience gained. It was also tested on the real input data logs gathered during the competition. For data acquisition, the SICK LMS 100 laser rangefinder was used (mounted on the robot during the competition), but other input source which provides a distance measuring could be used too.

The implementation of the main algorithm was done so that the solution could be used not only for this particular robot but also in other cases where object tracking is required. The core of the algorithm is written in C++ with the use of Boost library [12] (TCP/IP communication, thread management, input/output, data structures for mathematical calculations).

Supportive visualisation and debugging tools were written with the use of OpenGL [13] and Qt [14] libraries, however that part is not necessary for the main algorithm use.

The selection of libraries and tools makes it easy to port the algorithm to other platforms, both general and specialized embedded, as well as use the visualization tools at different platform than the core algorithm. Originally, the system was developed and tested on Microsoft Windows running on a mid-range laptop machine but it can be with no change compiled and used on Linux systems. Porting for ROS as a currently very popular system in robotics is straightforward (the algorithm can be nearly directly wrapped in a ROS node). It is certainly possible to use it in other systems too (e.g. FreeRTOS was considered but finally not ported as it was not needed).

6 Results

The implemented system was proposed to help solve ball tracking in a specific task in real life. Therefore, we judge its success by evaluating different situations which may in the model conditions occur. In the following text, these situations are listed and discussed with showing typical figures of the respective cases.

Fig. 6-9 show the respective cases. A ball is represented by its centre as a cross (calculated from the laser scan, see paragraph 4.3). Around the ball, the shadow cone typical for rangefinders is shown (black cone; adjacent laser scan points are connected by a line for better visibility). Grey line shows the previous trajectory of the ball (as recorded based on the ball matching, see paragraphs 4.4 and 4.5). Violet line is a current velocity vector (as tracked by the filter).

Fig. 6-8 show the evolution of the respective situation in time (each of the four snapshots depicts one typical phase). Fig. 9 shows ball clusters as a typical case when our ball detection fails.

6.1 Single Ball Tracking

This is the very basic situation. There is only a single ball in the scan area. The situation is shown at Fig. 6. The ball was travelling right to left on a straight path. Starting with speed 0, the Kalman filter first gets adapted to the real ball speed and noise. In the central part of the ball move, the filter well corresponds to the real ball position. As the ball leaves the scan sector, the ball detection algorithm starts to have problems, which are well compensated by the Kalman filter.

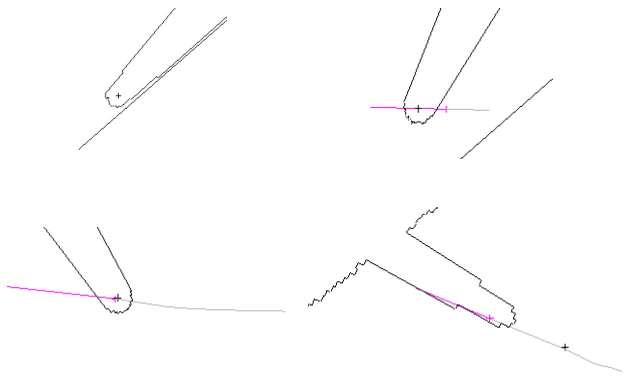

Fig. 6. Single ball tracking. Top left: First detection of the ball. Top right: Kalman filter adaptation to the noise, speed and position stabilization. Bottom left: Stabilized movement. Bottom right: Problems to detect the ball which is too far from the sensor.

6.2 Tracking Multiple Objects with Occlusion, without Collision

When two balls move towards each other, two situations may happen: either the balls collide and bounce, resulting in the balls travelling in directions opposite to their original ones. In this paragraph, the first situation is addressed.

When the two balls do not collide, the ball closer the scanner is tracked as if it moves alone (see Fig. 7 showing the evolution of this situation in time). The ball at bigger distance from the scanner is occluded, but continues to move with the same direction and speed as before the occlusion started. Here, the Kalman filter bridges the time when the ball is occluded and precisely tracks the ball meanwhile. That is well seen at the time the second ball leaves the first ball shadow.

