Fully Automatic Horizon Estimation for Surveillance Cameras

Vojtěch Bartl
ibartl@fit.vut.cz

Adam Herout
herout@fit.vut.cz

Graph@FIT, Brno University of Technology
Brno, Czech Republic

***Abstract*—This paper deals with automatic estimation of the horizon in videos from fixed surveillance cameras. The proposed algorithm is fully automatic in the sense that no user input is needed per-camera and it works with various scenes (indoor, outdoor, traffic, pedestrian, livestock, etc.). The algorithm detects moving objects, tracks them in time, assesses some of their geometric properties related to the object dimensions and infers observations related to the position of the horizon. We collected a dataset of 47 public camera streams observing suitable scenes of various nature. We annotated ground truth horizons based on geometric properties in the images and by direct human input. We evaluate the proposed algorithm and compare it to human guesses – it turns out that the algorithm is on par with humans or it outperforms them in the difficult scenes.**

I. Introduction

Estimation of viewpoint is critical for understanding a given scene. Quick (and possibly not quite accurate) guess of the viewpoint is an important component of the *gist* of the scene [1], [2]. Creating such a representation of an image can be beneficial not only for human visual processing but also in computer vision. One of the most mentioned viewpoint aspects is the image's *horizon* [1]. Although horizon is a fairly intuitive characteristic of the viewpoint, in some complicated scenes (urban, occluded, ...), establishing exact horizon can be complicated for humans and even more so for machines.

Horizon is very often simply explained as "the line where the sky meets the ground". This explanation corresponds to the intuitive human feeling about the horizon. The astronomical horizon is defined by the horizon plane, a plane that is perpendicular to gravity and located at the same altitude as the camera [1]. In a given image, the plane is only visible as a horizontal line, it is not dependent on the slant of the ground surface nor on the presence of occluders. Literature also defines a "horizon of an arbitrary plane" which is the line in the image where all parallel lines within this plane meet (i.e. the projection of the plane's ideal line) [3]. Horizontal lines that are parallel in the real world meet at a vanishing point which lies on the horizon. Horizon is thus naturally, and at no additional cost, established by algorithms that recognize vanishing points in Manhattan worlds, *e.g.* [4].

Horizon is very often used for camera calibration because two horizontal vanishing points lie on the horizon line. With the knowledge of the horizon line and the third (vertical) vanishing point, camera calibration can be done [5], [6]. An approach which estimates the vanishing points by connecting corresponding points of the same object and then constructs the horizon is often used for camera calibration with human tracking [7], [8], [9], [10], [11], where pedestrian's head and feet are connected by lines as the person moves across the scene and their intersection lies on the horizon line. Similarly, vanishing points and the horizon line can be localized for example by detecting pedestrians' toes in the ground plane [12]. Although camera calibration methods which avoid using horizon exist, they typically have some constraints, for example, known pedestrian height distribution [13], [14], 'Manhattan world' scene [15], [16], [17], or dominant motion only in one coherent direction [18], [19]. Horizon can also be used for estimating 3D scene geometry and object detection support [20], [21].

Our goal is to detect the horizon (in the sense of the ideal line of the ground plane perpendicular to the gravity) in a single static (often surveillance) uncalibrated camera stream based on motion of objects in the scene without any a priori constraints except that the majority of motions happen in horizontal planes (which share the same horizon by definition). Such a method can be later used for automatic camera calibration, scene understanding, and other computer vision tasks. We assume that the scene contains arbitrary objects (pedestrians, cars, dogs, cattle, machinery, ...) with an arbitrary height/size distribution. The scene does not need to be manhattanian and the motion can appear in any direction and in virtually any place in the scene. Existing methods for horizon estimation [22], [23] use a single static image without assumption of a movement in the scene. Although motion can be used for horizon estimation for example in the form of cloud motion together with wind velocity [24], in our work we assume surveillance cameras without any constraints, no clouds thus need to be present in the scene and no additional information is provided.

Although datasets with horizon position in image exist (HLW [25], ECD [26], YUD [27]), they only contain static images without any motion in the scene. To evaluate the method and to allow for future comparison (to our knowledge, there is no existing dataset dealing with this issue), we collected a dataset based on publicly available IP cameras. We manually constructed a ground truth by using geometric properties of objects in the scene and we also collected human annotations which cast a light on the algorithm's performance and allow for its comparison to human (well trained and routinely used)

gist of the scene mentioned earlier. The second contribution of this paper – after proposing the fully automatic horizon estimation algorithm – is making this dataset public[1].

II. Horizon Estimation by Observing Motion

This section describes the use of object trajectories in the scene and it proposes an algorithm for fully automatic horizon estimation from them.

