Háromdimenziós primitívek alkalmazása helymeghatározásra több nézetből tetszőleges talajon

Using 3D Primitives for Multiview-detection on Arbitrary Ground

Ákos Kiss[1,2] and Szirányi Tamás[1]

[1] Distributed Events Analysis Research Group, MTA SZTAKI, Budapest, Hungary
sziranyi@sztaki.mta.hu
[2] Dept. of Control Engineering and Information Technology - Budapest University of Technology and Economics, Budapest, Hungary
akos.kiss@sztaki.mta.hu

Abstract. Olyan módszert mutatunk, amely alkalmas emberek pontos, háromdimenziós helyének meghatározására több nézet alapján, függetlenül a talaj geometriájától. Két lényegében azonos, de nagyon eltérő tulajdonságokkal rendelkező eljárást ismertetünk. Egyik esetben a lábfejhez tartozó képpontokat visszavetítjük egyenesekre, és ezek metszéspontjait keressük. Másik esetben a lábfejeket ellipszisekkel fedjük, ezeket visszavetítjük kúpokra, és ezek metszete jelöli ki a lábfejek pozícióját. A kúpok bonyolult metszetének számítása helyett közelítést alkalmazunk, ezzel nagy mértékben gyorsítva az eljárást. Azt találtuk, hogy a lábfej pozíciója nagy pontossággal meghatározható, és a talaj magasságtérképe nagy pontossággal előállítható. Eljárásunkat összehasonlítottuk másokéval - némiképp különböző tesztelési eljárással -, azokkal összemérhető eredményt kaptunk, ám a mi módszerünk esetén nem követelmény a sík talaj. Bemutatjuk a módszer működését nem sík talaj esetén is. Egy szálon futó implementációnk alkalmas valós idejű futásra, de az algoritmus párhuzamosítható.

1 Introduction

Single camera detecting and tracking relies on some kind of descriptors, like color, shape, and texture. However, extraordinary and occluding objects are hard to detect. Multiple cameras are often used to overcome these limitations.

In these cases, consistency of object pixels among views will signal objects in 3D space. There are many methods for finding object pixels. Using stereo cameras, the disparity map can highlight foreground, or using wide-baseline stereo imaging, image-wise foreground detection is carried out.

The resulting set of object pixels can be further segmented, and consistency check might be reduced to likely correspondent segments. This might be done using color descriptors [1, 2], however color calibration [3] is necessary due to different sensor properties or illumination conditions.

Foreground masks can be projected to a ground plane, and overlapping pixels mark consistent regions [4]. Robustness can be improved using multiple parallel planes [5]. Furthermore, looking for certain patterns in the projection plane increases reliability [6].

In many works authors assume known homography between views and ground plane to carry out projection. In [7] homography parameters are estimated from co-motion statistics from multimodal input videos, eliminating the need of human supervision.

Projection of whole foreground masks is computationally expensive, but filtering pixels can reduce complexity. In some works, points associated with feet are searched, reducing foreground masks from arbitrary blobs to points and lines [8, 4].

Another gain of filtering foreground points is that - depending on the geometry - feet are less occluded than whole bodies, eliminating a great source of errors. Reducing occlusion is especially important in dense crowds, for example using top-view cameras [9].

2 Overview

Our goal was to design an algorithm for detecting people in a multiview environment with possibly many views where the geometry of the ground is arbitrary, which problem is not addressed in the field. Moreover we aimed at reaching real-time running to make it available for surveillance systems.

We utilized the fact that foreground pixels of a view correspond to lines in 3D scene space, and lines intersect in scene space at points inside the object. Finding such intersections mark the 3D position of objects [10]. The number of line pairs - or tests to be made - n_p is proportional to

$$n_p = (s_x s_y r_{fg})^2 \binom{n}{2} \tag{1}$$

where s_x, s_y is the resolution of the camera image, r_{fg} is the ratio of foreground pixels in image and n is the number of views.

Number of foreground pixels is great even for low resolution surveillance videos of people. Therefore we filtered foreground mask to pixels relevant for detecting people - candidate pixels of feet and bodies -, decreasing r_{fg}. This filtering is described in Section 3.

A foot covers different number of pixels with both change in resolution and distance from camera. As a result,

- number of line pairs to match is proportional to 4th power of resolution, resulting in great computational cost,
- different number of matches occur inside a foot in different locations, interpreting such data was hard.

