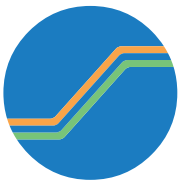


MTA SZTAKI

INSTITUTE FOR COMPUTER SCIENCE AND CONTROL
HUNGARIAN ACADEMY OF SCIENCES

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMÍTÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓINTÉZET

4D Scene Reconstruction in Multi-Target Scenarios

Csaba Benedek — Zsolt Jankó — Csaba Horváth — Dömötör Molnár — Dmitry Chetverikov — Tamás Szirányi

Technical Report

N° i4D-3

January 2013

Theme VISION

MTA SZTAKI
H-1111 Budapest, Kende u. 13-17, Hungary
H-1518 Budapest P.O.B. 63, Hungary
Phone: (+36 1) 279 6000
Fax: (+36 1) 466 7503

MTA SZTAKI

4D Scene Reconstruction in Multi-Target Scenarios

Csaba Benedek* , Zsolt Jankó† , Csaba Horváth* , Dömötör Molnár* , Dmitry Chetverikov† , Tamás Szirányi*

Theme VISION — Computer Vision
Divison: Distributed Events Analysis Research Laboratory & Geometric Modeling and Computer Vision Laboratory

Research report — January 2013 — 30 pages

Abstract: In this report, we introduce a complex approach on 4D reconstruction of dynamic scenarios containing multiple walking pedestrians. The input of the process is a point cloud sequence recorded by a rotating multi-beam Lidar sensor, which monitors the scene from a fixed position. The output is a geometrically reconstructed and textured scene containing moving 4D people models, which can follow in real time the trajectories of the walking pedestrians observed on the Lidar data flow. Our implemented system consists of four main steps. *First*, we separate foreground and background regions in each point cloud frame of the sequence by a robust probabilistic approach. *Second*, we perform moving pedestrian detection and tracking, so that among the point cloud regions classified as foreground, we separate the different objects, and assign the corresponding people positions to each other over the consecutive frames of the Lidar measurement sequence. *Third*, we geometrically reconstruct the ground, walls and further objects of the background scene, and texture the obtained models with photos taken from the scene. *Fourth* we insert into the scene textured 4D models of moving pedestrians which were preliminary created in a special 4D reconstruction studio. *Finally*, we integrate the system elements in a joint dynamic scene model and visualize the 4D scenario.

Key-words: rotating multi-beam Lidar, MRF, motion segmentation, 4D reconstruction

This work is connected to the i4D project funded by the internal R&D grant of MTA SZTAKI. The first author was also supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and by the Grant #101598 of the Hungarian Research Fund (OTKA)

* Distributed Events Analysis Research Laboratory, http://web.eee.sztaki.hu
† Geometric Modelling and Computer Vision Laboratory, http://visual.ipan.sztaki.hu

Institute for Computer Science and Control
Hungarian Academy of Sciences
H-1111, Budapest Kende utca 13-17, Hungary
Telephone: +36 1 279 6000, Fax: +36 1 466 7503
http://www.sztaki.hu

Helyszínek 4D rekonstrukciója több célpont követésével

Kivonat : Riportunkban komplex módszert mutatunk be dinamikus helyszínek 4D rekonstrukciójára, feltételezve, hogy a megfigyelt területen több mozgó személy tartózkodhat egyszerre. A rendszer bemenete egy fix pozícióban álló forgó Lidar eszközzel készült pontfelhő szekvencia. A kimenet egy geometriailag rekonstruált és textúrázott helyszín mozgó emberek 4D modelljeivel, melyek valós időben követhetik a gyalogosok Lidar adatok alapján megfigyelt valódi útpályáit. A létrehozott rendszer négy fő lépésből áll. *Először* elkülönítjük az előtér és háttér részeket a mért pontfelhőkben egy robosztus valószínűségi modellel. *Másodszor* detektáljuk és követjük a mozgó gyalogosokat, úgy hogy az előtérként osztályozott pontfelhő-régiókon belül elkülönítjük a különálló objektumokat, majd az összetartozó emberpozíciókat összerendeljük az egymást követő időkeretekben. *Harmadik lépésben* rekonstruáljuk a talaj, falak és további objektumok geometriáját, és textúrázzuk az elkészült háromszögelt modellt a helyszínről készített fotók segítségével. *Negyedik* elemként, a rendszer indítása előtt előzetesen elkészítünk és rögzítünk textúrált 4D modelleket mozgó gyalogosokról egy speciális 4D rekonstrukciós stúdióban. Végül integráljuk az egyes rendszerelemeket egy közös dinamikus helyszínmodellben és megjelenítjük a 4D jelenetet.

Kulcsszavak : Lidar, Markov mező, mozgáskövetés, 4D rekonstrukció

Contents

1 Introduction

Foreground detection and segmentation are a key issues in automatic visual surveillance. Foreground areas usually contain the regions of interest, moreover, an accurate object-silhouette mask can directly provide useful information for, among others, people or vehicle detection, tracking, activity analysis, or biometrical analysis. Another important application is the virtual reconstruction of the scene, which can be used in interactive virtual reality systems, like animation or movie industry.

Range image sequences offer significant advantages versus conventional video flows for scene segmentation, since geometrical information is directly available [1, 2], which can provide more reliable features than intensity, color or texture values [3, 4]. Using Time-of-Light (ToF) cameras [1] or scanning Lidar sensors [5] enable recording range images independently of the outside illumination conditions and we can also avoid artifacts of stereo vision techniques. From the point of view of data analysis, ToF cameras record depth image sequences over a regular 2D pixel lattice, where established image processing approaches, such as Markov Random Fields (MRFs) can be adopted for smooth and observation consistent segmentation [4]. However, such cameras have a limited Field of View (FoV), which can be a drawback for surveillance and monitoring applications.

Rotating multi-beam Lidar systems (RMB-Lidar) provide a 360° FoV of the scene, with a vertical resolution equal to the number of the sensors, while the horizontal angle resolution depends on the speed of rotation (see Fig. 2). For efficient data processing, the 3-D RMB-Lidar points are often projected onto a cylinder shaped range image [5, 6]. However, this mapping is usually ambiguous: On one hand, several laser beams with slight orientation differences are assigned to the same pixel, although they may return from different surfaces. As a consequence, a given pixel of the range image may represent different background objects at the consecutive time steps. This ambiguity can be moderately handled by applying multi-modal distributions in each pixel for the observed background-range values [5], but the errors quickly aggregate in case of dense background motion, which can be caused e.g. by moving vegetation. On the other hand, due to physical considerations, the raw data of distance, pitch and angle provided by the RMB-Lidar sensor must undergo a strongly non-linear calibration step to obtain the Euclidean point coordinates [7], therefore, the density of the points mapped to the regular lattice of the cylinder surface may be inhomogeneous. To avoid the above artifacts of background modeling, [6] has directly extracted the foreground objects from the range image by mean-shift segmentation and blob detection. However, we have experienced that if the scene has simultaneously several moving and static objects in a wide distance range, the moving pedestrians are often merged into the same blob with neighboring scene elements.

Instead of projecting the points to a range image, another way is to solve the foreground detection problem in the spatial 3D domain. However, 3D object level techniques principally aim to extract the bounding boxes of the pedestrians [8], instead of labeling each foreground point of the input cloud, which may be necessary for activity recognition by e.g. skeleton fitting to the silhouettes. MRF techniques based on 3D spatial point neighborhoods

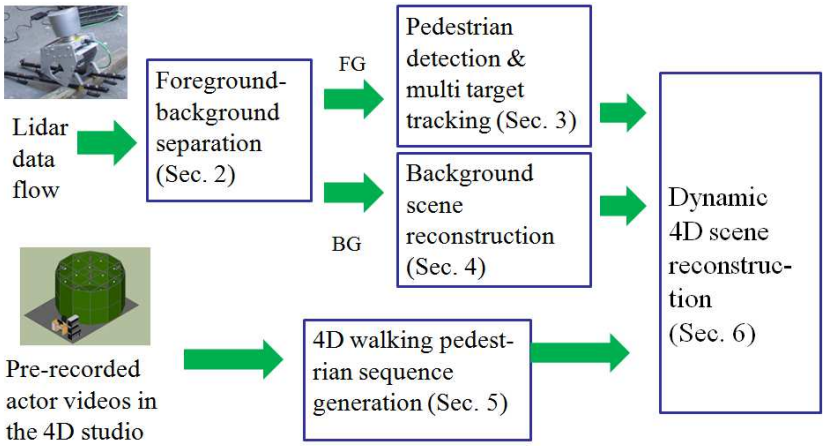


Figure 1: Flowchart of the proposed 4D scene reconstruction system, with marking for each step the corresponding section of the report

are frequently applied in remote sensing [9], however the accuracy is low in case of small neighborhoods, otherwise the computational complexity rapidly increases.