A. Trajectories and Horizons

Our horizon estimation is based on trajectories of objects tracked in a video (Figure 1). The objects can be arbitrary and heterogeneous in the scene (unlike many approaches we do not focus on a given known object class such as pedestrians or vehicles). The video is first transformed into a set of trajectories T by tracking foreground objects:

$$T = \{\mathbf{t}_1, \ldots, \mathbf{t}_N\} \tag{1}$$

Every trajectory is composed of individual object observations

$$\mathbf{t} = \{\mathbf{o}_1, \ldots, \mathbf{o}_{M_t}\} \tag{2}$$

which will be referenced as $\mathbf{t}(1), \mathbf{t}(2), \ldots, \mathbf{t}(M_t)$ for brevity. Each observation is described by a pair $(\mathbf{p}, \mathbf{d})$ whose parts are:

$$\mathbf{t}(i)^{\mathbf{p}} \quad - \quad \text{position of the observation} \tag{3}$$
$$\mathbf{t}(i)^{\mathbf{d}} \quad - \quad \text{dimensions of the observation} \tag{4}$$

Position $\mathbf{t}(i)^{\mathbf{p}}$ is naturally the 2D position of the center of the observation in the image. The dimensions $\mathbf{t}(i)^{\mathbf{d}}$ are possible dimensions of the observation (naturally width and height, possibly diagonals, other chord lengths, a measure related to the area, ...) and so $\mathbf{d}$ is potentially k-dimensional. Since we are dealing with quite low-resolution videos, we are only using width, height and diagonal of the axis-aligned bounding box in our experiments and k is thus 3 ($\mathbf{d}$ is three-dimensional). Here and later in the text, we are assuming 'normal' surveillance camera orientation; should the camera be placed so that the horizon turns 'vertical', such a situation would be easily detected and the algorithm can be adjusted in a straightforward way. We are omitting such adjustments here for simplicity.

Let us consider an arbitrary horizon

$$\mathbf{h} = (\mathbf{h}_m, \mathbf{h}_b), \tag{5}$$

a straight line in the slope-intercept form

$$y = \mathbf{h}_m x + \mathbf{h}_b. \tag{6}$$

We define the *distance* of a given observation with the position $\mathbf{t}(i)^{\mathbf{p}}$ from the given horizon $\mathbf{h}$ as

$$\Delta(\mathbf{t}(i), \mathbf{h}) = \left| \frac{\mathbf{h}_m \mathbf{t}(i)^{\mathbf{p}}_x - \mathbf{t}(i)^{\mathbf{p}}_y + \mathbf{h}_b}{\sqrt{\mathbf{h}_m^2 + 1}} \right| \tag{7}$$

If the object captured by a trajectory $\mathbf{t}$ is moving within the plane corresponding to the horizon (and no occlusion is involved etc.) and the dimensions $\mathbf{d}$ are correctly selected, the measured dimensions $\mathbf{t}(i)^{\mathbf{d}}$ must be linearly dependent on the distances $\Delta(\mathbf{t}(i), \mathbf{h})$, as illustrated in Figure 1:

$$\lambda\, \Delta(\mathbf{t}(i), \mathbf{h}) = \mathbf{t}(i)^{\mathbf{d}}. \tag{8}$$

[1] https://medusa.fit.vutbr.cz/trajectories

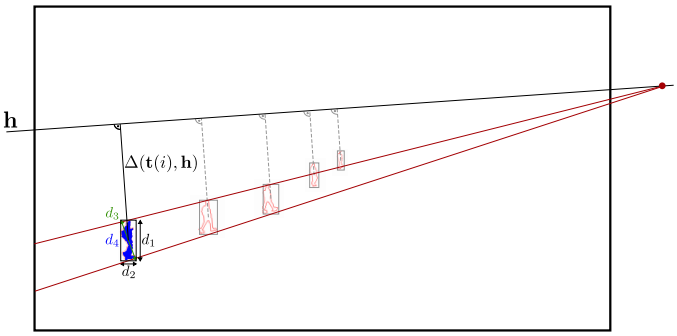


Fig. 1. One possible trajectory (a few selected observations) in the video frame. Each observation consists of position $\mathbf{t}(i)^{\mathbf{p}}$ and possible dimensions $\mathbf{t}(i)^{\mathbf{d}}$. The distance $\Delta(\mathbf{t}(i), \mathbf{h})$ from horizon $\mathbf{h}$ is also depicted.

Fig. 2. Example of a scene with one dominant direction of objects' motion. There are two directions of car movement in the road and a bidirectional stream of pedestrians on the sidewalk. Most of the tracks only testify for one vanishing point, only rare pedestrians crossing the road bring the missing information.

B. Trajectory Contribution

Some scenes are typically problematic for horizon estimation; mainly scenes with one dominant direction of motion present. In such cases, the motion provides information about one vanishing point (situated on the horizon line), but not enough information about the whole horizon. It is thus necessary to compute the *contribution value* of the given track, where tracks in rare directions should contribute more, since they provide valuable information. An example of a scene with one dominant motion direction is shown in Figure 2.

For computing the contribution value for each trajectory $\mathbf{t}$, histogram $\mathcal{D}$ is accumulated, which stores information about directions of motions which appear in the scene. Histogram $\mathcal{D}$ contains B bins separating the interval of angles $(0, \pi)$ uniformly. The trajectory is divided to C parts uniformly, each part described by its motion direction (see Figure 3).