Therefore we tried to cluster candidate pixels and form 3D primitives, cones from these clusters. This way number of cones does not depend on resolution, only number of objects present. Our approach has several advantages in means of both precision and speed:

– for determining line or cone parameters, undistortion may be carried out with few computations leading to accurate parameters,
– ground doesn't have to be flat (unlike using homographies in [6, 5, 11, 4, 12]),
– we can compute synergy map without height range presumption like in [6, 5, 11]

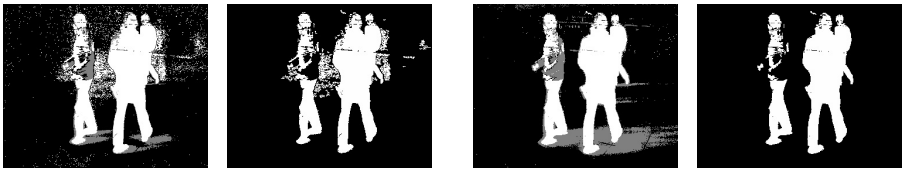
On the downside:

– our algorithm may fail on incorrect foreground mask. Occasionally, candidate pixel clusters don't overlap with feet, corrupting detection.
– precise calibration is required for reliable estimation of line and cone parameters (both intrinsic and extrinsic parameters of the camera). Errors increase with distance from camera, a small error in parameters can affect intersection.

We show details of intersection searching for both ways: line intersections in Section 4, and cone intersections in Section 5. Detection based on intersections is described in Section 6. We measured low precision/recall values using simply the detector, in Section 7 we show methods to improve the algorithm, finally resulting in precision/recall values comparable to SOA methods. In Section 9 we summarize our results.

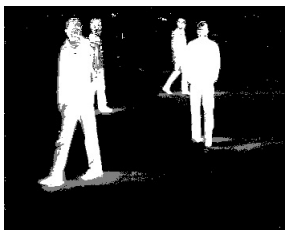
(a) (b) (c)

Fig. 1. Shadows may corrupt foreground detection even in carefully chosen color space. Foreground mask (b) for an input frame (a) is shown, as well as the resulting feet points and ellipses (c) (ellipses are visualized with lozenges). A number of false positives are introduced due to corrupted foreground mask.



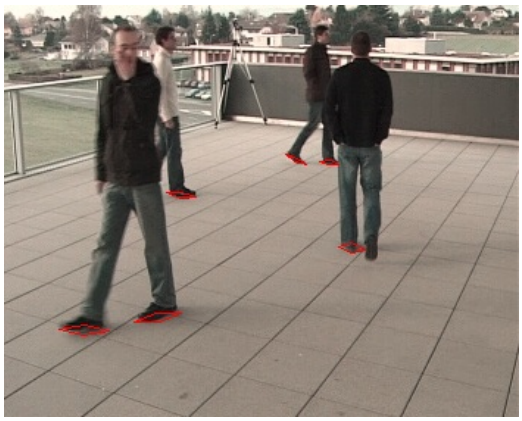
(a) (b)

Fig. 2. Foreground mask without (a) and with (b) white balance compensation (sub figures show foreground mask with shadow detection (left), and filtered mask (right)).



(a)

(b)



(c)

Fig. 3. Steps of extracting ellipses covering feet: (a) foreground detection, (b) finding candidate pixel set, (c) forming ellipses covering these pixel sets (ellipses are visualized with lozenges).

3 Preprocessing

We used a modified version of the foreground detector described in [13]: we added white balance compensation to the model.

Foreground detection was corrupted by shadows and reflections on the ground. We tried different color spaces to overcome this issue, and chose to use XYZ color space. It lead to best overall performance of our algorithm, however in certain situations foreground mask was still corrupted by shadows, as we can see in Fig 1.

We found that camera's auto white balance function leads to incorrect foreground mask. This function aims at adopting to changes in lightning, but it is also affected by large objects entering or exiting the image, thereby changing white balance even if lighting conditions remain the same.

To neglect these effects, we extended the background model with white balance adaptation. By computing and removing consistent changes in background pixel values,

we managed to compensate changes in white balance. This resulted in a more reliable foreground detection (see Fig 2)

3.1 Filtering foreground pixels

We assumed that cameras are in upright position so that pixels of feet are bottom pixels of vertical lines in the foreground mask. We consider these pixels as candidates for feet, we call them candidate pixels. A sample of extracted candidate pixels can be seen in Fig 3(b).