In this report, we propose a hybrid approach for dense foreground-background point labeling in a point cloud obtained by a RMB-Lidar system, which monitors the scene from a fixed position. Our method solves the computationally critical spatial filtering steps in the 2D range image domain by an MRF model, however, ambiguities of discretization are handled by joint consideration of the true 3D positions and the 2D labels. Using a spatial foreground model, we significantly decrease the spurious effects of irrelevant background motion, which is mainly caused by moving tree crowns. We provide evaluation versus three reference methods using our 3D point cloud Ground Truth (GT) annotation tool. Thereafter, we perform moving pedestrian detection and tracking, so that among the point cloud regions classified as foreground, we separate the different objects, and assign the corresponding people positions to each other over the consecutive frames of the Lidar measurement sequence. Next we transform the point cloud into a polygon mesh, with maintaining the information about individual objects, such as ground, walls, trees and further objects of the background scene, then we texture the obtained models with photos taken from the scene. Before starting the system, we create and record textured 4D models of moving pedestrians in a special 4D reconstruction studio. Finally, we integrate the system elements in a joint dynamic scene model and visualize the 4D scenario. The output is a geometrically reconstructed and textured scene containing moving 4D people models, which can follow in real time the trajectories of the walking pedestrians observed on the Lidar data flow. The flowchart of the developed system is shown Fig. 1 with marking for each step the corresponding section of the report.

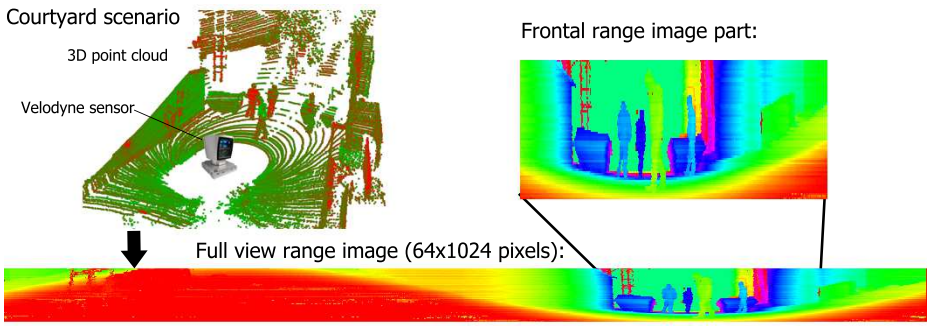


Figure 2: Point cloud recording and range image formation with a Velodyne HDL 64E RMB-Lidar sensor

2 Foreground-background separation

In this section, we propose a probabilistic approach for foreground segmentation in 360°-view-angle range data sequences, recorded by a rotating multi-beam Lidar sensor, which monitors the scene from a fixed position. To ensure real-time operation, we project the irregular point cloud obtained by the Lidar, to a cylinder surface yielding a depth image on a regular lattice, and perform the segmentation in the 2D image domain. Spurious effects resulted by quantification error of the discretized view angle, non-linear position corrections of sensor calibration, and background flickering, in particularly due to motion of vegetation, are significantly decreased by a dynamic MRF model, which describes the background and foreground classes by both spatial and temporal features. Evaluation is performed on real Lidar sequences concerning both video surveillance and traffic monitoring scenarios. The model has been originally published in [10].

2.1 Problem formulation and data mapping

Assume that the RMB-Lidar system contains R vertically aligned sensors, and rotates around a fixed axis with a possibly varying speed[1]. The output of the Lidar within a time frame t is a *point cloud* of $l^t = R \cdot c^t$ points: $\mathcal{L}^t = \{p_1^t, \ldots, p_{l^t}^t\}$. Here c^t is the number of point *columns* obtained at t, where a given column contains R concurrent measurements of the R sensors, thus c^t depends on the rotation speed. Each point, $p \in \mathcal{L}^t$, is associated to sensor distance $d(p) \in [0, D_{\max}]$, pitch index $\hat{\vartheta}(p) \in \{1, \ldots, R\}$ and yaw angle $\varphi(p) \in [0, 360°]$ parameters. $d(p)$ and $\hat{\vartheta}(p)$ are directly obtained from the Lidar's data flow, by taking the measured distance and sensor index values corresponding to p. Yaw angle $\varphi(p)$ is calculated from the

[1]The speed of rotation can often be controlled by software, but even in case of constant control signal, we must expect minor fluctuations in the measured angle-velocity, which may result in different number of points for different 360° scans in time.

Euclidean coordinates of p projected to the ground plane, since the R sensors have different horizontal view angles, and the angle correction of calibration may also be significant [7].

The goal of the proposed method is at a given time frame t to assign each point $p \in \mathcal{L}^t$ to a label $\omega(p) \in \{\text{fg}, \text{bg}\}$ corresponding to the moving object (i.e. foreground, fg) or background classes (bg), respectively.

For efficient data manipulation, we also introduce a range image mapping of the obtained 3D data. We project the point cloud to a cylinder, whose central basis point is the ground position of the RMB-Lidar and the axis is perpendicular to the ground plane. Note that slightly differently from [6], this mapping is also efficiently suited to configurations, where the Lidar axis is tilted do increase the vertical Field of View. Then we stretch a $S_H \times S_W$ sized 2D pixel lattice S on the cylinder surface, whose height S_H is equal to the R sensor number, and the width S_W determines the fineness of discretization of the yaw angle. Let us denote by s a given pixel of S, with $[y_s, x_s]$ coordinates. Finally, we define the $\mathcal{P} : \mathcal{L}^t \to S$ point mapping operator, so that y_s is equal to the pitch index of the point and x_s is set by dividing the $[0, 360°]$ domain of the yaw angle into S_W bins:

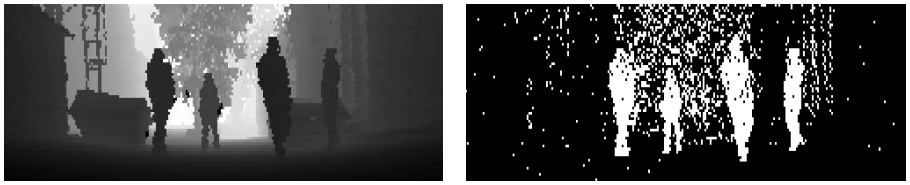
$$s \stackrel{\text{def}}{=} \mathcal{P}(p) \text{ iff } y_s = \hat{\vartheta}(p),\ x_s = \text{round}\left(\varphi(p) \cdot \frac{S_W}{360°}\right) \tag{1}$$

2.2 Background model

The background modeling step assigns a fitness term $f_{\text{bg}}(p)$ to each $p \in \mathcal{L}^t$ point of the cloud, which evaluates the hypothesis that p belongs to the background. The process starts with a cylinder mapping of the points based on (1), where we use a $R \times S_W^{\text{bg}}$ pixel lattice S^{bg} (R is the sensor number). Similarly to [5], for each s cell of S^{bg}, we maintain a Mixture of Gaussians (MoG) approximation of the $d(p)$ distance histogram of p points being projected to s. Following the approach of [11], we use a fixed K number of components (here $K = 5$) with weight w_s^i, mean μ_s^i and standard deviation σ_s^i parameters, $i = 1 \dots K$. Then we sort the weights in decreasing order, and determine the minimal k_s integer which satisfies $\sum_{i=1}^{k_s} w_s^i > T_{\text{bg}}$(we used here $T_{\text{bg}} = 0.89$). We consider the components with the k_s largest weights as the background components. Thereafter, denoting by $\eta()$ a Gaussian density function, and by $\mathcal{P}^{\text{bg}}$ the projection transform onto S^{bg}, the $f_{\text{bg}}(p)$ background evidence term is obtained as:

$$f_{\text{bg}}(p) = \sum_{i=1}^{k_s} w_s^i \cdot \eta\left(d(p), \mu_s^i, \sigma_s^i\right),\ \text{where } s = \mathcal{P}^{\text{bg}}(p). \tag{2}$$

The Gaussian mixture parameters are set and updated based on [11], while we used $S_W^{\text{bg}} = 2000$ angle resolution, which provided the most efficient detection rates in our experiments. By thresholding $f_{\text{bg}}(p)$, we can get a dense foreground/background labeling of the point cloud [5, 11] (referred later as *Basic MoG* method), but as shown in Fig. 7(a),(c), this classification is notably noisy in scenarios recorded in large outdoor scenes.