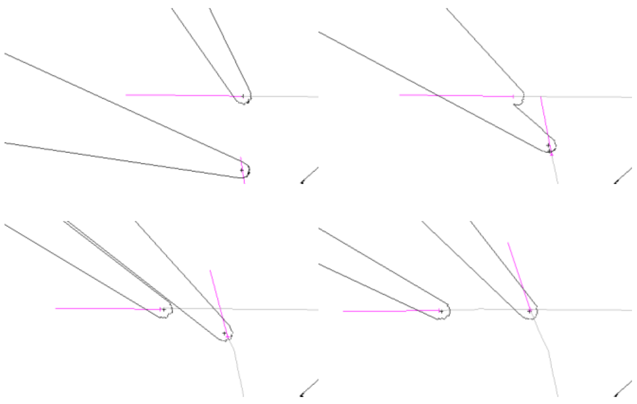
Fig. 7. Multiple balls, no collision. Ball 1 travels right to left, ball 2 bottom to top (top left), ball 2 partially occludes ball 1(top right), ball 1 leaves the shadow (bottom left), balls continue to move independently (bottom right)

6.3 Tracking Multiple Objects with Collision

When two balls collide, they form a single object on the scanned data and they cannot be separated as two balls. However, the Kalman filter continues to predict the movement individually. Fig. 8 shows this situation in time. When the two balls separate (Fig. 8 bottom left), their velocity vectors are deformed and do not correspond to data maintained by Kalman filter. During a few consecutive scans (pictures), it adapts and continues to track.

In our implementation, the balls move relatively slowly in comparison to the frequency of the laser scans (20 per second) and to data processing which makes the filtering successful. Preliminary tests have shown that even under real conditions with full number of balls in game we can afford dropping and skipping lot of scans[3] while still maintaining the correct tracking.

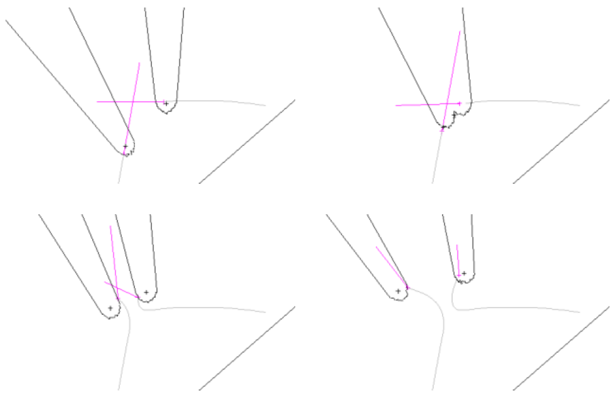
Fig. 8. Multiple balls, collision. Ball 1 travels right to left, ball 2 bottom to top (top left), the balls collide, forming one object on the scan (top right), balls separate (bottom left), balls continue to move independently (bottom right)

6.4 Object Clusters

In this paragraph we show an example of a situation when our algorithm fails.

When more balls are close to each other forming a cluster, we cannot easily distinguish whether it is a ball cluster or an obstacle of totally different nature, for example an opponent robot. On Fig. 9, the lower cluster is most likely formed by two balls. The upper cluster might be a single ball on the right plus two close balls on the left. However, it can be another obstacle as well (e.g. an opponent robot, room equipment, a person etc.).

Our algorithm was design to search and track individual balls and cannot cope with such situation. It may be possible to analyse the object shape – e.g. roundness of the obstacle – instead of just checking the edges like described in paragraph 4.3.