Note that using calibration information, images are easily rotated or search in a different upright direction may be carried out.

For line matching, these candidate pixels are used as described in Section 4. For cone matching, we cluster this set of candidate pixels and model every cluster with an ellipse, which can be back-projected into cone in scene space. This operation is detailed in Section 5.

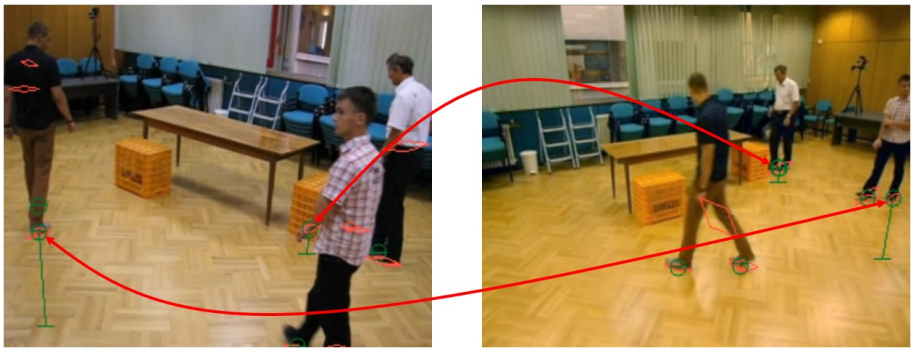


Fig. 4. Candidate pixels can appear on arms or on foreground artifacts, and also cones corresponding to different feet can intersect.

4 Line Correspondences

4.1 Back-projecting pixels

A line l is represented by a point p and a unit length directional vector v, noted by $l = (p, v)$. A straightforward way to back-project candidate pixel f in image i is to use the focal point of the camera p_i as the point and compute directional vector from undistorted camera coordinates u using the rotation matrix R_i. We get:

$$(p, v) = (p_i, \frac{R_i^{-1} u}{||R_i^{-1} u||}) \tag{2}$$

Clearly, most computation is due to undistorting camera coordinates.

4.2 Line intersections

Back-projected lines will practically never intersect. We are only able to compute the distance between two lines $l_1 = (p_1, v_1)$ and $l_2 = (p_2, v_2)$:

$$d(l_1, l_2) = \min_{s,t} ||(p_1 + sv_1) - (p_2 + tv_2)|| \quad (s,t \in \mathbb{R}) \tag{3}$$

The result can be derived by least-square approximation of a linear equation (using pseudo-inverse):

$$p_1 + sv_1 = p_2 + tv_2 \tag{4}$$

$$\begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{pmatrix} s \\ -t \end{pmatrix} = p_2 - p_1 = \Delta \tag{5}$$

If two lines are close to parallel, or $|v_1 v_2| = |cos(\alpha)| > |cos(\alpha_{min})|$, the result is not reliable. Otherwise, if distance between lines is smaller than a threshold value ($d(l_1, l_2) < d_{max}$), they are considered to intersect, or match. The position of the match is the point closest to both lines ($((p_1 + sv_1) + (p_2 + tv_2))/2$). d_{max} was set to $5cm$ in our experiments.

5 Cone Correspondences

5.1 Forming Ellipses

A cluster of candidate pixels is considered to be a sampling of an elliptical area. For an ellipse with major and minor radius a and b parallel to axis, we know:

$$\int x^2 = \frac{a^3 b\pi}{4} \qquad \int y^2 = \frac{ab^3\pi}{4} \qquad \int xy = 0 \qquad \int 1 = ab\pi \tag{6}$$

Rotating with α we get:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{7}$$

$$tg2\alpha = \frac{2\int x'y'}{\int y'^2 - \int x'^2} \qquad a^2 = \frac{4}{\int 1}(\int x'^2 + \int y'^2 + \frac{\int x'y'}{sc}) \qquad b = \frac{\int 1}{a\pi} \tag{8}$$

Where $s = sin\alpha$, $c = cos\alpha$. Now we can compute major and minor radius and rotation. We reject too small and upright ellipses. Minimal size can be adjusted for different setups and image resolution, in our experiments ellipses with diameter less than 7 were rejected, this ensured feet far from the camera are still detected. We defined maximal angle as 45°, foot direction does not depend on setup or resolution, so this threshold should work in any situation. Sample results can be seen in Fig. 3(c).