For every trajectory part vector defined by its endpoint indices a, b, i.e. $\vec{v} = \mathbf{t}(b)^{\mathbf{p}} - \mathbf{t}(a)^{\mathbf{p}}$, its angle to an arbitrary fixed

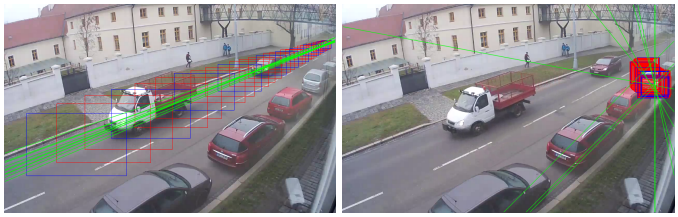


Fig. 3. An example of computing the trajectory contribution value. Rectangles are observations, endpoint observations of the trajectory parts are blue. Direction vectors $\vec{v} = \mathbf{t}(b)^{\mathbf{p}} - \mathbf{t}(a)^{\mathbf{p}}$ for each consequent couple of endpoint indices a, b are shown as green lines. **left:** Typical movement in the single dominant direction (car on the road). **right:** Motion in an uncommon direction (car leaving parking place).

reference horizon vector $\vec{\mathbf{h}}_{ref}$ (typically the vector horizontal in the image) is computed as

$$\phi = \arccos(\vec{\mathbf{h}}_{ref} \cdot \vec{v}), \tag{9}$$

with the assumption of normalized vectors. Angle ϕ determines which bin of histogram $\mathcal{D}$ is incremented for each trajectory's part. Histogram bin values are computed for all C parts of the trajectory and the final contribution value $\gamma(\mathbf{t})$ is computed by normalizing the sum of values of the corresponding histogram bins using the sum of all histogram values. It should be noted that for efficiency of the computation, the histogram accumulation and evaluation of the contribution value is done on the fly with further trajectories coming. The histogram is first initialized by a non-zero value Γ configuring the initial estimation of the contribution values.

C. Automatic Horizon Estimation

Parametric space $\mathcal{M} \times \mathcal{Y}$ of potential horizons is generated for horizon estimation. This parametric space $\mathcal{M} \times \mathcal{Y}$ samples the set of potential horizons $\mathbf{h} = (\mathbf{h}_m, \mathbf{h}_b)$. Each horizon is the line with slope $\mathbf{h}_m$, whose $\mathbf{h}_b$ is computed so that the line intersects the vertical central line of the image in the y coordinate. $\mathcal{M}$ are slopes of angles from interval $(-\beta, \beta)$ sampled with step s_m, and $\mathcal{Y}$ are vertical positions of the horizons from interval $(-y_{gen} f_h, y_{gen} f_h)$ sampled with step s_y, where f_h is the height of the video frame and y_{gen} is a configuration constant, in the experiments $y_{gen} = 80\,\%$, as illustrated by Figure 4.

For every potential horizon $\mathbf{h}$ and every trajectory $\mathbf{t}$, a confidence value is computed indicating how much the linear dependency assumption (8) of observation's distances from the horizon and their dimensions is met. In particular, least-squares linear regression is computed between the presumably linearly dependent values, and its error $\epsilon(\mathbf{t}, \mathbf{h})$ is computed as the root mean square error (RMSE):

$$\epsilon(\mathbf{t}, \mathbf{h}) = \sum_{j=1}^{k} \mathbf{w}_j \sqrt{\frac{1}{M_t} \sum_{i=1}^{M_t} \left(\mathbf{t}(i)_j^{\mathbf{d}} - \lambda\, \Delta(\mathbf{t}(i), \mathbf{h})\right)^2}, \tag{10}$$

where $\mathbf{w}$ is k-dimensional vector of weights for individual dimensions of observation $\mathbf{t}(i)^{\mathbf{d}}$. Regression error $\epsilon(\mathbf{t}, \mathbf{h})$ describes how (un)likely the potential horizon $\mathbf{h}$ can be the real horizon of trajectory $\mathbf{t}$. The trajectory's contribution value $\gamma(\mathbf{t})$

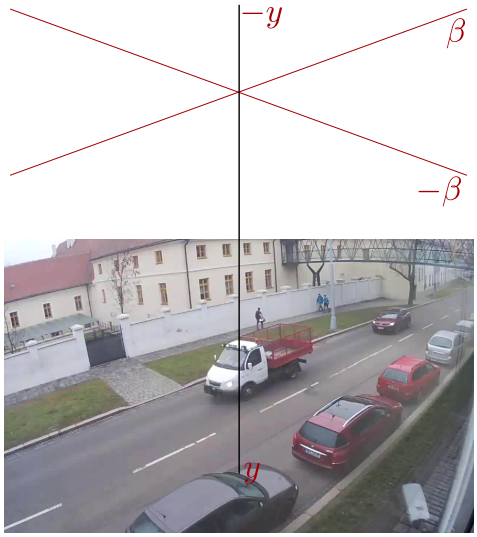


Fig. 4. Generated potential horizons which are used for horizon estimation. In this case with value $y_{gen} = 80\,\%$ (y values from interval defined by 0.8 times height of frame above and below top frame border) and slope angle values from interval $(-\beta, \beta)$.