(a) Range image part (90° horiz. view) (b) Basic MoG [5, 11]

(c) uniMRF [3] (d) Proposed DMRF segmentation

Figure 3: Foreground segmentation in a range image part with three different methods

2.3 DMRF approach on foreground segmentation

In this section, we propose a Dynamic Markov Random Field (DMRF) model to obtain smooth, noiseless and observation consistent segmentation of the point cloud sequence. Since MRF optimization is computationally intensive [12], we define the DMRF model in the range image space, and 2D image segmentation is followed by a point classification step to handle ambiguities of the mapping. As defined by (1) in Sec. 2.1, we use a $\mathcal{P}$ cylinder projection transform to obtain the range image, with a $S_W = \min(\hat{c}, S_W^{\text{bg}}/2)$ grid with, where $\hat{c}$ denotes the expected number of point columns of the point sequence in a time frame. By assuming that the rotation speed is slightly fluctuating, this selected resolution provides a dense range image. Let us denote by $P_s \subset \mathcal{L}^t$ the set of points projected to pixel s. For a given direction, foreground points are expected being closer to the sensor than the estimated mean background range value. Thus, for each pixel s we select the closest projected point $p_s^t = \text{argmin}_{p \in P_s} d(p)$, and assign to pixel s of the range image the $d_s^t = d(p_s^t)$ distance value. For pixels with undefined range values ($P_s = \emptyset$), we interpolate the d_s^t distance from the neighborhood. For spatial filtering, we use an eight-neighborhood system in S, and denote by $N_s \subset S$ the neighbors of pixel s.

Next, we assign to each $s \in S$ foreground and background energy (i.e. negative fitness) terms, which describe the class memberships based on the observed $d(s)$ values. The background energies are directly derived from the parametric MoG probabilities using (2):

$$\varepsilon_{\text{bg}}^t(s) = -\log\left(f_{\text{bg}}(p_s^t)\right).$$

For description of the foreground, using a constant ε_{fg} could be a straightforward choice [3] (we call this approach *uniMRF*), but this uniform model results in several false alarms

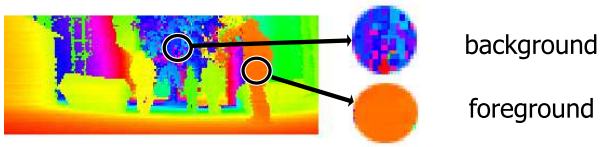


Figure 4: Demonstrating the different local range value distributions in the neighborhood of a given foreground and background pixel, respectively

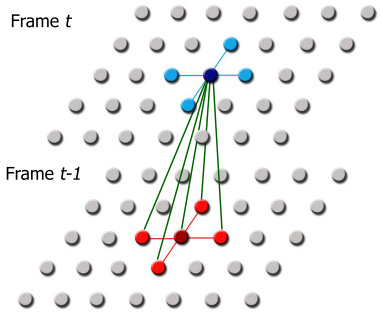


Figure 5: Structure of the dynamic MRF model

due to background motion and quantitization artifacts. Instead of temporal statistics, we use spatial distance similarity information to overcome this problem by using the following assumption: whenever s is a foreground pixel, we should find foreground pixels with similar range values in the neighborhood (Fig. 4). For this reason, we use a non-parametric kernel density model for the foreground class:

$$\varepsilon^t_{\mathrm{fg}}(s) = \sum_{r \in N_s} \zeta(\varepsilon^t_{\mathrm{bg}}(r), \tau_{\mathrm{fg}}, m_\star) \cdot k\left(\frac{d^t_s - d^t_r}{h}\right),$$

where h is the kernel bandwidth and $\zeta : \mathbb{R} \to [0, 1]$ is a sigmoid function:

$$\zeta(x, \tau, m) = \frac{1}{1 + \exp(-m \cdot (x - \tau))}.$$

We use here a uniform kernel: $k(x) = \mathbf{1}\{|x| \leq 1\}$, where $\mathbf{1}\{.\} \in \{0, 1\}$ is the binary indicator function of a given event.

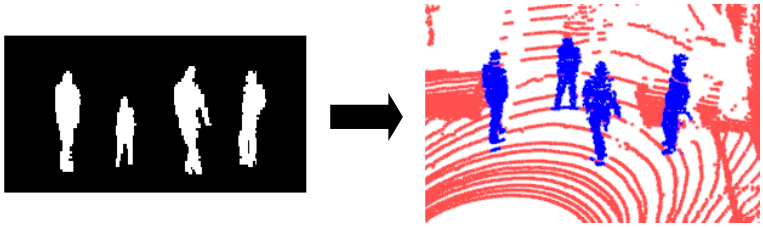


Figure 6: Backprojection of the range image labels to the point cloud

To formally define the range image segmentation task, to each pixel $s \in S$, we assign a $\omega_s^t \in \{\text{fg}, \text{bg}\}$ class label so that we aim to minimize the following energy function:

$$E = \sum_{s \in S} V_D(d_s^t|\omega_s^t) + \underbrace{\sum_{s \in S} \sum_{r \in N_s} \alpha \cdot \mathbf{1}\{\omega_s^t \neq \omega_r^{t-1}\}}_{\xi_s^t} + \underbrace{\sum_{s \in S} \sum_{r \in N_s} \beta \cdot \mathbf{1}\{\omega_s^t \neq \omega_r^t\}}_{\chi_s^t}, \tag{3}$$

where $V_D(d_s^t|\omega_s^t)$ denotes the data term, while ξ_s^t and χ_s^t are the temporal and spatial smoothness terms, respectively, with $\alpha > 0$ and $\beta > 0$ constants. Let us observe, that although the model is dynamic due to dependencies between different time frames (see the ξ_s^t term), to enable real time operation, we develop a causal system, i.e. labels from the past are not updated based on labels from the future.

The data terms are derived from the data energies by sigmoid mapping:

$$V_D(d_s^t|\omega_s^t = \text{bg}) = \zeta(\varepsilon_{\text{bg}}^t(s), \tau_{\text{bg}}, m_{\text{bg}})$$
$$V_D(d_s^t|\omega_s^t = \text{fg}) = \begin{cases} 1, & \text{if } \; d_s^t > \max_{\{i=1\ldots k_s\}} \mu_s^{i,t} + \epsilon \\ \zeta(\varepsilon_{\text{fg}}^t(s), \tau_{\text{fg}}, m_{\text{fg}}), & \text{otherwise.} \end{cases}$$

The sigmoid parameters τ_{fg}, τ_{bg}, m_{fg}, m_{bg} and $m_\star$ can be estimated by Maximum Likelihood strategies based on a few manually annotated training images. As for the smoothing factors, we use $\alpha = 0.2$ and $\beta = 1.0$ (i.e. the spatial constraint is much stronger), while the kernel bandwidth is set to $h = 30$cm. The MRF energy (3) is minimized via the fast graph-cut based optimization algorithm [12].

The result of the DMRF optimization is a binary foreground mask on the discrete S lattice. As shown in Fig. 6, the final step of the method is the classification of the points of the original $\mathcal{L}$ cloud, considering that the projection may be ambiguous, i.e. multiple points with different true class labels can be projected to the same pixel of the segmented range image. With denoting by $s = \mathcal{P}(p)$ for time frame t:

- $\omega(p) = \text{fg}$, iff one of the following two conditions holds:
  (a) $\omega_s^t = \text{fg}$ and $d(p) < d_s^t + 2 \cdot h$
  (b) $\omega_s^t = \text{bg}$ and $\exists r \in N_r : \{\omega_r^t = \text{fg}, |d_r^t - d(p)| < h\}$
- $\omega(p) = \text{bg}$: otherwise.

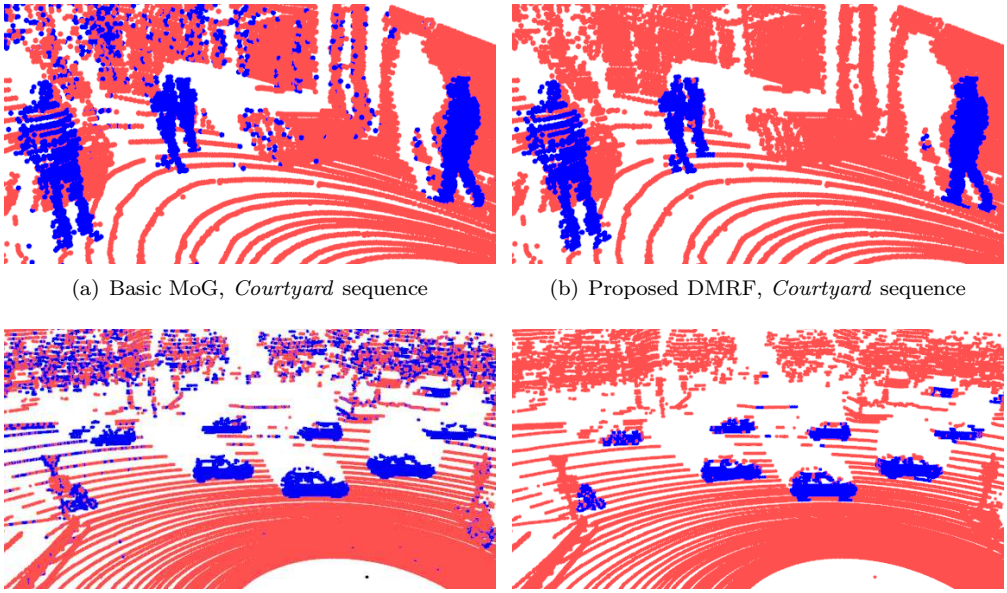
The above constraints eliminate several (a) false positive and (b) false negative foreground points, projected to pixels of the range image near the object edges.