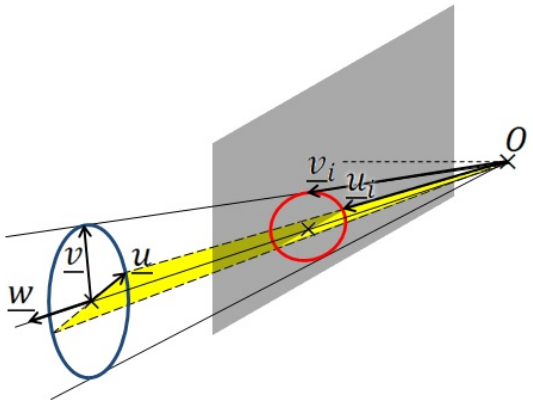


Fig. 5. Model of a cone. Vertex C is in the projection center O, $\underline{w}$ points toward center of ellipse in image plane. Bevel angles and axes directions ($\underline{u}$, $\underline{v}$) are computed from axes in image plane ($\underline{u}_i$, $\underline{v}_i$).

5.2 Forming Cones

We chose to describe a cone with a vertex C and three orthogonal vectors $\underline{u}$, $\underline{v}$ and $\underline{w}$, where $\underline{w}$ is the unit length direction vector of axis and $\underline{u}$, $\underline{v}$ are direction vectors of major and minor axes (Fig. 5). Bevel angles are determined by the length of $\underline{u}$, $\underline{v}$ vectors. Cone consists of points $\underline{p}$ where

$$\big((\underline{p}-C)\underline{u}\big)^2+\big((\underline{p}-C)\underline{v}\big)^2\le\big((\underline{p}-C)\underline{w}\big)^2,\quad(\underline{p}-C)\underline{w}\ge 0 \tag{9}$$

From extrinsic parameters of camera we know 3D position of points of image plane as well as the optical center O. Thus computing $\underline{w}$ is straightforward. We determined major and minor axes directions by vector products to ensure orthogonality of vectors:

$$\underline{v}=\underline{u}_i\times\underline{w}\qquad\qquad\underline{u}=\underline{w}\times\underline{v} \tag{10}$$

Finally, $\underline{u}$ and $\underline{v}$ are scaled according to major and minor bevel angles so that (9) stands.

5.3 Cone Matching

Intersection of cones is a complex body, especially when bevel-angles differ for major and minor axes. Our experiments showed that we don't need the exact intersection, an approximate solution is sufficient for searching cone matches. We simplified computation in two ways:

1. generally a foot covers a small area of the image, so cones will have small bevel-angles. Consequently in a relatively small space segment, they can be replaced by cylinders. We assume that intersection will take place near the point where the two

cone axes are closest, so we compute major and minor radius at this offset ($\overrightarrow{CP_{close}}$ in Fig. 6).

2. Computing exact intersection of cylinders is still a hard problem, and the intersection body is still complex. Therefore instead of computing intersection body, we tried to find an optimal point $\underline{p}$ in space, for which distance from cylinder axes is minimal - considering different major and minor radii.

After computing $|\overrightarrow{CP_{close}}|$ distance, we scale $\underline{u}$ and $\underline{v}$ vectors to match major and minor cylinder radii:

$$\underline{u'} = \underline{u}|\overrightarrow{CP_{close}}| \quad \underline{v'} = \underline{v}|\overrightarrow{CP_{close}}| \tag{11}$$

Now, to consider different major and minor axes, distance from cylinder axis can be computed as

$$\left((\underline{p}-C)\underline{u'}\right)^2 + \left((\underline{p}-C)\underline{v'}\right)^2 \tag{12}$$

For two cylinders, optimal solution would be a point, where distance is zero from both axes (lower indexes refer to different cylinders):

$$\begin{aligned} \left((\underline{p}-C_1)\underline{u'_1}\right)^2 + \left((\underline{p}-C_1)\underline{v'_1}\right)^2 = 0 \\ \left((\underline{p}-C_2)\underline{u'_2}\right)^2 + \left((\underline{p}-C_2)\underline{v'_2}\right)^2 = 0 \end{aligned} \tag{13}$$

Of course, axes will practically never intersect, we can only strive for minimizing some error function. (13) expresses sum of squared errors (SSE) of position errors. All error terms can be packed into the following linear equation:

$$\begin{bmatrix} \underline{u'_1}^T \\ \underline{v'_1}^T \\ \underline{u'_2}^T \\ \underline{v'_2}^T \end{bmatrix} \underline{p} = \begin{pmatrix} \underline{u'_1}C_1 \\ \underline{v'_1}C_1 \\ \underline{u'_2}C_2 \\ \underline{v'_2}C_2 \end{pmatrix} \qquad A\underline{p} = \underline{b} \tag{14}$$

optimizing with minimal SSE criteria is straightforward and practical in our case. It is straightforward, because it is a distance from axes considering different major and minor radii, and it is practical, because this optimization is easy to carry out using pseudo inverse.

