Vision-based SLAM and moving objects tracking for the perceptual support of a smart walker platform

Paschalis Panteleris* and Antonis A. Argyros**

Institute of Computer Science (ICS)
Foundation for Research and Technology - Hellas (FORTH)***

Abstract. The problems of vision-based detection and tracking of independently moving objects, localization and map construction are highly interrelated, in the sense that the solution of any of them provides valuable information to the solution of the others. In this paper, rather than trying to solve each of them in isolation, we propose a method that treats all of them simultaneously. More specifically, given visual input acquired by a moving RGBD camera, the method detects independently moving objects and tracks them in time. Additionally, the method estimates the camera (ego)motion and the motion of the tracked objects in a coordinate system that is attached to the static environment, a map of which is progressively built from scratch. The loose assumptions that the method adopts with respect to the problem parameters make it a valuable component for any robotic platform that moves in a dynamic environment and requires simultaneous tracking of moving objects, egomotion estimation and map construction. The usability of the method is further enhanced by its robustness and its low computational requirements that permit real time execution even on low-end CPUs.

Keywords: Visual Tracking; Human Detection and Tracking; Mapping; Egomotion Estimation; SLAMMOT; Smart Walker

1 Introduction

Tracking an unknown number of unknown objects from a camera moving in an unknown environment, is a challenging problem whose solution has important implications in robotics. In this work, we investigate how moving object tracking, ego-motion estimation and map construction can be performed simultaneously using 3D information provided by an RGBD sensor.

Our work is motivated by the perceptual requirements of the DALi c-Walker platform. The c-Walker is a device that aims to safely guide people with cognitive impairments through public spaces like airports and shopping centers. Using its on-board sensors, this "cognitive navigation prosthesis" monitors the environment in real time to

* Paschalis Panteleris is with the Insitute of Computer Science (ICS), FORTH, N. Plastira 100, Vassilika vouton, GR70013, Heraklion, Crete, Greece. padeler@ics.forth.gr

** Antonis Argyros is with ICS/FORTH and also with the Department of Computer Science, University of Crete, Greece. argyros@ics.forth.gr

*** This work was partially supported by the European Commission, through the FP7 project DALi (FP7-288917)

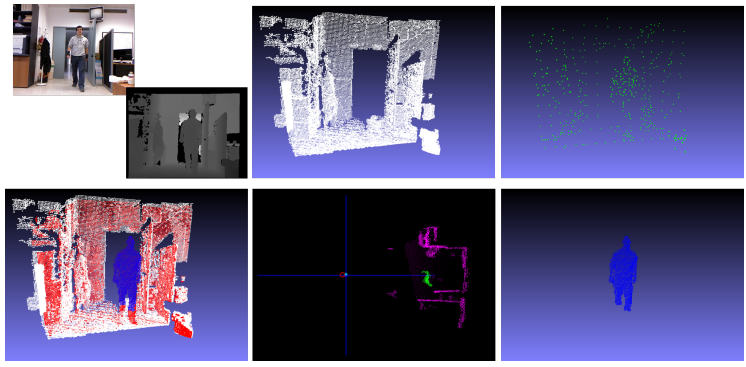

Fig. 1. Method overview. Top left: an RGBD frame. Top middle: The current map of the environment. Top right: A sparse set of 3D points used for egomotion estimation. Bottom left: registration of the whole point cloud with the current map of the environment. Bottom middle: a top view showing the camera position, the local environment map and the independently moving objects. Bottom right: The point cloud of the foreground moving object.

figure out a path that poses little risk for the user. It can also perform active replanning when, for example, passages are blocked because of groups of standing people.

To create such a smart walker that can function reliably, a number of challenges associated with its perceptual capabilities must be addressed effectively and efficiently:

- Independently moving objects need to be detected and tracked so that a safe path that avoids collision can be planned.
- Egomotion needs to be estimated so that the platform can be localized in its environment.
- The map of the environment which is required for platform localization, cannot be assumed to be known a priori but must be constructed on the fly. This is because the structure of the environment may be subject to frequent changes that render an offline map construction process very impractical.
- All the above needs to be available in unprepared environments, even in the absence of other technologies. Thus, the above perceptual tasks need to be supported strictly by sensors integrated onto the platform.
- To reduce costs and energy consumption, the computational requirements of the sensory processing modules should fit low-end computers that are on-board the platform.

In this paper, we present a solution that addresses simultaneously, robustly and efficiently all the above issues. The proposed method is designed to track moving objects such as humans, estimate camera egomotion and perform map construction based on visual input provided by a single RGBD camera that is rigidly attached to the moving platform. From a computational point of view, the method performs in real time even with limited computational resources.