2.4 Evaluation of foreground detection

We have tested our method in real Lidar sequences concerning both video surveillance (*Courtyard*) and traffic monitoring (*Traffic*) scenarios (see Fig. 7). The data flows have been recorded by a Velodyne HDL 64E S2 camera, which operates with $R = 64$ vertically aligned beams. The *Courtyard* sequence contains 2500 frames with four people walking in a $25m^2$ area in 1-5m distances from the Lidar, with crossing trajectories. The rotation speed was set to 20Hz. In the background, heavy motion of the vegetations make the accurate classification challenging. The *Traffic* sequence was recorded with 5Hz from the top of a car waiting at a traffic light in a crowded crossroad. The adaptive background model was automatically built up within a few seconds, then 160 time frames were available for traffic flow analysis. We have compared our DMRF model to three reference solutions:

1. *Basic MoG*, introduced in Sec. 2.2, which is based on [5] with using on-line K-means parameter update [11].
2. *uniMRF*, introduced in Sec. 2.3, which partially adopts the uniform foreground model of [3] for range image segmentation in the DMRF framework.
3. *3D-MRF*, which implements a MRF model in 3D, similarly to [9]. We define here point neighborhoods in the original $\mathcal{L}^t$ clouds based on Euclidean distance, and use the background fitness values of (2) in the data model. The graph-cut algorithm [12] is adopted again for MRF energy optimization.

Qualitative results on two sample frames are shown in Fig. 7. For Ground Truth (GT) generation, we have developed a 3D point cloud annotation tool, which enables labeling the scene regions manually as foreground or background. Next, we manually annotated 700 relevant frames of the *Courtyard* and 50 frames of the *Traffic* sequence. For quantitative evaluation metric, we have chosen the point level F-rate of foreground detection [4], which can be calculated as the harmonic mean of precision and recall. We have also measured the processing speed in frames per seconds (fps). The numerical performance analysis is given in Table 1. The results confirm that the proposed model surpasses the *Basic MoG* and *uniMRF* techniques in F-rate for both scenes, and the differences are especially notable at the *Courtyard*. Compared to the *3D-MRF* method, our model provides similar detection accuracy, but the *proposed DMRF* method is significantly quicker. Observe that differently



(a) Basic MoG, *Courtyard* sequence (b) Proposed DMRF, *Courtyard* sequence

(c) Basic MoG, *Traffic* sequence (d) Proposed DMRF, *Traffic* sequence

Figure 7: Point cloud classification result on sample frames with the *Basic MoG* and the proposed DMRF model: foreground points are displayed in blue (dark in gray print).

Aspect	Sequence	Seq. prop.	Bas. MoG	uniMRF	3D-MRF	**DMRF**
Detection rate (F-mes in %)	*Courtyard*	4 obj/fr.	55.7	81.0	88.1	95.1
	Traffic	20 obj/fr.	70.4	68.3	76.2	74.0
Proc. speed (fr per sec)	*Courtyard*	65K pts/fr.	120 fps	18 fps	7 fps	16 fps
	Traffic	260K pts/fr.	120 fps	18 fps	2 fps	16 fps

Table 1: Numerical evaluation on the *Courtyard* and *Traffic* sequences: detection accuracy (F-rate in %) and processing speed (fps, measured in a desktop computer)

from 3D-MRF, our range image based technique is less influenced by the size of the point cloud. In the *Traffic* sequence, which contains around 260000 points within a time frame, we measured 2fps processing speed with 3D-MRF and 16fps with the proposed DMRF model.

3 Pedestrian detection and multi-target tracking

In this section, we introduce the pedestrian tracking module of the system. The input of this step is a Velodyne point cloud sequence, where each point is marked with a segmentation label of foreground or background, while the output consists of clusters of foreground regions

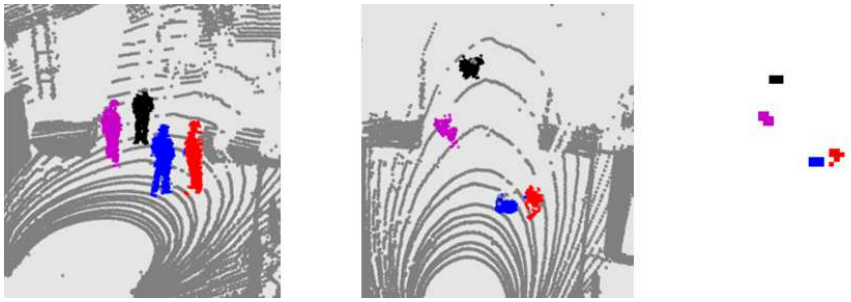


Figure 8: Pedestrian separation. Left: side view of the segmented scene, centered: top view, right: projected blobs in the image plane

so that the points corresponding to the same person receive the same label over the sequence. We also generate a 2D foot point trajectory of each pedestrian, which will be directly used by the 4D scene reconstruction module (see Fig. 9 and Sec. 6).

3.1 Separation of moving pedestrians

In the starting step of the module the point cloud regions classified as foreground are clustered to obtain separate blobs for each moving person. First, we fit a regular lattice onto the ground plane and foreground regions are projected onto this lattice. Then in the image plane apply morphological filters to obtain spatially connected blobs for the different people. Thereafter we extract appropriately sized connected components, which satisfy area constraints determined by lower and higher thresholds. The latter step is demonstrated in Fig. 8. Each extracted blob center is considered as a pedestrian foot-position candidate in the 2D ground plane. Note that in this way, connecting people may be merged into the same blob, or blobs of partially occluded pedestrians be missing or broken into several parts. Instead of proposing various heuristic rules to eliminate these artifacts at the level of the individual time frames, we developed a robust multi-tracking module which efficiently handles the problems at sequence level.

3.2 Pedestrian tracking

The multi target tracking problem can be formulated in the following way. On each current frame the k detected target candidates (in our case 2D points in ground plane's coordinate system) have to be assigned to n tracked objects from the previous frames. While processing the current frame, we have to simultaneously handle the following cases:

- If a given detected point is the continuation of an existing track, we have to find the correct assignment.

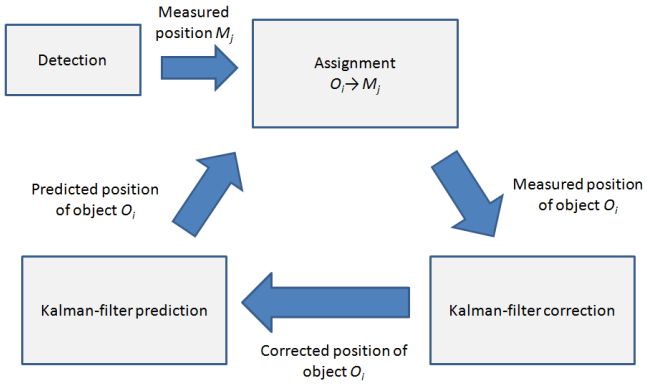


Figure 9: Workflow of the tracking algorithm

- A given detected point may be a false alarm.
- A given detected point in the current frame may be the starting point of a new track.
- An existing object track may be finished in the previous frame.
- The detector may ignore some of the targets in the current frame, which should not result in broken or re-started object tracks. Temporal discontinuities of the tracks must be filled later with estimated position values.

The workflow of the proposed algorithm can be followed in Fig. 9. Three steps are iterated for each frame: (A) Assignment, (B) Kalman filter correction and (C) Kalman prediction.

3.2.1 Assignment

The assignment step is the key part of the algorithm. First, the coordinates of the measured positions are normalized to fit into the $[0, 1]$ domain for each dimension. We define the distance of the targets as the Euclidean in the normalized data cube. Let us denote by O_j $(j = 1, \ldots, k)$ the normalized target positions (i.e. Measurements) detected in the current frame, and by M_i $(i = 1, \ldots, n)$ the predicted position of the object which corresponds to the ith track maintained by the detector. A distance matrix D is calculated encapsulating the distances of O_i and M_j for different i and j values:

$$D_{ij} = d(O_i, M_j) \quad \text{for } i \leq n, j \leq k$$

Here the D_{ij} matrix element expresses how good the jth measurement fits the ith maintained track.

Based on the D distance matrix, the trajectories and the current measurements are assigned with the Hungarian Graph Matching algorithm [Kuhn55].