(Section II-B) is also accounted for in the resulting confidence $c(\mathbf{t}, \mathbf{h})$ of the trajectory $\mathbf{t}$ being assigned to potential horizon $\mathbf{h}$. The resulting confidence is then

$$c(\mathbf{t}, \mathbf{h}) = \frac{1}{\delta_e + \epsilon(\mathbf{t}, \mathbf{h})} \cdot \frac{1}{\delta_c + \gamma(\mathbf{t})}, \tag{11}$$

where parameters δ_e and δ_c control the weight of regression error and trajectory contribution value in the resulting confidence and they also ensure computational stability. This confidence is computed for every trajectory $\mathbf{t}$ and every potential horizon $\mathbf{h}$ and it is accumulated to the parametric space of potential horizons $\mathcal{M} \times \mathcal{Y}$ in a manner similar to the Hough transform and also following the work of Litman *et al.* [28] and the work by Zhai *et al.* [22]. After accumulating all the tracks' confidences, the horizon in the parametric space with the highest confidence is selected as the most likely solution.

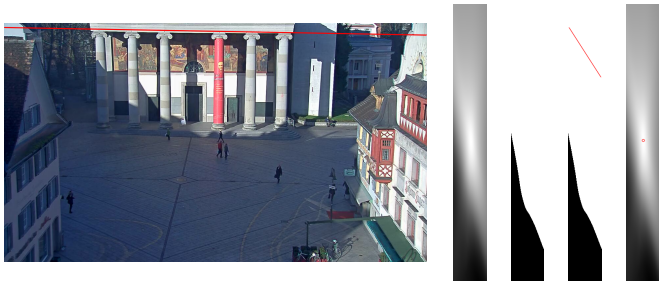
D. Motion with One Dominant Direction

Although the trajectory contribution value, as described in Section II-B, is computed to emphasize trajectories with unusual direction, in some specific scenes, the motion happens solely in one dominant direction. In such cases, the trajectories do not provide enough information about the horizon, only information about a single vanishing point corresponding to the dominant direction. This is typical mainly for traffic surveillance scenes, where only the single motion direction parallel to the road direction can be seen. An example of such a scene is shown in Figure 5 (a).

In such problematic scenes, there is a typical pattern in the parametric space $\mathcal{M} \times \mathcal{Y}$, where confidences are accumulated mainly for potential horizons all of which are coincident with the vanishing point given by the motion in the scene. The parametric space for such a scene is shown in Figure 5 (b). It manifests as a pattern resembling a 'line' in the parametric

(a) (b) (c) (d) (e)

Fig. 5. Scene with a single dominant motion direction. (a) The scene. The final horizon given by the fallback solution is marked by red line; (b) Parametric space $\mathcal{M} \times \mathcal{Y}$ of the scene with apparent vanishing point pattern; (c) Parametric space $\mathcal{M} \times \mathcal{Y}$ after OTSU threshold computation; (d) Line fitted to the non-zero values from (c) for points' mean distance computation to 'line' pattern detection; (e) Parametric space $\mathcal{M} \times \mathcal{Y}$ with fallback solution (zero slope) marked by red circle.



(a) (b) (c) (d) (e)

Fig. 6. Scene with multi-direction movement. (e) Parametric space $\mathcal{M} \times \mathcal{Y}$ with solution marked by red circle (no fallback)

space – all potential horizons with high confidence intersect near one vanishing point. Simply selecting one point with the highest confidence would be random (provided that the point is on the line corresponding to the vanishing point). This situation is recognized by thresholding the parametric space by Otsu's algorithm [29] (Figure 5 (c)). A least-squares error line is fitted to the non-zero points in the thresholded image (Figure 5 (d)) and if the mean non-zero points' distance to the fitted line is below a threshold, then the one dominant motion situation is recognized. As a fallback solution, the horizon is declared to be the one horizon with zero slope and the highest confidence (Figure 5 (e)). A non-problematic scene with multi-direction movement is shown in Figure 6 for contrast; the 'line' pattern in the parametric space is not present and the fallback solution is not necessary.

III. Dataset – Geometric and Human Annotations

This section describes the dataset collected for testing the horizon estimation and its annotations (geometric and human).

A. Data Collection

Most of the recordings were taken from publicly available IP cameras, some recordings were captured by a camcorder.

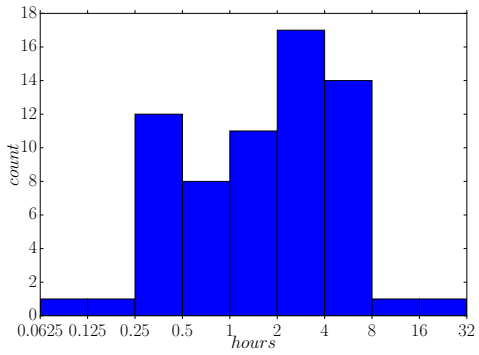


Fig. 7. Histogram of video durations in the dataset.

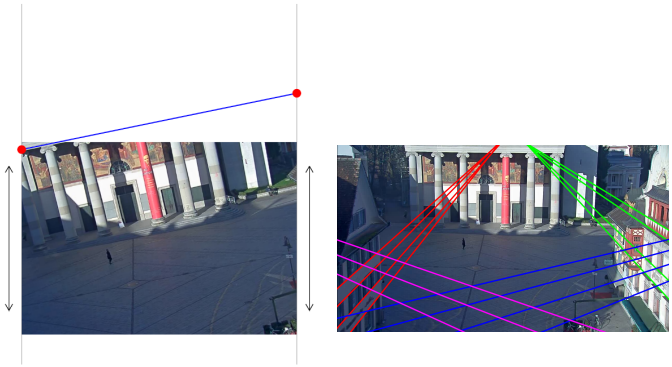


Fig. 8. Scene horizon annotation principles. **left:** Web annotation tool for crowsourced data collection. **right:** 'Geometric' ground truth annotation by using scene parallel lines.