The proposed method (see Fig. 1) segments and tracks objects that move independently in the field of view of a moving RGBD camera. The camera is assumed to move

with 6 degrees of freedom (DOFs), while moving objects in the environment are assumed to move on a planar floor. This last assumption is the only a priori knowledge made regarding the environment, and is exploited in order to reduce the computational requirements of the method since the primary goal of this work is to support perceptually an indoors smart walker platform. Motion is estimated with respect to a coordinate system attached to the static environment. In order to segment the static background from the moving foreground, we first select a small number of points of interest whose 3D positions are estimated directly from the sensory information. The camera motion is computed by fitting those points to a progressively built model of the environment. A 3D point may not match the current version of the map either because it is a noise-contaminated observation, or because it belongs to a moving object, or because it belongs to a structure attached to the static environment that is observed for the first time. A classification mechanism is used to perform this disambiguation. Based on its output, noise is filtered, points on independently moving objects are grouped to form moving object hypotheses and static points are integrated to the evolving map of the environment.

Experimental results demonstrate that the proposed method is able to track correctly moving objects. Interestingly, the performance of egomotion estimation and map construction aspects practically remains unaffected by the presence of independently moving objects, demonstrating the robustness of the ego-motion estimation module to noise as well as the capabilities of the foreground segmentation pipeline. From a computational point of view, the method performs at a frame rate of 50 fps on a laptop with an "Intel i7" CPU without the use of GPU acceleration, and can perform at near real-time speeds on ARM based embedded hardware.

2 Related Work

Our work is related to three fundamental and heavily researched problems in computer vision and robotics, that of multi-target tracking, camera egomotion estimation / localization and 3D scene structure estimation / map construction. A complete review of the related work constitutes a huge task even for any of the individual subproblems and is beyond the scope of this paper.

For the problem of object tracking in RGB images Yilmaz et al. [30] provides a comprehensive review. A number of methods try to address the problem of the changing appearance of the tracked objects through learning on-line the appearance model of a specific target and using it to track that target [4,24]. Several algorithms are specific to humans, as the detection and tracking of moving people is very important in several applications. Such algorithms [28,31,9,14,13] have improved a lot and provide reliable results when applied to simple scenes, especially from static cameras. The methods by Ferrari et al. [14] and Felzenszwalb et al. [13] are able to detect humans in non-pedestrian (i.e. sitting) poses with reasonable accuracy. However, camera motion in real-world, crowded environments which include large amounts of occlusion and clutter, as well as wide pose variation still poses a lot of challenges. Additionally, most of these methods are mainly concerned with the detection of people and not their temporal tracking. To address the challenges of tracking from a moving platform, several ap-

proaches [11,12,6,7] have recently been proposed that combine multiple detectors and work with various sensor configurations. These works perform data association to track people and are capable of estimating the camera motion. In our work we are limited to a single RGB-D sensor fixed on our target platform and thus we need to conform with the limitations of the hardware.

For the problem of simultaneous localization and mapping (SLAM), a recent review is provided in [15]. Most of the works depend on the assumption of a static environment. Deviations from this can be tolerated at different degrees, depending on the internals of the methods used. The consideration of SLAM in dynamic environments formulates the so called *SLAMMOT* problem. As stated in [21], SLAMMOT involves SLAM together with the detection and tracking of dynamic objects. SLAMMOT was introduced by Wang [5]. Their approach combines SLAM and moving object tracking that are performed based on a 2D laser scanner. Although conceptually very interesting, the practical exploitation of the method is limited by the 2D nature of laser scans and by the relatively high costs of the employed sensor. On top of the usage of a 2D laser scanner, Gate et al. [16] introduces a camera that aid the classification of moving objects. A purely vision-based method was proposed by Agrawal et al. [1] who employ a calibrated stereo configuration of conventional RGB cameras. Dense stereo 3D reconstruction builds local 3D models of the environment whose rigidity is tested in consecutive frames. Deviations are attributed to moving objects. The corresponding pixels are filtered for noise, grouped, and tracked based on Kalman filtering. The problem of deciding whether an object is static or moving has also been addressed by Sola [27] who employed contextual rules for that purpose. Wangsiripitak et al. [29] track a single, known 3D object. This information is used to safeguard an independent SLAM method from observations that are not compatible to the rigid world assumption.

The recent introduction of RGBD sensors has provided a cheap way of acquiring relatively accurate 3D structure information from a compact sensor. This has enabled the development of SLAM [18,17,10] and object tracking [20] methods with impressive results. The goal of our work is to address the combined SLAMMOT problem by investigating the coupling of the individual subproblems under the limited computational resources of the target smart walker platform.