We must notice that the Hungarian algorithm always expects a squared distance matrix, which is automatically fulfilled if the input of the matcher consists of the same number of tracks and measurements. However, this condition does usually not hold. For this reason, if $n > k$ we temporarily generate $n - k$fictional measurements which have maximum distance from all trajectories:

$$d(O_i, M_j) = 2 \quad \text{for } i \leq n, n \geq j > k$$

Here 2 is chosen as default distance value, because it is greater than the distance of any point pairs within the normalized data cube.

On the other hand, if $k > n$, we generate $k - n$ fictional tracks to complete the D matrix:

$$d(O_i, M_j) = 2 \quad \text{for } k \geq i > n, j \leq k$$

The output of the Hungarian matcher is a unique assignment $j \rightarrow A(j)$ between the measurements and the trajectories, where j (resp. $A(j)$) index may also correspond to real or fictive measurements (resp. trajectory).

Let t_{dist} be a distance threshold. The obtained $j \rightarrow A(j)$ assignment is interpreted in the following way:

- if $A(j) \leq n$, $j \leq k$ and $d(O_{A(j)}, M_j) < t_{\text{dist}}$: **DO** measurement j is `matched` to trajectory $A(j)$
- otherwise
  - if $A(j) \leq n$, $j \leq k$, but $d(O_{A(j)}, M_j) \geq t_{\text{dist}}$: both the jth measurement and the $A(j)$th trajectory are marked as `unmached`.
  - if $A(j) \leq n$ and $n \geq j > k$ the $A(j)$th trajectory is marked as `unmached`.
  - if $k \geq A(j) > n$ and $j \leq k$ the jth measurement is marked as `unmached`.

If the M_j measurement is `matched` to the O_i trajectory point, we can expect that M_j corresponds to the new position of the ith target: therefore it will be used for trajectory update in the following steps.

`Unmached` measurements are potential initial points of new trajectories, or they are caused by false positive targets of the detector. We cannot distinguish these two cases at the current frame. Thus for each unmatched measurement value, we start a new potential object track, which is investigated during the upcoming iterations. We expect that by the end of the tracking process, false target candidates will result in short or stationary tracks, which can be eliminated in the post processing phase. If a trajectory is `unmached`, it may be caused by two reasons: (i) either it has already ended, (ii) or the detector produced a mis-detection in the current frame. Therefore, `unmached` tracks are not closed immediately, but they are marked as INACTIVE. If a trajectory is inactive for longer than a time threshold t_{time}, it is marked as DELETED, and excluded from the further investigations during the tracking process.

3.2.2 Kalman correction

Each object trajectory consists of a point sequence over the consecutive frames. Since with the applied sampling rate, the target motion can be considered smooth, we can make predictions from the trajectory for the next expected track point. Moreover, since the measurement may be distorted by the detector noise, the true object position should be estimated by both considering the actual detector output and the previous trajectory part of the target.

For this reason, we maintain a Kalman filter for each track, which is updated in each frame with the assigned measurements values. For INACTIVE (but still not DELETED) tracks - which do not have actual measurements - the Kalman filter of the trajectory is updated with the latest prediction value of the current position. In both cases, the next point of the trajectory will be the corrected state of the filter. Note that the estimation of the Kalman filter becomes only reliable after a couple of frames. Therefore, we use a t_{Kalman} frame number threshold, and if the trajectory is shorter than t_{Kalman}, we directly use the matched measurement value as the next track point, instead of the corrected state. In this way, we must expect that the track will be noisy (less smooth) in its initial phase, but the probability of completely loosing the trajectory is significantly decreased.

The trajectory points of INACTIVE targets are collected first in a temporary queue. If the target is re-activated later, this queue is appended to the real trajectory, which is continued with the latest state correction. If the target is DELETED (inactive for more than ttime frames), the temporary trajectory points are removed.

3.2.3 Kalman prediction

The final step of the trajectory update is to make the prediction for the next point of each track (marked with O_i above), which can be used for measurement assignment in the next frame. Here again, we only apply the Kalman prediction after t_{Kalman} frames. In the initial part of the track, we use the last known trajectory point as the prediction for the next position.

3.2.4 Post processing and filtering

The above tracker may assign object trajectories for point sequences resulted by measurement noise. These false tracks should be eliminated after the iterative tracking process has finished. We have observed that the false tracks are either short, or contain targets with nearly constant position (our assumption is that our targets are in motion). Therefore, in the post processing phase, we used two constraints for the trajectories:

- The length of the trajectory should be higher than t_{length}
- The average variance of the point coordinates over the track should be higher than t_{variance}

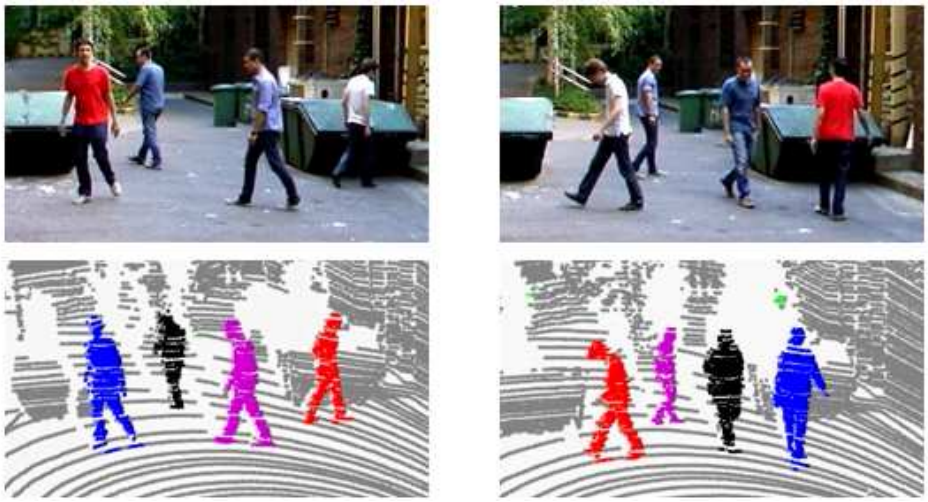


Figure 10: Demonstration of successful pedestrian tracking in Lidar sequences. Point cloud regions corresponding to the same person are displayed with permanent color (video frame can be used for verification)

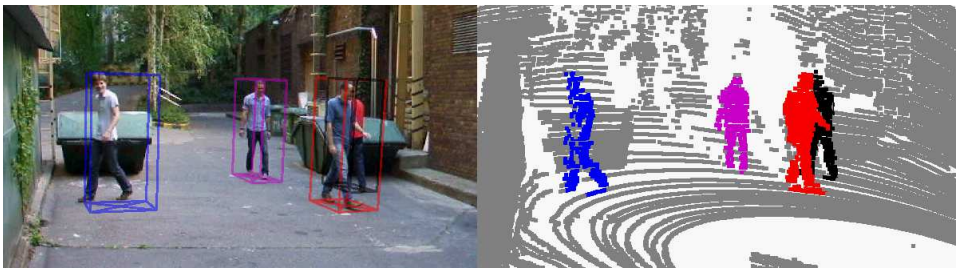


Figure 11: Bounding boxes of pedestrians - obtained by Lidar based tracking - projected to the image plane of the video camera

Trajectories which fail any of the above constraints are removed, and only the remaining ones are used as output of the tracker.

3.3 Label backprojection and camera registration

The tracker module provides a set of pedestrian trajectories, which are 2D foot center point sequences in the ground plane. To determine the points corresponding to each pedestrian

in a selected frame, the connected foot blobs around a given trajectory point should be vertically backprojected to the 3D point cloud. The result of tracking is demonstrated in Fig. 10, which shows two segmented point cloud frames from a measurement sequence in a courtyard. We also show video frames taken in parallel as reference, which confirm that during the tracking the Lidar clouds of the same pedestrian obtain the same color.

We note that a camera calibrated to the point cloud may also provide efficient features for long term person identification and scene analysis. In the future part of this project we aim to investigate the options of Lidar and camera fusion. As a first step, we projected the bounding boxes of the people - obtained solely from the Lidar data - to the camera plane, as shown in Fig. 11.

4 Background scene reconstruction

In this section, we describe the static environment reconstruction method. If we subtract the foreground points (mostly people) from the *Courtyard* sequence, the result is a dense point cloud, which represents the ground, walls, trees, or other background objects. The ground points were automatically detected by the RANSAC [13] algorithm, fitting an optimal plane onto the point cloud. If we are able to extract the points, which corresponds to the ground, we can calculate the average height of the ground, which will be used in the next step. The rest of the points is projected vertically to the calculated ground level, then a Hough transform [14] is applied to find lines, which probably correspond to wall points in the cloud. From these points we create a polygon mesh with the Ball-pivoting algorithm. The automatic reconstruction of vegetation, other smaller objects, and automatic texturing is part of our future goals, currently we do this manually. Environment reconstruction flowchart is displayed in figure 12, example frames of the process are in figure 13.