$A^TA\underline{p} = A^T\underline{b}$ can be quickly solved using Gauss elimination. Note that terms in A^TA and $A^T\underline{b}$ can be precomputed per cone, speeding up matching.

Optimization results in a point $\underline{p}$, which is closest to cylinder axes considering different major and minor radii. If $\underline{p}$ is outside any cylinder, we conclude cones are not intersecting, otherwise, they intersect and $\underline{p}$ is the position of the intersection - which we call match. Using the distances of cylinder axes (the SSE of the approximate solution) we also assign a weight to matches.

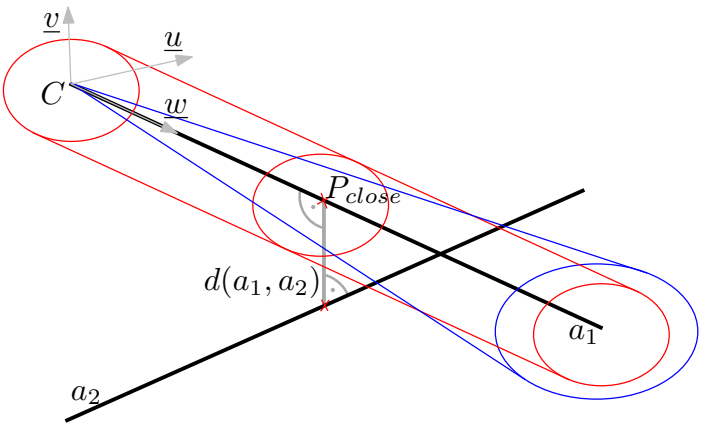


Fig. 6. Cylinder for cone matching

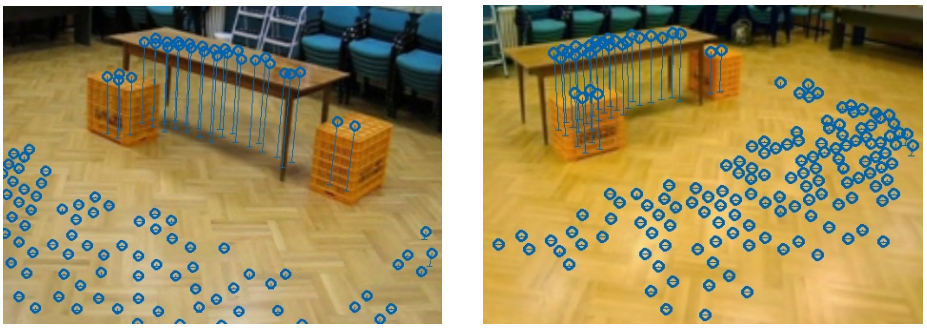


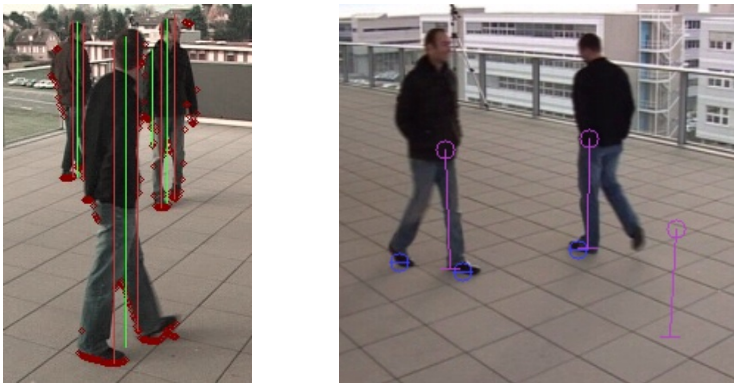
Fig. 7. Generated height map points for our test case.

6 Detection

A single object may result in multiple matches, close to each other. For this reason, we merged close matches to form detections. The 3D position of the detection is the baricenter of matches - matches might be weighted. These detections have to be filtered to eliminate false positives, and different method should be used for line and cone mathings.