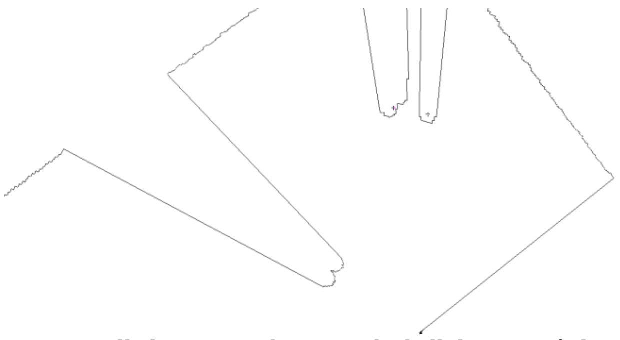
Fig. 9. Ball clusters On this scan, the ball detection fails: two balls close to each other (bottom / left), several balls in one area or completely another obstacle (top / right)

[3] This is by intention, not as a result of e.g. slowness of the algorithm

7 Conclusion and Future Work

The presented algorithm detects and tracks moving unisized balls on a laser rangefinder scans. It tracks their movement in longer time and well copes with the occlusions and collisions.

However, there are also situations this algorithm in principle cannot handle well (as shown for example in paragraph 5.4). Based on the real behaviour in our model situations, several ideas arose. These two are the major ones:

One of the improvements considered is a different algorithm for ball centre detection in the raw scan data (for example, Fig. 4 shows slight asymmetry which is not taken into account). Consecutively, as mentioned in paragraph 4.3, knowing the expected ball size, we calculate the expected cone width at the measured distance from the observer and compare that with the measured width, discarding objects where these data does not correspond. But we could also test the shape of the detected obstacle. That might detect multiple balls located close to each other; however, that might also compromise simplicity and speed of the edge detection.

Secondly, the pairing might be improved. Currently, only the position information is used. Adding speed and direction to the pairing algorithm might help better distinguish between two balls passing each other and bouncing and thus exchanging their positions after they separate.

Acknowledgment

This work was partially supported by the Czech Science Foundation under the project No. P103/10/1287.

References

[1] A.M. Cheriyadat, B.L. Bhaduri, R.J. Radke, “Detecting Multiple Moving Objects in Crowded Environments with Coherent Motion Regions,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008, p. 1-8.

[2] J. Shi, C. Tomasi, “Good Features to Track,“ *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, 1994, pp. 593-600.

[3] E. Rosten, T. Drummond, R. Boyle, “Machine Learning For High-Speed Corner Detection.” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, 1994, pp. 430-443.

[4] B.D. Lucas, T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision,” *Proc. of 7th International Joint Conference on Artificial Intelligence*, 1981, pp. 674-679.

[5] N.T. Siebel, S.J. Maybank, “Fusion of Multiple Tracking Algorithms for Robust People Tracking,” *Proceedings of the 7th European Conference on Computer Vision ECCV’02 – Part IV*, 2002, pp. 373-387.

[6] J. Cui, X. Song, H. Zhao, H. Zha. R. Shibasaki, “Real-Time Detection and Tracking of Multiple People in Laser Scan Frames,” *Augmented Vision Perception in Infrared*, 2009, p. 405-439.

[7] E. Parzen, “On Estimation of a Probability Density Function and Mode,” *Ann. Math. Statistics 33*, 1962, pp. 1065-1076.

[8] J. Cui, H. Zha, H. Zhao, R. Shibasaki, “Multi-modal Tracking of People Using Laser Scanners and Video Camera,” *Image and Vision Computing 26 vol.2*, 2008, pp. 240-252.

[9] “SICK Robot Day 2012 Competition”, *http://www.sick. com/robotday2012*

[10] H.W. Kuhn, “The Hungarian Method for the Assignment Problem,” *Naval Research Logistic Quarterly*, 1955 vol.2, pp.1-2.

[11] G. Welch, G. Bishop, “An Introduction to the Kalman Filter,” *University of North Carolina at Chapel Hill, Department of Computer Science*, 1995.

[12] “Boost C++ Libraries”, *http://www.boost.org*

[13] “OpenGL – The Industry’s Foundation for High Performance Graphics”, *http://www.opengl.org*

[14] “Qt Project”, *http://qt-project.org*