5 4D walking pedestrian model generation

A realistic solution for people reconstruction cannot be obtained based on the Velodyne point cloud only, because it is too sparse and it gives 2.5D information. We have decided to place models of walking actors into the virtual environment, which had been captured earlier, and follow the trajectories based on real measurement results. The realistic textured models were created in the 4D Reconstruction Studio being developed at MTA SZTAKI by the Geometrical Modeling and Computer Vision Laboratory [15] [16].

A 4D studio is an advanced, intelligent sensory environment operated by sophisticated programming tools. This environment can be used for computer vision research as well as for technological development in a variety of applications. To our best knowledge, the 4D Reconstruction Studio at MTA SZTAKI is a pioneering project in Central and Eastern Europe. The main motivation for building the Studio was the desire to bring advanced knowledge and technology to this region in order to facilitate testing new ideas and developing new methods, tools and applications.

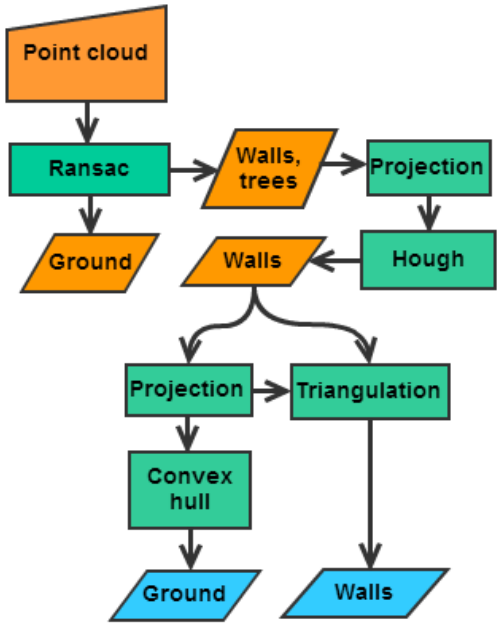


Figure 12: Environment reconstruction flowchart.

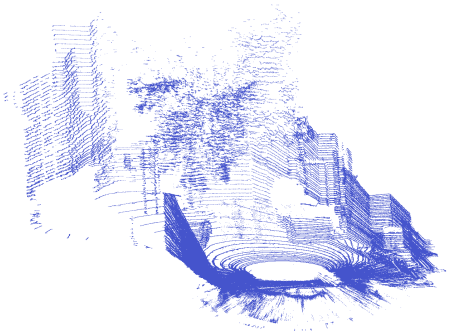
In this section, we discuss the main hardware and software elements of the 4D Reconstruction Studio, and the process to obtain realistic walking pedestrian models.

5.1 Hardware of the 4D Reconstruction Studio

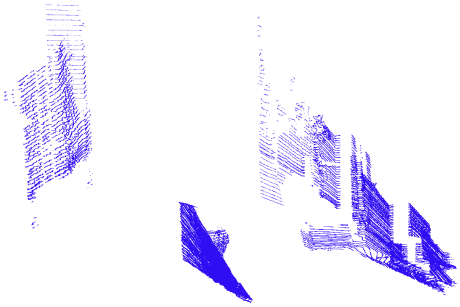
The 4D Reconstruction Studio is a "green box": green curtains and carpet provide homogeneous background. The massive, firm steel frame is a cylinder with dodecagon base. The size of the frame is limited by the size of the room. The diameter is around five meters; originally, a seven-meter studio was planned. The frame carries 12 video cameras placed uniformly around the scene and one additional camera on the top in the middle (Fig. 14).

The cameras are equipped with wide-angle lenses to cope with relatively close views; this necessitates careful calibration against radial distortion. The resolution of the cameras is 1624×1236 pixels; they operate at 25 fps and use GigE (Gigabit Ethernet).

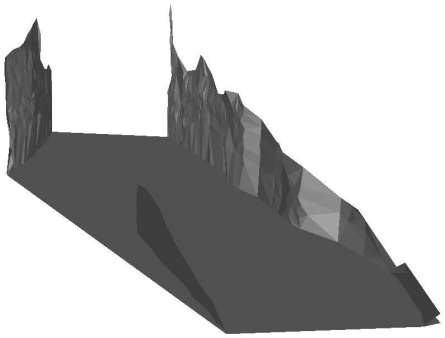
Special, innovative lighting has been designed for the Studio to achieve better illumination. Apart from the standard diffuse light sources, we use light-emitting diodes (LEDs) placed around each camera, as illustrated in Fig. 15. The LEDs can be turned on and off with high frequency. A micro-controller synchronizes the cameras and the LEDs: when a camera takes a picture, the LEDS opposite to the camera are turned off. This solution



(a) Segmented background points.



(b) Points classified as walls.



(c) Walls and ground polygon mesh.

(d) Fully reconstructed environment.

Figure 13: Example frames on the segmentation steps.

improves illumination and allows for more flexible configuration of the cameras. The Studio uses seven conventional PCs; each of them but one handles two cameras.

5.2 Software modules of the Studio

The Studio has two main software blocks: the image acquisition software for video recording and the 3D reconstruction software for creation of dynamic 3D models. The software system includes elements from the OpenCV [17]; otherwise, the entire system has been developed at SZTAKI.

The image acquisition software configures and calibrates the cameras and selects a subset of the cameras for video recording. The easy-to-use, robust and efficient Z. Zhang's method [18] implemented based on OpenCV routines is used for intrinsic and extrinsic camera calibration and calculation of the parameters of radial distortion. During calibration, the

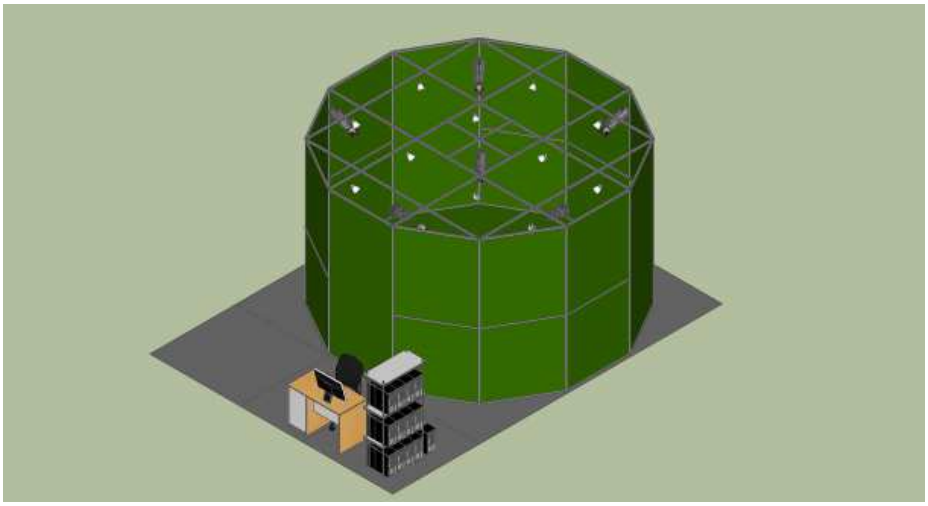


Figure 14: A sketch of the 4D Studio at SZTAKI.

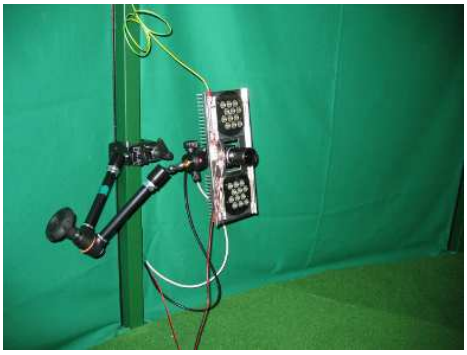


Figure 15: Adjustable platform with a video camera and LEDs mounted on the frame.

operator repeatedly shows a flat chessboard pattern to the cameras. The complete procedure takes a few minutes.

The main steps of the 3D reconstruction process are as follows:

1. Extract *color images* from the raw data captured.
2. Segment each color image to *foreground and background*.
3. Create *volumetric model* using the Visual Hull algorithm.

Figure 16: Sample input images of the Studio.

4. Create *triangulated mesh* from the volumetric model using the Marching Cube algorithm.

5. Add *texture* to the triangulated mesh.

Segmenting input images into foreground and background is a critical step. Our image segmentation procedure is a novel method developed at SZTAKI for this project. The method assumes that the background is larger than the object, which is normally the case since the object needs room to move in the scene. The principles of segmentation are listed below.