3 Method

The approach we propose is outlined graphically in Fig. 1 while Fig. 2 shows a detailed flow diagram. The top left images in Fig. 1 show an RGBD frame from an indoors sequence. Using the RGBD input we generate a point cloud P_c and we select a number of sparse 3D points P_g (top right) that is used for localization. Although P_g could be identical to P_c, it is shown that a limited number of carefully selected points suffice to retain the information about the structure/map of the environment, M. This allows camera localization that does not rely on expensive feature descriptors and matching. The output of the registration process is the motion of the camera (egomotion) measured in the coordinate system of M. P'_c is the registered point cloud on M (bottom left in Fig. 1). All points that do not correspond to a point on the model are considered outliers. By clustering these outliers and monitoring their temporal behavior, these are further

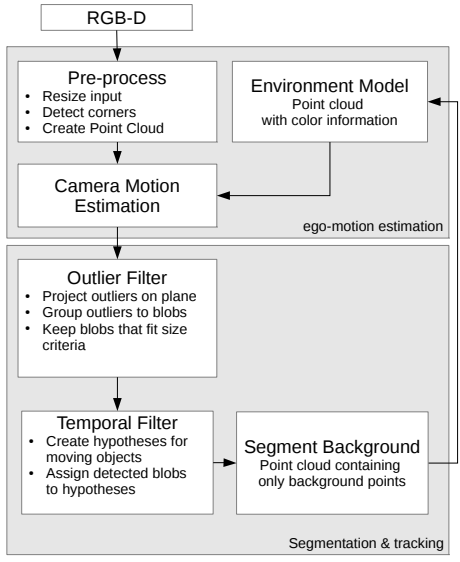

Fig. 2. Flow diagram of the method, see text for details.

characterized as noise, parts of independently moving objects or features that belong to the static environment. Hypotheses H about the moving objects are generated and tracked in subsequent frames. Points that belong to H are removed from P'_c and the resulting point cloud P'_{bg} is used to update M for the next frame. The rest of this section describes the steps of the employed approach in more detail.

3.1 Preliminaries

We consider point clouds P that consist of sets of colored 3D points $p = (D(p), C(p))$. $D(p)$ denotes the 3D coordinates of p in some coordinate system and $C(p)$ its associated color. Since we want to be able to compare points based on their color values, we represent color in the YUV space and perform comparisons in the UV dimensions. This reduces the influence of luminance changes to our comparisons. Contemporary RGBD cameras provide all necessary information to extract the $(D(p), C(p))$ representation for each point p they observe.

For two 3D points p and q we first define their distance in space

$$D_E(p,q) = ||D(p) - D(q)||_2$$

and in color

$$D_C(p,q) = ||C(p) - C(q)||_2.$$

Given a point p, and a point cloud P, we define p to be depth-compatible to P if there exists a point q in P within a distance threshold τ_d from p. In notation, depth-compatibility

is expressed with a boolean function $DC(p,P)$ defined as

$$DC(p,P) = \begin{cases} 1 & \exists q \in P : D_E(p,q) < \tau_d \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Similarly, we define the notion of color compatibility:

$$CC(p,P) = \begin{cases} 1 & DC(p,P) \wedge \exists q \in P : D_C(p,q) < \tau_c \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

It should be noted that, according to Eq.(2), color compatibility is judged only for depth-compatible points.

3.2 Camera motion estimation

Given a dense 3D point cloud M that represents the environment and an RGBD frame that gives rise to another dense 3D point cloud P_c, a first goal is to register P_c to M. Since P_c is measured at a camera-centered coordinate system, this registration process gives, as a result, the rigid 3D motion that relates P_c to M which is the camera egomotion. We formulate this as an optimization (maximization) problem that is solved with a generative, hypothesize-and-test optimization framework. More specifically, if the camera motion with respect to M is m_{RT} then the transformed point cloud P'_c defined as

$$P'_c = m_{RT} P_c, \quad (3)$$

should be registered to M. In Eq.(3) m_{RT} is a $4x4$ transformation matrix modeling the motion of the camera with 6 degrees of freedom (3 for rotation and 3 for translation) and P_c is a $4xN$ matrix with the points in the observed point cloud in homogeneous coordinates. One way to evaluate the quality of a camera motion candidate solution m_{RT} is by measuring how effectively this motion registers P'_c with M. A quantification of the quality of registration is provided in the following objective function:

$$O(m_{RT}) = \sum_{p \in P'_c} DC(p,M) + \sum_{p \in P'_c} CC(p,M). \quad (4)$$

Intuitively, the first term of $O(m_{RT})$ measures the number of points in P'_c that are depth-compatible to the environment model M and the second term measures the number of points that are also color-compatible. Thus, a motion hypothesis m_{RT} that scores higher in $O(m_{RT})$ represents a motion that brings P'_c in better registration to M.

In general, the registration process could rely only on 3D structure information. However, in certain indoor environments, this proves to be insufficient. For example, if the camera moves along a long corridor, there is not enough information to infer the egomotion just from the changes in the structure of the observed point cloud. This is illustrated in Fig. 3, where even though the RGB input obviously changes as the camera moves, the depth images look almost identical. The incorporation of color-compatibility serves to disambiguate such situations.