For line matching, a detection is rejected upon insufficient number of matches. For a foot, many candidate pixels are present, thus many matches should appear. The number depends on the viewing angle and the distance from the camera, so minimal number of matches is hard to specify.

For cone matching, if a foot is visible from two views, exactly one match is to occur. In this case a qualitative feature is required to discard false detections. We assign a weight to the match depending on the error of the solution of (14).



(a) (b)

Fig. 8. (a) vertical foreground lines selected for body cones, and (b) detection results - blue marks foot, purple marks body detection.

7 Increasing Precision

Detection turned out to be very noisy, a lot of false positives appeared due to accidentally intersecting primitives as shown in Fig 4. We therefore examined possibilities of increasing precision.

7.1 Foot-Body Correspondences

Due to scene geometry, feet and body center show different occlusion patterns. Consequently, false correspondences of feet and bodies usually does not co-occur. We filtered vertical lines of foreground mask to generate candidate lines covering whole bodies. These lines, treated as greater bevel axis of an ellipse, can also be back-projected to cones, enabling us to detect body positions parallel to feet positions (see Fig 8 for illustration).

Our tests proved false positives usually don't co-occur, raising the precision significantly at a cost of a small decrease in recall.

7.2 Height Map

We also found that false detections of feet usually occur far from the ground, and are quite randomly distributed, thus appearing as a noise.

To suppress this noise, we aggregated all detections for a video, and highlighted dense regions with statistical filtering. Resulting dense regions mark the height points of the ground.

The phrase height map is somewhat misleading, because at the border of surfaces in different height, it is possible to have height points above each other, however in our tests this never occurred. Thereby we kept the height map expression.

A sample height map of our non planar test case is shown in Fig 7. There are certain areas where few detections took place, this led to incomplete height map.

After extracting height map, we can ignore detections far from any height point, leading to a much more reliable method.

Fig. 9. Corresponding images showing time skew

8 Experiments

We tested our algorithm on a commonly used test sequence for multiview detection, the *EPFL terrace* sequence [14] (sample results can be seen in Fig 10), and our own test videos (*SZTAKI* sequence).

In out test scene, more surfaces are present at different heights. This was required to demonstrate capabilities of our method as available multiview test sets present planar ground.

Detecting feet has advantages and disadvantages as well. Feet are always near ground, which makes height map detection possible and helps rejecting false positives. Also, for certain camera setups, feet are less likely to be occluded compared to whole bodies.

On the other hand, precise time synchronization is necessary, as a small time skew can corrupt detection. In Fig. 9, the hardly noticeable error is actually greater than the size of the foot.

8.1 Height Map Reconstruction

We reconstructed height map for both *EPFL* and *SZTAKI* sequences. We found that height map reconstruction was carried out with high precision in both cases.

EPFL sequence In case of *EPFL* sequence, we measured that the reconstructed ground is a flat plane with small error: $\mu = -1.6cm$, $\sigma = 1.7cm$. Fig 11(a) shows the histogram of height values.

Fig. 10. Corresponding frames from different cameras with the detected feet marked

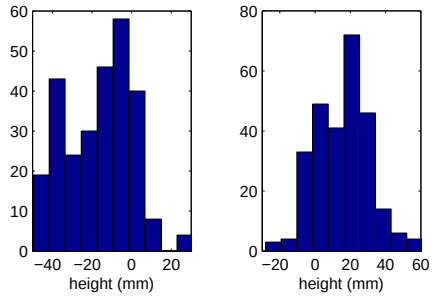
Assuming noise-like error, height values on a flat ground should follow normal distribution, however we found a different distribution. A possible source of systematic error is incorrect plane normal, so we computed optimal normal vector by fitting a plane on height points. The optimal normal vector was very close to upright direction specified by camera calibration. It diverged by only 0.36°, resulting in slightly smaller error: $\mu = 1.6cm$, $\sigma = 1.5cm$. However, using this normal led to a significantly better distribution (Fig 11(b)).

SZTAKI sequence In case of our sequence, we found that all three surfaces were found, and their height was determined with high accuracy. Measurements are summarized in Table 1. Fig 12 shows the histogram of heights for points on floor.

8.2 Detecting People

We tested our method using manually created ground truth information of feet positions. We implemented an application to create ground truth, in which we have to mark a foot in at least 2 views and its position is determined by solving a linear equation similar to (14).