- Acquire a *reference background image* in the absence of any object.
- Convert the input RGB image to the *spherical color representation*.
- Calculate the absolute *difference* between the input image and the reference background image.
- In the difference image, select object pixels as outliers using robust *outlier detection*.
- Clean the resulting object image using *morphologic operations* such as erosion and dilation by disc.

Fig. 16 shows sample input images acquired in our Studio. The binary segmented images are demonstrated in Fig. 17.

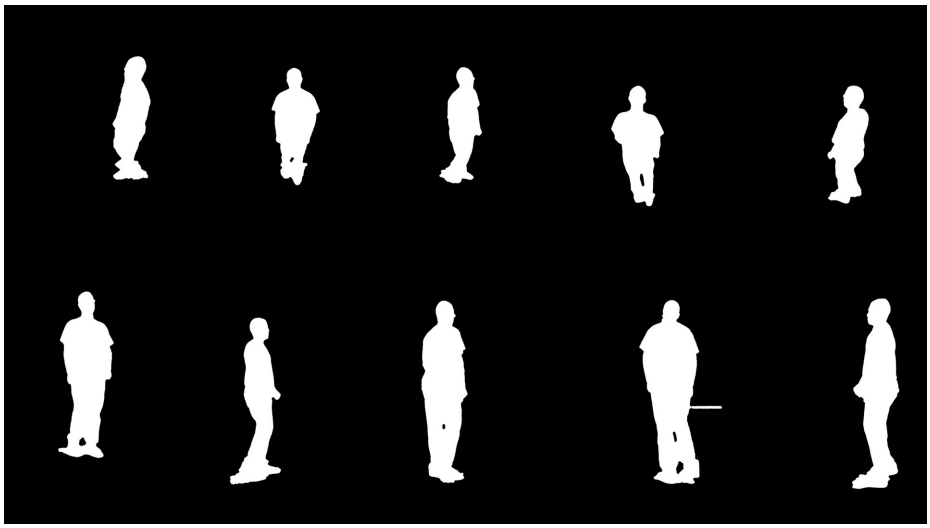


Figure 17: Segmentation of the images shown in Fig 16.

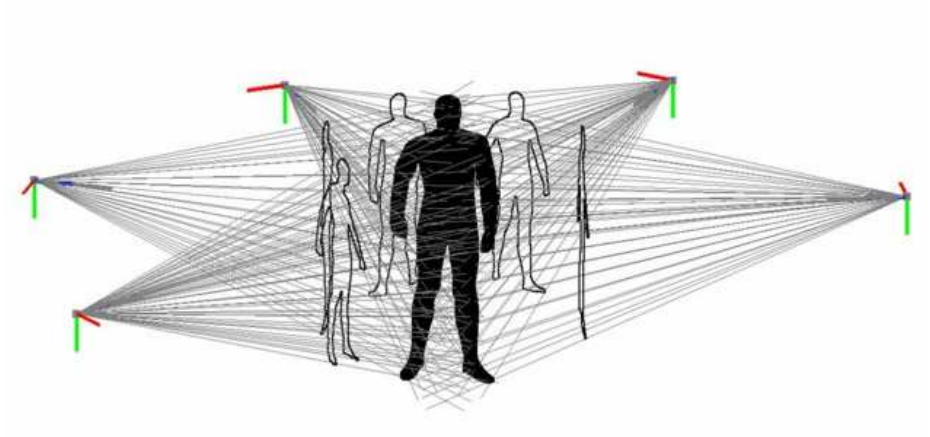


Figure 18: Extracting the visual hull from silhouettes.

A shape-from-silhouettes technique is used to obtain a volumetric model of the dynamic shape [19]. Object silhouettes obtained by the cameras are back-projected to 3D space as the generalized cones whose intersection gives the visual hull, a bounding geometry of the actual 3D object. Using more cameras results in a finer volumetric model, but some concave details may be lost anyway. The process of obtaining the visual hull from silhouettes is illustrated in Fig. 18.

Currently, video frames are processed separately, i.e., the dynamic model obtained is a sequence of separate, instantaneous shapes.

The volumetric visual hull is transformed into a surface mesh which is textured by selecting the most appropriate view for each unit of the mesh based on visibility, or by combining several views. The mesh is obtained from the hull using the standard Marching Cubes algorithm [20]. The algorithm for texturing the triangulated surface [21] calculates a measure of visibility for each triangle and each camera. The triangle should be visible from the camera and its normal vector should point towards camera. Then, a cost function is formed with visibility and regularization terms to balance between visibility of triangle and smoothness of texture. The regularization term reduces sharp texture edges between adjacent triangles. The cost function is minimized using graph cuts.

Fig. 5.2 shows examples of textureless and textured models. Figs. 20-21 illustrate our system's capability to create mixed reality.

6 Dynamic 4D scene reconstruction

The last step of the workflow is the integration of the different components. The walking pedestrian models are placed into the reconstructed background scene, and their foot-center points follow the trajectories extracted from the Lidar point cloud sequence. In the current stage, we used the assumptions that the people are walking forward along the trajectory, and the orientation from top view is calculated from the gradient of the 2D track. A sample frame from the reconstruction results can be seen in figure 23. We show six frames from a video about the reconstructed dynamic scene using a simulated moving camera in Fig. 24. Finally consecutive frames of the processed pointclouid and the reconstructed scenario are displayed in Fig. 25.

Figure 19: Examples of textureless and textured models.

References

[1] I. Schiller and R. Koch. Improved video segmentation by adaptive combination of depth keying and Mixture-of-Gaussians. In *Proc. Scandinavian Conference on Image Analysis, Ystad, Sweden*, volume 6688 of *LNCS*, pages 59–68, 2011.

[2] B. Langmann, S.E. Ghobadi, K. Hartmann, and O. Loffeld. Multi-modal background subtraction using gaussian mixture models. In *ISPRS Symposium on Photogrammetric Computer Vision and Image Analysis*, pages 61–66, 2010.

[3] Y. Wang, K-F Loe, and J-K Wu. A dynamic conditional random field model for foreground and shadow segmentation. *IEEE Transactions on Pattern Analysis and Machine*

Figure 20: Obtained 3D models can be multiplied.

Intelligence, 28(2):279 –289, 2006.

[4] C. Benedek and T. Szirányi. Bayesian foreground and shadow detection in uncertain frame rate surveillance videos. *IEEE Transactions on Image Processing*, 17(4):608 – 621, 2008.

[5] R. Kaestner, N. Engelhard, R. Triebel, and R.Siegwart. A Bayesian approach to learning 3D representations of dynamic environments. In *Proc. International Symposium on Experimental Robotics (ISER)*, Berlin, 2010. Springer Press.

[6] B. Kalyan, K. W. Lee, W. S. Wijesoma, D. Moratuwage, and N. M. Patrikalakis. A random finite set based detection and tracking using 3D LIDAR in dynamic environments. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2288–2292, Istanbul, Turkey, 2010. IEEE.

[7] N. Muhammad and S. Lacroix. Calibration of a rotating multi-beam Lidar. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 5648–5653, Taipei, Taiwan, 2010. IEEE.

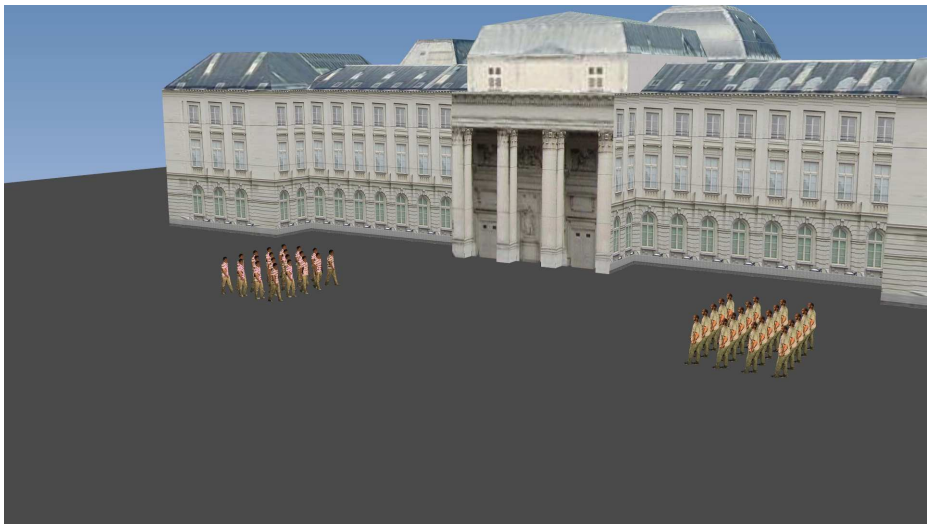


Figure 21: Obtained 3D models can be combined with arbitrary virtual environment.

Figure 22: Different gait phases of a 4D studio object

[8] L. Spinello, M. Luber, and K.O. Arras. Tracking people in 3D using a bottom-up top-down detector. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1304–1310, Shanghai, China, 2011.

