Surface Prediction for a Single Image of Urban Scenes

Foat Akhmadeev

Institute of Computer Mathematics and Information Technologies,
Kazan Federal University, Kazan, Russian Federation

Abstract. In the paper we present a novel method for three-dimensional scene recovering from one image of a man-made environment. We use image segmentation and perspective cues such as parallel lines in space. The algorithm models a scene as a composition of surfaces (or planes) which belong to their vanishing points. The main idea is that we exploit obtained planes to recover neighbor surfaces. Unlike previous approaches which use one base plane to place reconstructed objects on it, we show that our method recovers objects that lie on different levels of a scene. Furthermore, we show that our technique improves results of other methods. For evaluation we have manually labeled two publicly available datasets. On those datasets we demonstrate the ability of our algorithm to recover scene surfaces in different conditions and show several examples of plausible scene reconstruction.

1 Introduction

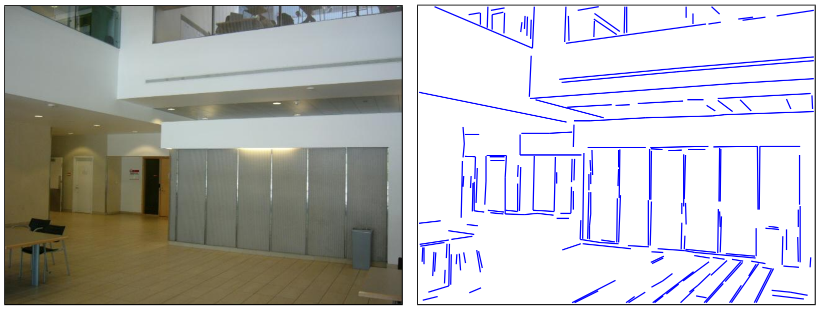

(a) (b)

Fig. 1. Most people can infer the 3D structure of the scene from the image above. It is a little harder from (b), but it can be done too.

Generally, some computer vision and image processing applications may benefit from scene reconstruction. E.g. some reconstructed objects like pillars can be used to avoid them by robots [13]. This knowledge may greatly improve object detection and their understanding [13, 23, 12, 18]. Moreover, full scene recovering can be applied in rendering synthetic objects into photos [22, 13, 20]. Image surfaces can be used in image segmentation, e.g. road, buildings [8]. In addition, spatial layout and spatial understanding can be improved by scene recovering [23, 12].

A plausible recovery of a 3D scene is a major problem for most single view 3D reconstruction algorithms. Look at the image in Fig. 1a. From this single image you can see the basic structure of the scene, which consists of orientation of main surfaces like walls, floor and ceiling. Nevertheless, automatic reconstruction of a scene from a monocular image is challenging due to the ambiguity of the perspective projection. Often an image of interests comes from urban or indoor scenes where certain structural regularities are presented, this can help us to disambiguate this problem. Those scenes usually contain a lot of straight lines (see Fig. 1b) which can be grouped as parallel lines in space. Most viewers can recognize structure of a scene even from that image. However, there are a lot of missing lines, because not all of them are perfectly detected by low level algorithms. People can still understand that scene because a) we see all objects in a case of perspective, so we can recognize objects if we see their projections on a plane, b) we can show where objects boundaries are, because we see how lines are grouped on that image.

In the paper we present an algorithm whose purpose is to reconstruct a 3D model from a single image. The basic assumption which we chose is that the main characteristics of a scene should have an agreement with "Manhattan" worlds [3]. This assumption states that most of horizontal lines in space are divided into two orthogonal groups.

The first step of our method includes the use of parallel lines in space for recovering scene planes. Then, those planes are used to predict other scene surfaces. The information extracted by the algorithm contains main planes of a scene like floors, ceilings, walls and facets of big objects. This can be done only through getting main camera parameters from an image. Camera characteristics like the focal length are gotten automatically. To get the focal length we use vanishing points. Vanishing point is a single point on the image plane which is created by intersection of projections of parallel lines on an image.

The novelty of the algorithm is that it recovers objects from different levels of a scene and it does not use only one bearing surface, e.g. ground. In addition, our method works on both indoor and outdoor man-made scenes. Next, we propose the formulation of *surface prediction* approach. Eventually, we show combinations of our algorithm with other approaches that outperform state-of-the-art methods.