According to Eq.(4), the registration of P'_c to M is based on all the points of P'_c. In practice, a much smaller set of points P_g that captures the environment structure proves

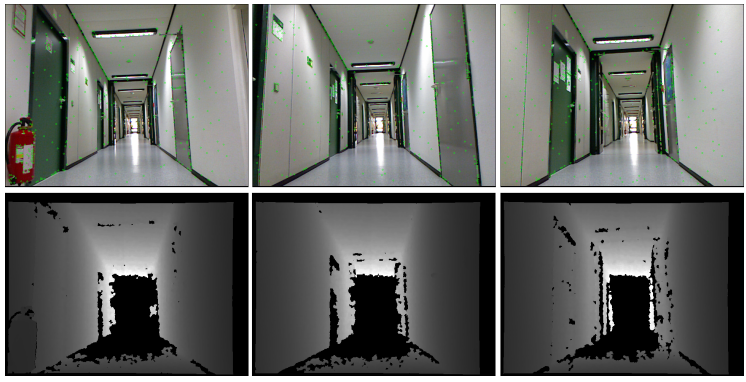

Fig. 3. Moving in a corridor. RGB (top) and depth (bottom) input for a sequence of 3 frames. Depth appears almost identical in all the frames, making motion estimation difficult unless scene color information is taken into account.

enough to solve the task. The points in P_g are chosen using two methods. First, we employ a corner detector [25] on the RGB part of the frame. These corners are then filtered and only the ones that have a depth value are kept. We also filter out corners that are associated with depth values but have low accuracy due to large quantization error [26]. Since the proposed method is using the scene structure in order to extract motion information and does not rely on feature matching, a simple corner detector is sufficient. Indeed exploiting cheap corners as features is one of the reasons the method requires low computational resources. Next, we choose the 3D points that are defined on a sparse (16×12) grid that is aligned to the RGBD input. This grid provides enough points in case there is not enough texture information in the observed RGB image. Using these two methods, the total number of selected points in a frame for a typical indoors scene is between 200 and 600. This represents a $0,2\%$ of the total number of points (640×480), a fact that reduces dramatically the computational requirements of the method.

The problem of egomotion estimation is now converted to the problem of finding the motion that maximizes Eq.(4). This is performed using the Particle Swarm Optimization (PSO) algorithm. It should be stressed that this top-down, generative, hypothesize-and-test solution to the problem is the one that remove the requirements for feature matches and, therefore, the need for elaborate feature extraction mechanisms.

3.3 Particle Swarm Optimization

The optimization (i.e., maximization) of the objective function defined in (Eq.(4)) is performed based on Particle Swarm Optimization (PSO) [19] which is a stochastic, evolutionary optimization method. It has been demonstrated that PSO is a very effective and efficient method for solving other vision optimization problems such as head pose estimation [23], hand articulation tracking [22] and others. PSO achieves optimization based on the collective behavior of a set of particles (candidate solutions) that

evolve in runs called generations. The rules that govern the behavior of particles emulate "social interaction". Essentially, a population of particles is a set of points in the parameter space of the objective function to be optimized. PSO has a number of attractive properties. For example, it depends on very few parameters, it does not require differentiation of the objective function to be minimized and converges to the solution with a relatively limited computational budget [2].

Every particle holds its current position (current candidate solution, set of parameters) in a vector x_t and its current velocity in a vector v_t. Each particle i keeps in vector p_i the position at which it achieved, up to the current generation t, the best value of the objective function. The swarm as a whole, stores the best position p_g across all particles of the swarm. All particles are aware of the global optimum p_g. The velocity and position update equations in every generation t are

$$v_t = K(v_{t-1} + c_1 r_1 (p_i - x_{t-1}) + c_2 r_2 (p_g - x_{t-1})) \tag{5}$$

and

$$x_t = x_{t-1} + v_t, \tag{6}$$

where K is a constant *constriction factor* [8]. In Eqs. (5), c_1 is called the *cognitive component*, c_2 is termed the *social component* and r_1, r_2 are random samples of a uniform distribution in the range $[0..1]$. Finally, $c_1 + c_2 > 4$ must hold [8]. In all experiments the values $c_1 = 2.8$, $c_2 = 1.3$ and $K = \frac{2}{\left|2-\psi-\sqrt{\psi^2-4\psi}\right|}$, with $\psi = c_1 + c_2$ were used.

In our problem formulation, the parametric space consists of the $6D$ space of camera motions m_{RT}. The rotation component of candidate camera moves is parametrized in terms of yaw (θ), pitch (ϕ), and roll (ω) angles, correspondingly yielding $R = R_x(\theta) \cdot R_y(\phi) \cdot R_z(\omega)$ for each parameter combination. Translation is parametrized by the XYZ coordinates of the camera center c. Particles are initialized at a normal distribution around the center of the search range with their velocities set to zero. Each dimension of the multidimensional parameter space is bounded in some range. During the position update, a velocity component may force a particle to move to a point outside the bounded search space. Such a component is zeroed and the particle does not move at the corresponding dimension. Since the camera motion needs to be continuously tracked in a sequence instead of being estimated in a single frame, temporal continuity is exploited. More specifically, the solution over frame t is used to restrict the search space for the initial population at frame $t+1$. In related experiments, the search range (or the space in which particle positions are initialized) extend $\pm 150\,mm$ and $\pm 10^\circ$ around the position estimated in the previous frame.