One scene was used from the PETS dataset but it is not a very suitable one because of its short duration. One overcrowded scene was used from [30] to cover as much complex scenes as possible.

The recordings differ in many aspects – places, camera positions, day time, scene type, duration, resolution, ... The collection includes scenes from traffic, indoor, outdoor, pedestrians, etc. Some recordings were taken during night so different light conditions are also available. The duration of the recordings is in the range from 5 minutes to 30 hours, mean length is about 2.9 hours (details in Figure 7). The resolution is largely varying with the given IP camera's quality in the range from 320×240 to $1\,920 \times 1\,080$ pixels. In total, 47 different usable scenes were obtained. Some scenes were re-captured under different conditions (lighting, crowd density, ...), yielding 66 recordings in total.

B. Horizon Annotations

Obtaining horizon ground truth turned out to be a challenging problem mostly due to very frequent occlusion of the natural horizon in the scenes (buildings, horizon out of frame, ...). In order to obtain a geometrical estimation of the horizon, we extracted one representative frame from the video recording and manually annotated groups of lines that are parallel in the original 3D scene (edges of a house's windows, markings on the streets, patterns in the pavings, etc.). Each

Fig. 9. Examples of humans' annotations. Blue lines are individual annotations, cyan is the mean horizon location. **left:** Convenient scenes. **right:** Scenes with high variance in the annotations.

of these groups of lines provides one estimated vanishing point; all vanishing points should be collinear – coincident with the line of the horizon. The horizon is obtained by using the least-squares linear regression on the set of the estimated vanishing points obtained as the minimal error intersections of the lines in the individual groups. Such obtained horizon is referenced as the 'geometric' annotation later in this text. This geometric horizon line is established for every scene of the dataset; an example of this annotation process is depicted in Figure 8 right.

Aside from the geometric horizon estimation, we collected horizon annotations by humans. We created a web annotation tool (Figure 8 left) and knowledgeable people were asked to estimate the horizon in the scene frame as precisely as possible. As described by Herdtweck and Wallraven [1], people are able to localize the horizon in a given image with small error after a short description of what horizon really is. To prevent people of simply assuming a horizon line being always horizontal in the image (though this is common in many camera shots), the images given to the users for annotation were rotated by $\pm 20°$ and the maximal rectangle was slightly cropped as in Figure 8 left.

TABLE I
ALGORITHM PARAMETERS

Name	Usage	Value
B	Section II-B	45 (each bin covers $4°$)
C	Section II-B	10
Γ	Section II-B	10.0
$\vec{\mathbf{h}}_{ref}$	Section II-B	$[1, 0]$ – zero slope
β	Section II-C	$20°$
s_m	Section II-C	$0.5°$
y_{gen}	Section II-C	0.8
s_y	Section II-C	1.0 px
δ_e	Section II-C	5e−2
δ_c	Section II-C	5e−4

Participants estimated the horizon by moving a visual line with markers on its sides as precisely as possible – the users could try different positions of the controlled line and look for the best match. Every participant marked 20 least annotated scenes. The annotations were filtered to rid of annotations clearly skipped or carelessly performed. Finally, 16 – 21 annotations for each of the 47 scenes are available (mean number 18.42 annotations per scene by different human subjects). Some examples of annotated scenes are depicted in Figure 9. It is apparent that in some scenes, it is very difficult for a human to mark the horizon (Figure 9 right) – mostly in cases when the horizon is 'somewhere above' the frame or totally occluded. In some scenes (Figure 9 left), the correlation between the horizons indicated by humans is very high.

C. Trajectories Data

Object tracking (together with the necessary video decompression) is computationally the most difficult part of the whole process of horizon estimation and so these data are stored for faster processing and also for possible later usage of these data by other users as part of our dataset. We used our own implementation of the object tracking method proposed by Yang *et al.* [31]. For every scene in the dataset, the following information are stored:

- Scene name
- 'Geometric' annotation
- Humans' annotations
- Contours for all observations of individual trajectories

IV. Experimental Results

This section evaluates the algorithm by comparing it to the humans' and 'geometric' annotations on all the 47 scenes (66 recordings) from our dataset.

A. Experimental Setup

All experiments used parameters defined in Table I. Preliminary experiments showed that the computation is not very sensitive to settings of parameters δ_e and δ_c. As was mentioned in Section II-A, axis-aligned bounding boxes of the observations are used for computation of observations' dimensions $\mathbf{t}(i)^{\mathbf{d}}$. Experiments with contour usage instead of axis-aligned bounding box were done but contours' dimensions incline to be more noisy.