1 or 2 legs can be specified for a person, because sometimes only one foot is visible from more views.



(a) (b)

Fig. 11. Histogram of height of ground points for *EPFL* dataset in (a) reference coordinate system (according to calibration data), and (b) in direction of computed normal vector.

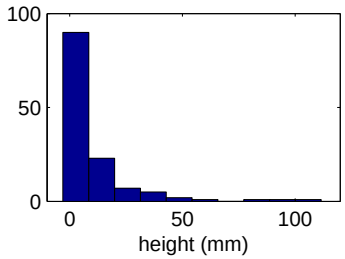


Fig. 12. Histogram of height of floor points for *SZTAKI* dataset.

Table 1. Statistical information on surfaces found in scene

surface	floor	box	table top
height (ground truth)	0cm	50cm	73cm
μ	1.1cm	50.2cm	74.4cm
σ	1.7cm	0.6cm	1.2cm
nr. of points	131	6	23
min	-0.3cm	49.1cm	70.3cm
max	11.2cm	50.8cm	75.9cm

For the evaluation, we defined a region of interest (ROI), a rectangle on the ground plane and matched detections to ground truth inside this area. In case of *EPFL* dataset, this rectangle is specified along with the dataset, for our test case this area was chosen so that every part is visible from at least 3 views.

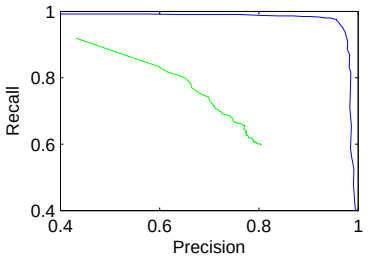
We accepted a detection as true positive, if it was not further than 25cm from a ground truth foot position. This is approximately the length of a foot. Afterward we eliminated people and detections outside the ROI, and computed false negatives and false positives from left people and detections accordingly.

Other multiview detection methods detect people unlike our method, which detects feet. Consequently, a different evaluation criteria had to be used: we considered someone detected, if either of his legs were detected.

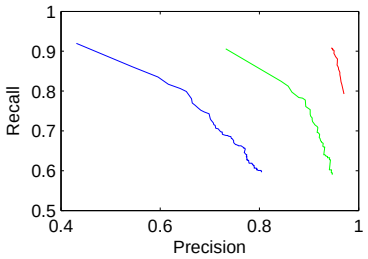
We generated precision-recall curves to evaluate our method by changing the detection threshold introduced in Section 6. Detection was very accurate using line intersection, but cone intersections lead to terrible results (see Fig 13(a)).

This performance could be highly improved using methods described in Section 7, Fig 13(b) shows results. According to these results, filtering detections to height map is mandatory for reliable operation of detector when relying on cone intersections.

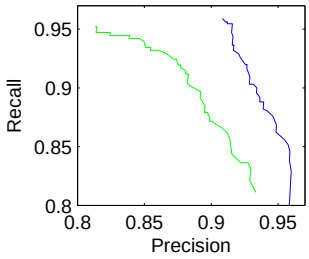
Evaluation of both data sets can be seen in Fig 13(c), in case of cone intersecting algorithm with height map filtering. These results are comparable to SOA methods according to Table 2.



(a) ROC measurement on *EPFL* sequence usgin line (blue) and cone (green) intersections



(b) ROC measurement on *EPFL* sequence: raw (blue), using body-foot correspondences (green), and height map (red)



(c) ROC measurement on *EPFL* (blue) and *SZTAKI* (green) sequence using cone matching and filtering to height map

Fig. 13. ROC curves in function of detection threshold. Evaluating different (a) detection methods, (b) filtering methods, (c) datasets.

8.3 Running Time

We measured running time of a single threaded implementation on a system with 2.4GHz Core 2 Quad CPU, on both *EPFL* (4 views with 360×288 resolution) and *SZTAKI* (4 views with 360×240 resolution) dataset. We measured similar speeds, however, without foreground detection, differences are remarkable. Measurements are shown in Table 3.

Table 2. Evaluation results on EPFL terrace sequence.

method	POM[15][a]	3DMPP[6][a]	Our method
Precision	87.20	97.5	91.28
Recall	95.56	95.5	95.01

[a] evaluation results found in [6]. In their experiments, 395 frames with 1554 object were processed.