3.4 Handling camera motion estimation failures

A failure to estimate accurately the camera motion can be identified using the score of the objective function. If the score at a certain frame is below a threshold τ_s, then the camera motion calculated for I_t is considered inaccurate. This may occur in cases where there is not enough depth information in the scene, or when there are too many moving objects in the foreground. In our experiments we choose $\tau_s = 0.15|P_g|$ where $|P_g|$ is the number of points in P_g. If this condition holds then less than 15% of the scene features

where registed during the optimization step. In this case the camera motion estimation is considered unreliable, and no tracking steps or environment update is performed.

3.5 Identifying and tracking foreground objects

If the motion m_{RT} of the camera with respect to the environment model M is known, we define a point $p \in P_c$ to be an outlier when $p' = m_{RT} * p$ and p' is not color-compatible to M. Formally, the set of outliers is defined as:

$$O_c = \{p \in P_c : p' = m_{RT} * p \wedge CC(p', M) = 0\}. \quad (7)$$

The set of outliers O_c is partitioned to the following classes:

- M-class: Independently moving foreground objects (e.g., moving humans).
- S-class: Static background objects that are not yet part of the environment model. This is because as the camera moves in an unknown environment, new parts of the "static" background will become visible.
- N-class: Sensor noise. Depth measurements obtained from commercial RGBD sensors like the Microsoft Kinect and the Asus Xtion have a considerable amount of noise, especially at distances greater than 3-4 meters.

The correct assignment of outliers to these classes is very important. N-class points need to be discarded from further consideration. M-class points should support the formation of hypotheses for objects to be tracked. Finally, S-class points need to be incorporated in M. To perform this type of classification in a robust way, we capitalize on the fact that the camera moves in an environment where objects move on the ground plane. Thus, we aggregate all observations on a floor plan F of the environment defined as

$$F = \Pi V O_c. \quad (8)$$

In Eq.(8), Π is a $4x4$ orthographic projection matrix and V is the $4x4$ camera view matrix for a virtual camera above the scene, while O_c is the $4xN$ matrix of the outlier points in homogeneous coordinates. Then, F is a 2D floor plan of the outliers which can be efficiently grouped using a blob detector and tracker. An example F is shown in Fig. 4 for one frame of an indoors sequence. For a blob B of outliers we define the following properties:

- Blob height B_h: The maximum distance of a point in the blob from the ground floor.
- Blob area B_a: The area occupied by the blob on the ground plane.
- Blob size B_s: The number of outlier points that produced the blob.

Simple rules defined over the parameters B_h, B_a and B_s suffice to identify outliers and to characterize tracking candidates for the next phase of the algorithm. When choosing the values for the blob classification parameters one must take into account that small errors in the camera motion estimation due to noise or accumulated drift can create relatively large number of outlier clusters. This limits the smallest size of a moving object that can be detected, since allowing small blobs to be considered possible candidates will result in a great number of false positive tracks.

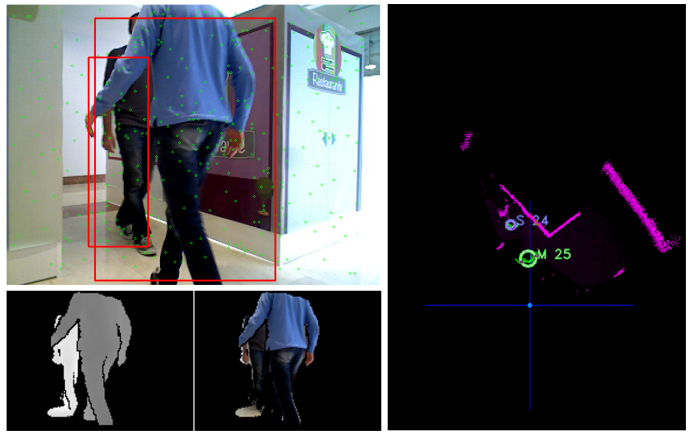

Fig. 4. Segmentation and tracking. In the floor plan view (right image), background objects are shown in purple and foreground objects in green. Foreground objects are segmented (bottom left) and tagged. The RGBD camera is mounted on a prototype smart walker during trials. The current location of the camera is on the center of the blue cross.

In our use cases we are mainly interested in tracking human-sized objects, which is well above the size limits of the method. For our experiments the B_h was set to be between $1.0m$ and $1.8m$ while the area of the blobs, B_a, was limited between $0.19m^2$ and $0.78m^2$. The B_s parameter was empirically chosen to be less than 500, since errors on the egomotion tend to produce blobs with high number of outliers.

In order to track foreground objects, we create and maintain a number of hypotheses about moving objects in the scene. It is assumed that at the previous frame $t-1$ there have been O_{t-1} object hypotheses with which the current set of blobs need to be associated with. This is performed with a variant of the blob tracking method proposed in [3]. The original version of that tracker was devised to track an unknown/dynamically changing number of skin coloured blobs. Instead, our own implementation of this algorithm operates on the detected blobs corresponding to moving object hypotheses.