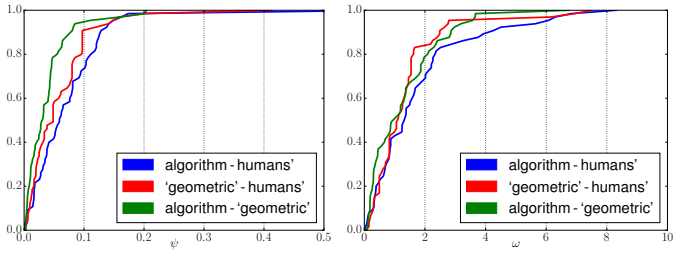


Fig. 10. Cumulative histograms of differences between different methods of obtaining the horizons (human crowdsourced annotation, 'geometric' horizons, algorithmic method). **left:** differences in vertical position ψ (relative to frame height), **right:** differences in the horizon's angle ω in degrees.

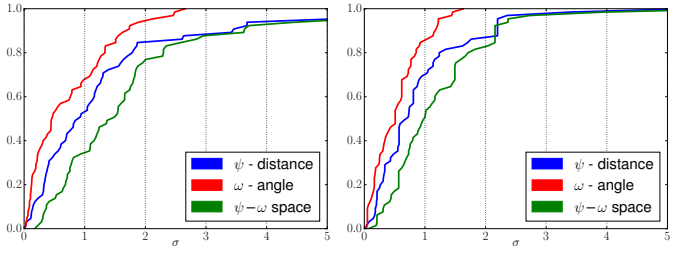


Fig. 11. Cumulative histograms of output distances to the human input means in terms of σ. **left:** algorithmic horizons, **right:** 'geometric' horizons.

After the preliminary experiments, the weight vector $\mathbf{w}$ (Section II-C) was set to the value $\mathbf{w} = (0.7; 0.3; 0.6)$, where the observation dimensions $\mathbf{t}(i)^{\mathbf{d}}$ are sequentially: bounding box height, width, and diagonal length. Height provides the most valuable contribution to the resulting error computation (10) and is well usable for humans (objects which do not change dimensions rapidly during motion). Width and diagonal lengths help with computation for objects, which change their dimensions by rotation in the scene (typically vehicles).

B. Evaluation and Results

The experiments are done by comparing the proposed algorithm outputs with the humans' and the 'geometric' annotations in their values of ω (angle of horizon's slope in degrees) and ψ (vertical position of the horizon, relative to image height, i.e. 0 at image top, 1.0 at image bottom.

The left graph in Figure 10 displays the absolute differences between the horizons' positions ψ, comparing the algorithmic outputs to the humans' annotations (mean horizon) and to the 'geometric' references: when compared to the humans' annotations, in $80\,\%$ cases, the relative vertical position difference is under 0.112 (cca $11\,\%$ of the image height), and $95\,\%$ of cases fit under 0.144 relative vertical position difference. When compared to the 'geometric' annotations, the values of ψ are bellow 0.055 in $80\,\%$ of cases and bellow 0.083 in $95\,\%$ cases. The right plot in Figure 10 shows the absolute differences between the horizon angles, compared both to the humans' annotations and to the 'geometric' reference. When compared to the human annotations, in $80\,\%$ cases the angular error (in degrees) was below $2.33°$ and in $95\,\%$ cases below $5.7°$. When compared to the 'geometric' annotations, in $80\,\%$ cases the value was below $2.08°$, and in $95\,\%$ cases it fit under $3.3°$.

The graphs in Figure 10 indicate that the algorithm's outputs compared to 'geometric' ground truth are on par or outperform the human 'guesses'. We therefore conclude that the algorithmic approach based on observing moving objects can provide the 'gist' of the scene [1], [2] used by humans for understanding an arbitrary visual scene.

For another comparison with the humans' annotations, the mean of annotation horizons is taken as the reference value and the errors are expressed in terms of standard deviation σ – see Figure 11 for the results. The left graph shows the difference of the algorithmically obtained horizons from the mean of human annotations; the right graph shows the distance of the 'geometric' annotations from the mean of the human annotations. It is apparent that humans can indicate the horizon quite accurately, because the Mahalanobis distance in 2D $\psi-\omega$ space (as depicted also in Figure 12) is bellow 1σ in $51\,\%$ of cases and in $83\,\%$ cases bellow 2σ.

Some examples of the proposed algorithm outputs can be seen in Figure 12 together with the humans' and 'geometric' references.

The main source of inaccuracies is the noise in the tracked data – occlusions of the tracked objects, pixel segmentation imprecisions, etc. Despite that, the method works with comparable accuracy as the human annotators; in difficult scenes, our method outperforms humans (with the 'geometric' horizon as the reference). One source of inaccuracy is the radial distortion of many of the cameras. In such cases, the horizons (lines) produced by our method behave similarly to the human annotations and to the geometric construction. Thus established approximate horizon can still serve the purpose of basic understanding of the scene. The final observation is that with longer videos, the results improve (as expected) and the noise cancels out. Our method is therefore suitable for fixed surveillance cameras, because with more coming data, it has the chance to improve (contrary to methods only processing one frame of the video).

State-of-the-art method for horizon detection by only processing one image by Zhai *et al.* [22] fails when short line segments or curved structures are present in a scene (relies on line segments). Our proposed method does not rely on visual information thus it can also handle this type of scenes. By contrast, compared to methods only processing one image, the proposed method can finish with different results for the same scene (depends on the motions happening in the scene at the given time). Figure 13 shows results for different recordings of the same scene.