Table 3. Average processing speed for 4 view setup (single threaded implementation, 2.4GHz Core 2 Quad CPU).

	EPFL (lines)	*EPFL* (cones)	*SZTAKI* (cones)
preprocessing	46ms	54.5ms	35.3ms
forming primitives	0.1ms	0.1ms	2ms
matching, detection	3ms	0.9ms	0.6ms
processing speed	20.4fps	18fps	26.4fps

Using cones outperforms line intersection, it is a more scalable solution - as number of matchings increase much slower with resolution or number of views. Note, that parallel processing is also possible, where preprocessing and primitive forming is done imagewise (even on smart camera), only matching and detection has to be centralized. In this architecture, low computational cost of cone matching makes our method scalable.

Above mentioned SOA methods are not well suited for real-time processing because projections are computationally expensive, and projecting parts or whole foreground masks to planes make distributed computing impossible due to data dependencies.

9 CONCLUSION

We proposed a multiview-detection algorithm that retracts 3D position of people using multiple calibrated and synchronized views. In our case, unlike other algorithms, non-planar ground can be present. This is done by modeling possible positions of feet with 3D primitives, cones in scene space and searching for intersections of these cones.

For good precision, height map of ground should be known. Our method can compute height map on the fly, reaching high precision after a startup time.

After height map detection we measured precision and recall values comparable to SOA methods on commonly used data set. Our algorithm worked well also on our test videos we made to demonstrate capabilities of handling non-planar ground.

In the future we plan to examine tracking people by their leaning leg positions [16].

References

1. A. Mittal and L. Davis, "Unified multi-camera detection and tracking using region-matching," in *Multi-Object Tracking, 2001. Proceedings. 2001 IEEE Workshop on*, 2001, pp. 3 –10.
2. Anurag Mittal and Larry Davis, "M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo," in *Computer Vision - ECCV 2002*, vol. 2350 of *Lecture Notes in Computer Science*, pp. 18–33. 2002.
3. Kideog Jeong and Christopher Jaynes, "Object matching in disjoint cameras using a color transfer approach," *Machine Vision and Applications*, vol. 19, pp. 443–455, 2008.
4. Sachiko Iwase and Hideo Saito, "Parallel tracking of all soccer players by integrating detected positions in multiple view images," in *IEEE International Conference on Pattern Recognition*. 2004, pp. 751–754, IEEE Computer Society.
5. S.M. Khan and M. Shah, "Tracking multiple occluding people by localizing on multiple scene planes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 3, pp. 505 –519, march 2009.
6. Á. Utasi and C. Benedek, "A 3-D marked point process model for multi-view people detection," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 3385–3392.
7. L. Havasi and Z. Szlavik, "A method for object localization in a multiview multimodal camera system," in *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2011, pp. 96–103.
8. Kyungnam Kim and Larry S. Davis, "Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering," in *In ECCV*, 2006, pp. 98–109.
9. Ran Eshel and Yael Moses, "Tracking in a dense crowd using multiple cameras," *International Journal of Computer Vision*, vol. 88, pp. 129–143, 2010.
10. Ákos Kiss and Tamás Szirányi, "Emberek észlelése több nézetbol tetszoleges felszínen," *VI. Magyar Számítógépes Grafika és Geometria Konferencia*, pp. 17–22, 2012.
11. Saad Khan and Mubarak Shah, "A multiview approach to tracking people in crowded scenes using a planar homography constraint," in *Computer Vision - ECCV 2006*, vol. 3954 of *Lecture Notes in Computer Science*, pp. 133–146. 2006.
12. Jerome Berclaz, Francois Fleuret, and Pascal Fua, "Robust people tracking with global trajectory optimization," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, Washington, DC, USA, 2006, CVPR '06, pp. 744–750, IEEE Computer Society.
13. C. Benedek and T. Szirányi, "Bayesian foreground and shadow detection in uncertain frame rate surveillance videos," *IEEE Trans. on Image Processing*, vol. 17, no. 4, pp. 608–621, 2008.
14. EPFL, "Multi-camera pedestrian videos," 2011, http://cvlab.epfl.ch/data/pom/.
15. Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua, "Multicamera people tracking with a probabilistic occupancy map," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 267–282, Feb. 2008.
16. Laszlo Havasi, Zoltán Szlávik, and Tamás Szirányi, "Detection of gait characteristics for scene registration in video surveillance system.," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 503–510, 2007.