3.6 Updating the environment model

The environment model is built and maintained on the fly, without any prior knowledge. In order to bootstrap the process, the point cloud P_c produced in the first frame I_0, is used as the initial model M and the location of the sensor on this frame is set to be the world origin. To update the environment model we use the background points of frame I_n created on Sec. 3.5 registered on the environment model M of frame I_{n-1} using the camera motion m_{RT} (Eq. 3).

The environment model should be updated only when there is high confidence on the quality of the calculated m_{RT}. In our experiments, we set the minimum threshold for the quality of the camera motion estimation as explained in Sec. 3.4 to be $\tau_u = 0.30|P_g|$.

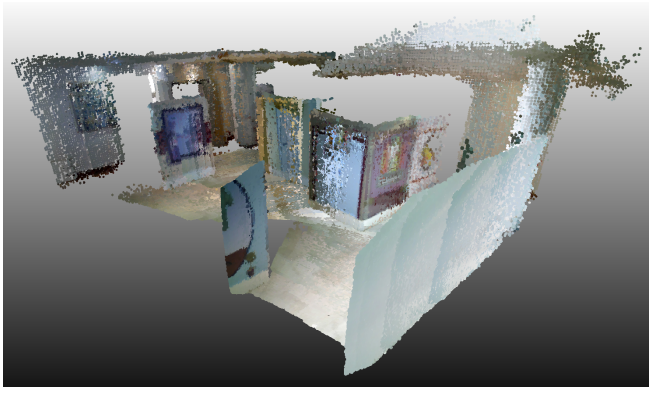

Fig. 5. The resulting point cloud (with color) created from a sequence grabbed using the prototype smart walker. The method produces an accurate model of the environment.

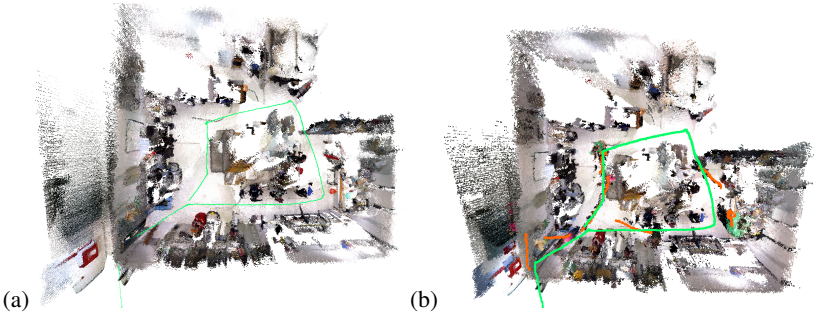

Fig. 6. (a) A floormap of an environment built in the absence of moving objects. The estimated platform trajectory shown in green. (b) The map of the environment that was built in presence of moving objects. Camera (green) and moving object (orange) trajectories are also shown.

Choosing this value for τ_u means that when less than 30% of the scene features where registered during the optimization step, the ego-motion estimation may not be accurate enought to update the scene model.

The resulting environment model is used in order to detect moving objects in the next frame. In Fig. 5 the point cloud of a full environment model from a test dataset is shown. In practice, there is no need to keep the whole environment model in memory in order to perform object tracking. As the camera moves in the environment, older parts of the model can be discarded once they are far enough from the platform or when they get older than a pre-set age. In our experiments, the environment model is discarded every 40 frames but the knowledge about the moving objects in the scene is kept during the reset. This way, we reduce the amount of resources required to maintain a constantly expanding model, without loosing information about the foreground in the scene.

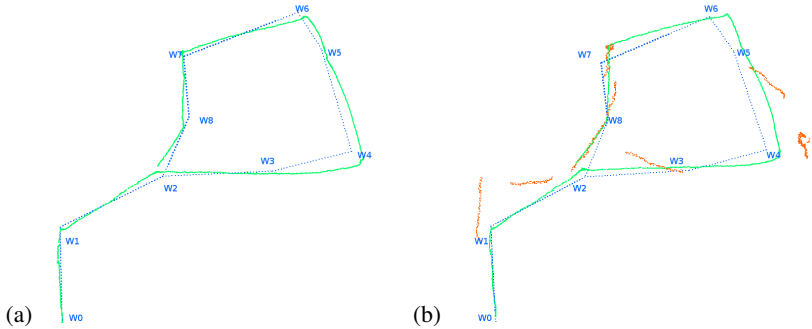

Fig. 7. View of the calculated camera trajectory (green) aligned with the ground truth trajectory (blue) in two experiments without MOT (a) and with MOT (b). The trajectories of the moving objects for the second experiment are also shown in orange. The user of the walker was asked to follow the same track in both experiments.

4 Experiments

The proposed approach has been evaluated quantitatively and qualitatively in a number of experiments. The experiments were performed in an indoors environment whose reconstructed top down view is shown in Fig. 6. The environment consists of a hallway and a room with a total area of $120m^2$. A number of way-points were set in order to create a closed path. The location of each way-point was measured and used to compute a ground truth trajectory that the user of the walker was asked to follow. The error on the real world location of the way-points was measured to be around $30mm$. A total of 9 way-points (marked from $W0$ to $W8$) were set creating a 9 segment closed path that goes through the corridor and around the room. That way the system was evaluated on a demanding real world scenario similar to the cluttered indoor environments that the actual device will operate in.