V. Conclusions

This paper introduced an algorithm for estimating the horizon in surveillance videos based only on motion in the scene. Contrary to existing approaches, our algorithm does not assume presence of particular objects (such as vehicles or humans) in the scene, but it works with arbitrary scenes.

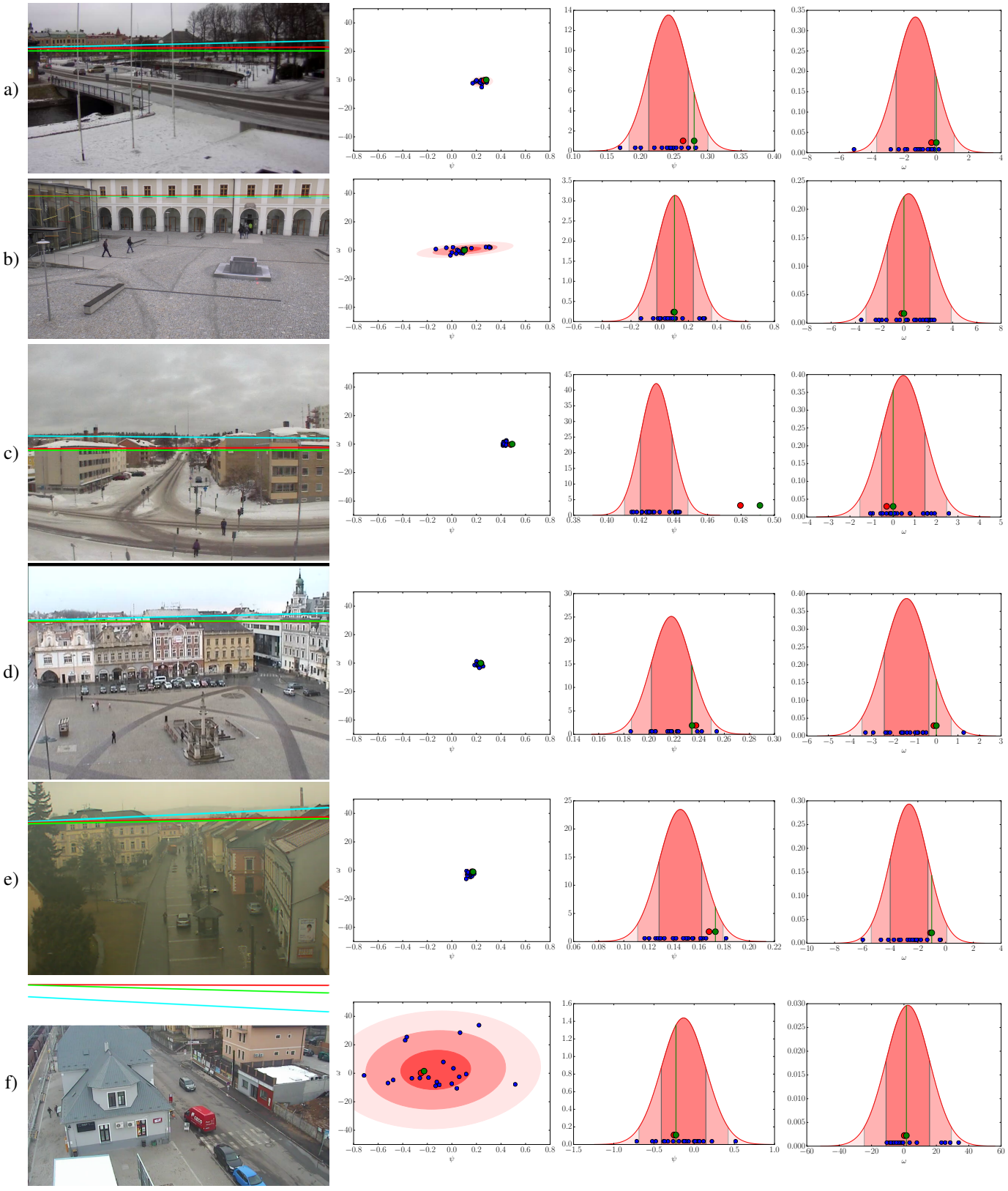


Fig. 12. Selected scene samples. **camera images** include alternative horizons; *red:* 'geometric' horizon, *green:* horizon computed from the video, *cyan:* mean of human inputs. **graphs:** *blue dots* are individual human annotations, *green dot* is computed by the algorithm, *red dot* is the 'geometric' horizon, *shades of red* are parts of normal distribution of the human inputs $(1\sigma, 2\sigma, 3\sigma)$ *left:* $\psi - \omega$ plot, *center:* distribution along ψ axis, *right:* distribution along ω axis.

Fig. 13. Resulting detected horizons for different recordings of the same scene (different lightning conditions, daytime, present objects, ...); *red:* 'geometric' horizon, *green:* horizon computed from the video, *cyan:* mean of human inputs

We collected a set of videos from real-life web cameras, surveillance cameras, and other scenes, and make it public along with this paper. The videos in this dataset are very diverse (in terms of scale, nature of the scene, type of objects appearing in, horizon position, lighting, etc.). We provide two kinds of annotations of this dataset: geometrically extracted horizons, and direct human annotations. The dataset also contains the tracks of the objects moving in the scene.