[9] F. Lafarge and C. Mallet. Creating large-scale city models from 3D-point clouds: A robust approach with hybrid representation. *Int. J. of Computer Vision*, 2012.

[10] C. Benedek, D. Molnár, and T Szirányi. A dynamic MRF model for foreground detection on range data sequences of rotating multi-beam lidar. In *International Workshop on Depth Image Analysis, LNCS*, Tsukuba City, Japan, 2012.

[11] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:747–757, 2000.

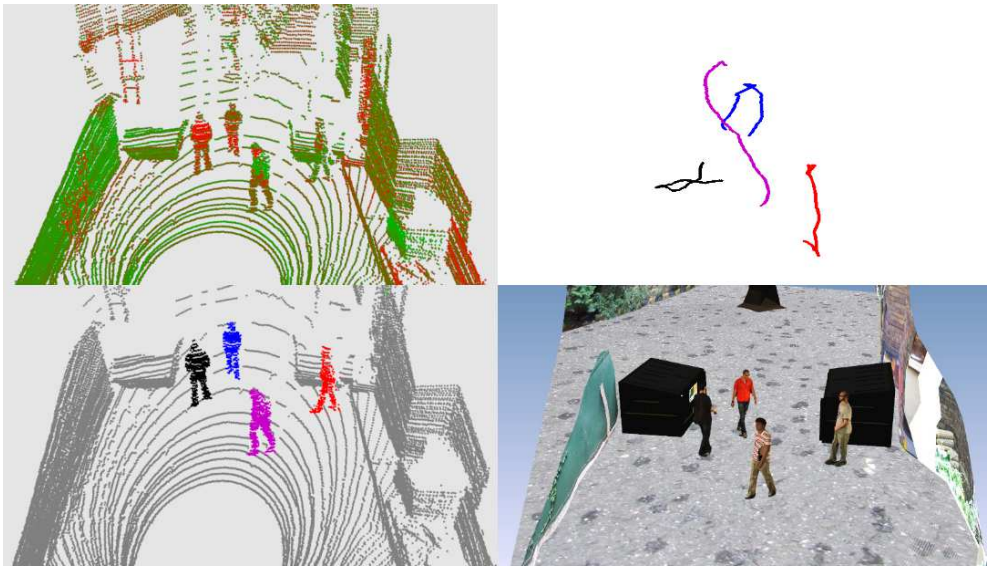


Figure 23: Object tracking and reconstruction results. Upper left: raw point cloud; lower left: segmented and separated objects; upper right: trajectories from upper view; lower right: reconstructed environment, the studio objects placed to the original positions.

Figure 24: Sample frames from a video about the reconstructed dynamic scene using a simulated moving camera

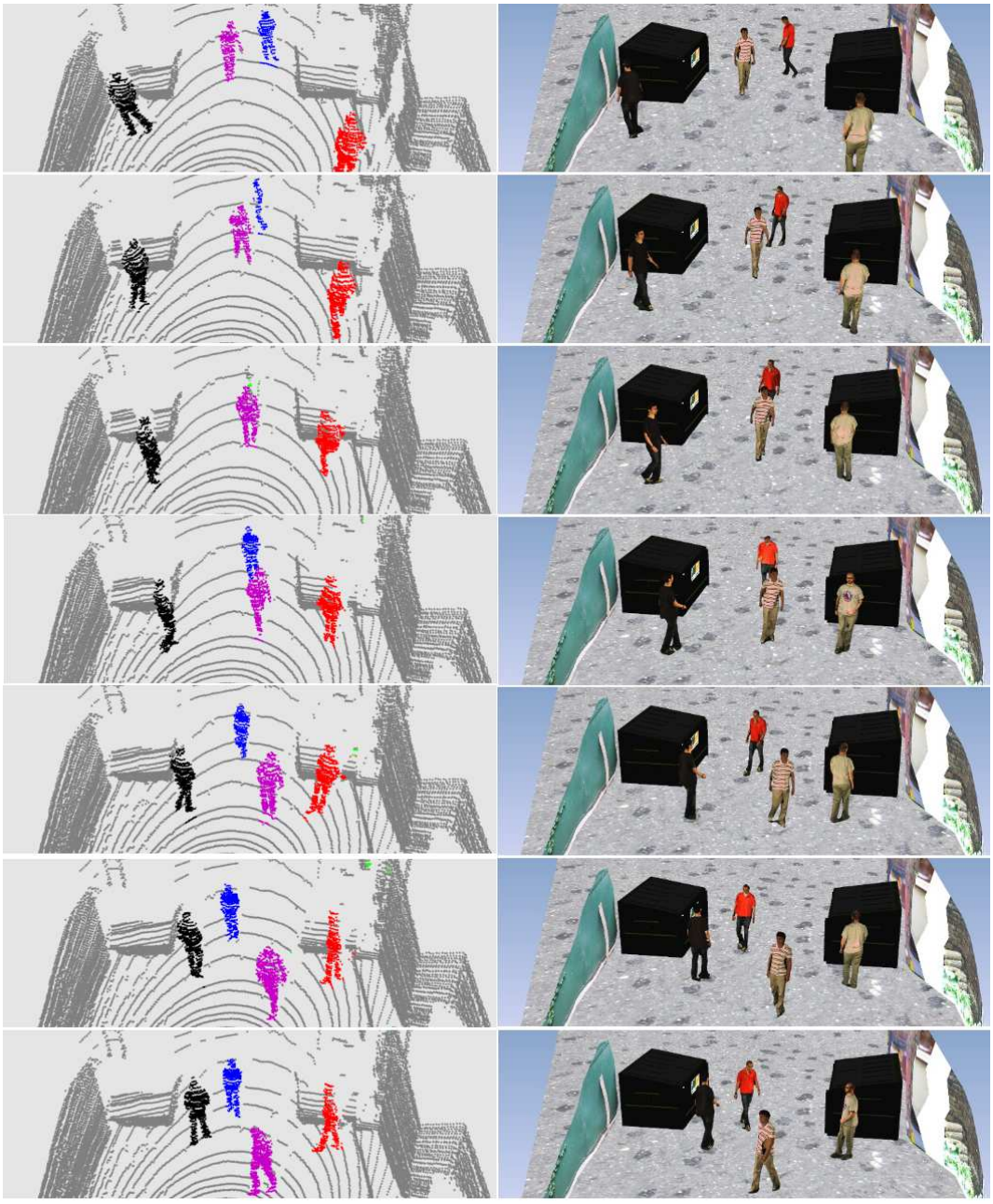


Figure 25: Consecutive frames of the processed pointclouid and the reconstructed scenario

[12] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[13] Bolles R.C. Fischler, M.A. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Comm. of the ACM*, volume 24, pages 381–395, 1981.

[14] Hart P.E. Duda, R.O. Use of the hough transformation to detect lines and curves in pictures. In *Comm. of the ACM*, volume 15, pages 11–15, 1972.

[15] J. Hapák, Z. Jankó, and D. Chetverikov. Real-time 4d reconstruction of human motion. In *Proc. 7th International Conference on Articulated Motion and Deformable Objects (AMDO 2012)*, volume 7378, pages 250–259, 2012.

[16] Z. Jankó, D. Chetverikov, and J. Hapák. 4d reconstruction studio: Creating dynamic 3d models of moving actors. In *Hungarian Computer Graphics and Geometry Conference*, 2012.

[17] OpenCV. Open Computer Vision Library. `sourceforge.net/projects/opencvlibrary/`, 2012.

[18] Z. Zhang. A flexible new technique for camera calibration. 22:1330–1334, 2000.

[19] A. Laurentini. The visual hull concept for silhouette-based image understanding. 16:150–162, 1994.

[20] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proc. ACM SIGGRAPH*, volume 21, pages 163–169, 1987.

[21] Z. Janko and J.-P. Pons. Spatio-temporal image-based texture atlases for dynamic 3-D models. In *Proc. ICCV Workshop 3DIM'09*, pages 1646–1653, 2009.

Departments of the institute
http://www.sztaki.hu/departments/

3D Internet-based Control and Communications Laboratory, Cellular Sensory and Optical Wave Computing Laboratory
Computer Integrated Manufacturing Laboratory, Department of Distributed Systems, ELearning Department
Distributed Events Analysis Research Laboratory, Geometric Modelling and Computer Vision Laboratory
Informatics Laboratory, Internet Technologies and Applications Department, Laboratory of Parallel and Distributed Systems
Network Security Department, Research Laboratory on Engineering & Management Intelligence, Systems and Control Lab

MTA SZTAKI Institute for Computer Science and Control
Hungarian Academy of Sciences
Kende utca 13-17 H-1111 Budapest (Hungary)
http://www.sztaki.hu