In a first experiment, we assessed the SLAM aspects of the proposed method, that is, we performed simultaneous localization and map construction in the absence of moving people. A user moved with the walker along the pre-planned track. The results of the reconstruction as well as the computed camera trajectory is shown in Fig. 6(a).

Figure 7(a) shows the ground truth trajectory of the walker (blue color) and the one estimated by the proposed method (green color). In the same figure, the locations of the way-points are marked along the path. Table 1 shows the ground truth measurements for the distances of these points (first column) and those estimated by the proposed method (second column). It can be verified that the estimation of these distances are rather accurate.

The same experiment was repeated in the presence of moving people. The rightmost column of Table 1 shows the estimated distances in this case. The actual reconstruction, moving object trajectories and camera trajectory are shown in Fig. 6(b). It can be verified that camera and object motion trajectories are rather accurate. Additionally, the independently moving objects did not affect considerably the SLAM process. This is

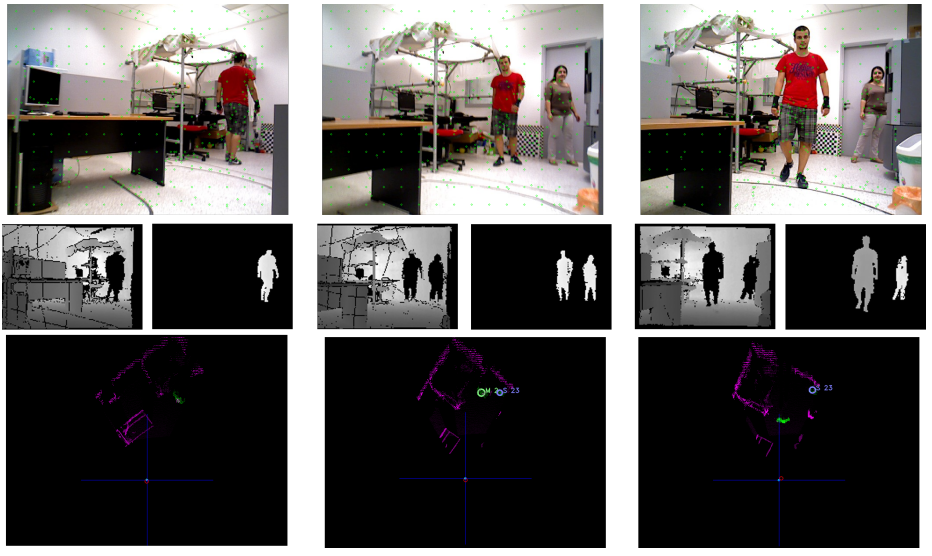

Fig. 8. Tracking moving targets. Three frames from a lab sequence. On the top row: The RGB input for each frame is shown with the points selected for the ego-motion estimation superimposed with green. Middle row: The environment model M and the foreground moving objects H from the point of view of the camera. Bottom row: The top down view of the scene. An orthographic projection of both the environment (purple) and the tracked objects (green).

also illustrated in Fig. 7(b) that shows the estimated track of the camera and the moving objects, aligned with the ground truth trajectory. Further intermediate results are illustrated in Fig. 8.

For the whole course of the experiments, we also measured the average distance of the walker position as estimated by the method, to the corresponding line segment (ground truth). Table 2 shows these localization errors in the experiments with and without moving objects. It can be verified that, on average, the errors are comparable. A full run of the experiments is also shown on the supplementary material of this paper[1].

A prototype walker implementation has also been exhibited in the Vilnius ICT'2013 Conference[2] with great success. In that context, the DALi c-Walker had to detect and track people moving in very crowded environments.

5 Summary

In this paper, we presented a new approach to the problem of simultaneous object tracking, localization and map construction. The proposed method exploits the visual input provided by a single RGBD camera. The method can handle an arbitrary and temporally varying number of moving objects which it localizes on a map of the environment

[1] `https://www.youtube.com/user/AntonisArgyros/videos`
[2] `http://ec.europa.eu/digital-agenda/en/ict-2013`

Table 1. Localization errors (in mm)

Segment	Average error without MO	Average error with MO
W0-W1	45	32
W1-W2	125	121
W2-W3	68	200
W3-W4	333	233
W4-W5	197	444
W5-W6	136	431
W6-W7	118	286
W7-W8	101	127
W8-W2	140	253
All	140	236

Table 2. Actual and measured distances (in mm)

Segment	Actual length	Measured without MO	Measured with MO
W0-W1	3222	3144	3121
W1-W2	3816	3886	3968
W2-W3	3643	3668	3639
W3-W4	2713	3011	3141
W4-W5	3559	3599	3719
W5-W6	1423	1183	1374
W6-W7	4032	4067	3760
W7-W8	1999	1923	1986
W8-W2	2150	1954	1877

which is progressively built from scratch. The obtained quantitative experimental results demonstrate that the method is capable of handling effectively challenging SLAMMOT scenarios. It has also been shown that the SLAM accuracy of the method is not affected significantly by the presence of moving objects which are also tracked accurately. The computational requirements of the method are rather low as its implementation on conventional contemporary architectures performs at super real time frame rates while being able to achieve near real time performance on ARM based embedded systems. Thus, with the proposed method, we achieve a robust solution to several interesting problems under loose assumptions and with limited computational resources. A practical proof of the above has been the successful deployment of the approach in use cases using the DALi c-Walker prototype.