Our algorithm manages to get the 'gist' of the scene, that could help other tasks of computer vision (as it has been shown that it helps humans in their understanding). The experiments show that the accuracy achieved by our solution is comparable to the performance of human annotators; some scenes even confused the human annotators so much that our algorithm outperformed humans. Our purpose was to show that this task is possible to solve and to establish a baseline for further development.

REFERENCES

[1] C. Herdtweck and C. Wallraven, "Estimation of the horizon in photographed outdoor scenes by human and machine," *PLoS ONE*, vol. 8, no. 12, pp. 1–14, 12 2013.

[2] G. A. Rousselet, O. R. Joubert, and M. Fabre-Thorpe, "How long to get to the "gist" of real-world natural scenes?" *Visual Cognition*, vol. 12, no. 6, pp. 852–877, 2005.

[3] K. Andersen, *Brook Taylor's Work on Linear Perspective*. Springer New York, 1992.

[4] J. Košecká and W. Zhang, "Video compass," in *ECCV*, 2002.

[5] B. Caprile and V. Torre, "Using vanishing points for camera calibration," *International Journal of Computer Vision*, vol. 4, no. 2, pp. 127–139, 1990.

[6] R. Orghidan, J. Salvi, M. Gordan, and B. Orza, "Camera calibration using two or three vanishing points," in *2012 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sept 2012, pp. 123–130.

[7] I. Junejo and H. Foroosh, "Robust auto-calibration from pedestrians," in *2006 IEEE International Conference on Video and Signal Based Surveillance*, Nov 2006, pp. 92–92.

[8] W. Kusakunniran, H. Li, and J. Zhang, "A direct method to self-calibrate a surveillance camera by observing a walking pedestrian," in *2009 Digital Image Computing: Techniques and Applications*, Dec 2009, pp. 250–255.

[9] F. Lv, T. Zhao, and R. Nevatia, "Self-calibration of a camera from video of a walking human," in *Object recognition supported by user interaction for service robots*, vol. 1, 2002, pp. 562–567 vol.1.

[10] ——, "Camera calibration from video of a walking human," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1513–1518, Sept 2006.

[11] G. M. Y. E. Brouwers, M. H. Zwemer, R. G. J. Wijnhoven, and P. H. N. de With, *Automatic Calibration of Stationary Surveillance Cameras in the Wild*. Cham: Springer International Publishing, 2016, pp. 743–759.

[12] S. Huang, X. Ying, J. Rong, Z. Shang, and H. Zha, "Camera calibration from periodic motion of a pedestrian," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] L. Teixeira, F. Maffra, and A. Badii, *Scene Understanding for Auto-Calibration of Surveillance Cameras*. Cham: Springer International Publishing, 2014, pp. 671–682.

[14] J. Liu, R. T. Collins, and Y. Liu, "Surveillance camera autocalibration based on pedestrian height distribution," *British Machine Vision Conference*, Jan 2010.

[15] S. C. Lee and R. Nevatia, "Robust camera calibration tool for video surveillance camera in urban environment," in *CVPR 2011 WORKSHOPS*, June 2011, pp. 62–67.

[16] J. Deutscher, M. Isard, and J. MacCormick, "Automatic camera calibration from a single manhattan image," in *Computer Vision—ECCV 2002*. Springer, 2002, pp. 175–188.

[17] H. Wildenauer and A. Hanbury, "Robust camera self-calibration from monocular images of manhattan worlds," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 2831–2838.

[18] M. Dubská, J. Sochor, and A. Herout, "Automatic camera calibration for traffic understanding," *British Machine Vision Conference*, Jan 2014.

[19] M. Dubská, A. Herout, R. Juránek, and J. Sochor, "Fully automatic roadside camera calibration for traffic surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1162–1171, June 2015.

[20] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 3–15, 2008.

[21] P. Wang, K. Morton, P. Torrione, and L. Collins, "Viewpoint adaptation for rigid object detection," 2017, arXiv:1702.07451.

[22] M. Zhai, S. Workman, and N. Jacobs, "Detecting vanishing points using global image context in a non-manhattan world," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] J. Lezama, R. Grompone von Gioi, G. Randall, and J.-M. Morel, "Finding vanishing points via point alignments in image primal and dual domains," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[24] N. Jacobs, M. T. Islam, and S. Workman, "Cloud motion as a calibration cue," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[25] S. Workman, M. Zhai, and N. Jacobs, "Horizon lines in the wild," in *British Machine Vision Conference (BMVC)*, 2016.

[26] O. Barinova, V. Lempitsky, E. Tretiak, and P. Kohli, "Geometric image parsing in man-made environments," in *ECCV*, 2010.

[27] P. Denis, J. H. Elder, and F. J. Estrada, "Efficient edge-based methods for estimating manhattan frames in urban imagery," in *ECCV*, 2008.

[28] R. Litman, S. Korman, A. Bronstein, and S. Avidan, "Inverting ransac: Global model detection via inlier rate estimation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 5243–5251.

[29] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[30] S. Yi, H. Li, and X. Wang, "Understanding pedestrian behaviors from stationary crowd groups," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3488–3496.

[31] T. Yang, Q. Pan, J. Li, and S. Z. Li, "Real-time multiple objects tracking with occlusion handling in dynamic scenes," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 970–975 vol. 1.