References

1. M. Agrawal, K. Konolige, and L. Iocchi. Real-time detection of independent motion using stereo. In *IEEE WACV/MOTIONS '05*, volume 2, pages 207–214, 2005.
2. P.J. Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. *Evolutionary Programming VII, LNCS*, 1447:601–610, 1998.
3. Antonis A. Argyros and Manolis I. A. Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *In: ECCV*, pages 368–379, 2004.
4. Charles Bibby and Ian Reid. Robust real-time visual tracking using pixel-wise posteriors. In David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, editors, *ECCV (2)*, volume 5303 of *Lecture Notes in Computer Science*, pages 831–844. Springer, 2008.
5. Chieh chih Wang, Charles Thorpe, Martial Hebert, Sebastian Thrun, and Hugh Durrant-whyte. Simultaneous localization, mapping and moving object tracking. *International Journal of Robotics Research*, 2004.
6. Wongun Choi, Caroline Pantofaru, and Silvio Savarese. Detecting and tracking people using an rgb-d camera via multiple detector fusion. In *ICCV Workshops*, pages 1076–1083. IEEE, 2011.

7. Wongun Choi, Caroline Pantofaru, and Silvio Savarese. A general framework for tracking multiple people from a moving camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7):1577–1591, 2013.
8. M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.
9. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, USA, 2005.
10. F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3d mapping with an RGB-D camera. *IEEE Transactions on Robotics (T-RO)*, 2013.
11. Andreas Ess, Bastian Leibe, Konrad Schindler, and Luc J. Van Gool. A mobile vision system for robust multi-person tracking. In *CVPR*. IEEE Computer Society, 2008.
12. Andreas Ess, Bastian Leibe, Konrad Schindler, and Luc J. Van Gool. Robust multiperson tracking from a mobile platform. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(10):1831–1846, 2009.
13. Pedro F. Felzenszwalb, Ross B. Girshick, David A. McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, 2010.
14. V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, Alaska, 2008.
15. Jorge Fuentes-Pacheco, Jos Ruiz-Ascencio, and JuanManuel Rendn-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, pages 1–27, 2012.
16. G. Gate, A. Breheret, and F. Nashashibi. Fast pedestrian detection in dense environment with a laser scanner and a camera. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1–6, 2009.
17. Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *Int. J. Rob. Res.*, 31(5):647–663, April 2012.
18. Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *In Proc. UIST*, pages 559–568, 2011.
19. J. Kennedy, R.C. Eberhart, and Yuhui. Shi. *Swarm intelligence*. Morgan Kaufmann Publishers, 2001.
20. Matthias Luber, Luciano Spinello, and Kai O. Arras. People tracking in rgb-d data with on-line boosted target models. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, USA, 2011.
21. David Marrquez-Gomez and Michel Devy. Active visual-based detection and tracking of moving objects from clustering and classification methods. In *Proceedings of the 14th International Conference on Advanced Concepts for Intelligent Vision Systems*, ACIVS'12, pages 361–373, Berlin, Heidelberg, 2012. Springer-Verlag.
22. Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. Tracking the articulated motion of two strongly interacting hands. In *CVPR*. IEEE, June 2012.
23. Pashalis Padeleris, Xenophon Zabulis, and Antonis A. Argyros. Head pose estimation on depth data based on particle swarm optimization. In *IEEE CVPRW - HAU3D 2012*, 2012.
24. Deva Ramanan, David A. Forsyth, and Andrew Zisserman. Tracking people by learning their appearance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(1):65–81, 2007.

25. Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer Berlin Heidelberg, 2006.
26. Jan Smisek, Michal Jancosek, and Toms Pajdla. 3d with kinect. In *ICCV Workshops*, pages 1154–1160. IEEE, 2011.
27. Joan Solà. *Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilistic Approach.* PhD thesis, Institut National Polytechnique de Toulouse, 2007.
28. P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, Nice, France, 2003.
29. Somkiat Wangsiripitak and David W. Murray. Avoiding moving outliers in visual slam by tracking moving objects. In *IEEE ICRA'09*, pages 705–710, Piscataway, NJ, USA, 2009. IEEE Press.
30. Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), 2006.
31. Tao Zhao and Ramakant Nevatia. Tracking multiple humans in crowded environment. In *CVPR (2)*, pages 406–413, 2004.