2 Prior Work

Recent years have seen a large progress in scene understanding. Many works state that the Manhattan world assumption can be enough to generate plausible interpretation of man-made scenes. E.g. In [2] Boulanger et al. have extracted camera parameters and generated a simple 3D model from an image. They have used straight lines and three vanishing points to produce this result.

Spatial layout reconstruction was shown in [10–12, 24, 23]. The main base of those works is the Manhattan world assumption. Gradient features have been used to produce 3D boxes for big objects like beds in [11, 12]. In [24, 23] *orientation map* was used for this purpose. We applied this orientation map as a starting point of our algorithm.

It is important to get vanishing points for scenes of Manhattan type. In [6] Denis et al. have shown that a better way for extracting vanishing points is to use localized edges of objects rather than gradient maps. In [31] a fast algorithm was presented to extract three vanishing points. To improve this result an Expectation–maximization algorithm was used. In [33] vanishing points recovering was expanded to high-level geometric primitives like horizon and zenith. In addition, they have presented framework which can be used for images that are not in agreement with the Manhattan world assumption.

Recovering 3D structure of urban scenes can be done by using 3D model which consists of vertical walls and ground plane, where ground-vertical boundary is a continuous polyline. This was shown in [1].

Some works have shown that understanding the type of a surface can produce a good 3D model. Labeling geometric classes for image surfaces was shown in [15–17]. In [13, 18] those works were combined to produce 3D models of scenes. They have used various cues like color, texture gradient and projection information (vanishing points). In [28, 29] surface connectivity and image depth was used to generate plausible 3D model. All those works use this information to train a surface classifier on a training set of images. Moreover, they have shown that the use of superpixels instead of pixels significantly improves algorithm speed and surface recovering. We combined our approach with [17] to improve our result and to show that the proposed algorithm can be easily combined with other methods.

The significance of occlusion boundaries was shown in [13, 18, 19]. In [8] occluded surfaces were restored from both indoor and outdoor images. Those results can improve 3D scene reconstruction.

In [30] depth ordered planes were used. Those planes are generated using vanishing points and relative depth cues. Spatial layout of cluttered rooms was successfully applied for rendering synthetic objects into photographs in [20]. An interesting observation has been made in [9] that the world can be presented as a set of 3D blocks. Besides, it uses density classes to estimate each block, such as "light" for trees and "high-density" for buildings.

As we see, there are various methods to get a three-dimensional model from one image. For a wide range of image types training algorithms have shown the highest percentage of getting correct surfaces, but we focus only on a man-made environment which is mostly in agreement with the Manhattan world assumption. As a result, we decided to use this assumption as a base of our work.

3 Overview of Approach

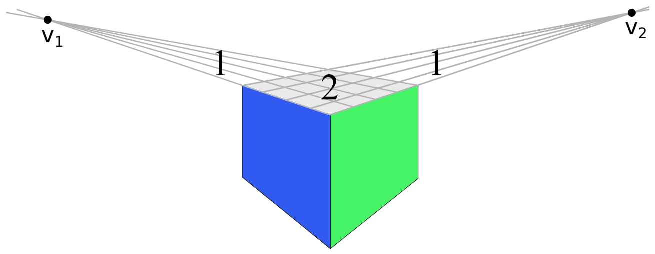


Fig. 2. We recovered two planes of a figure, but we do not have the third (top surface). Numbers indicate votes for surfaces. v_1 and v_2 are two horizontal vanishing points.

In the paper we introduce *surface prediction* approach. By surface prediction here, we understand how recovered surfaces can be used to predict their neighbor planes. As we can see on Fig. 2 there can be a cube, but one surface is missing (top). Moreover, there are can be two missing surfaces instead of one. What can we do with that? Assume, that if we know one object surface or surface of neighbor object, then we can find other surfaces. We say that each plane or surface votes for neighbor planes. On Fig. 2 top surface receives 2 votes. As we will see, this simple assumption helps us to recover main surfaces of a scene.

The main difference of our method is that it restores most of the planes of a scene without building hypothesis for objects. We do not use only one bearing surface (like ground in [1]). Furthermore, we recover objects from different levels of a scene, because all surfaces are generated iteratively bearing on each other. Moreover, we use the Manhattan world assumption as the base for recovering scene surfaces. Consequently, our approach works on both indoor and outdoor man-made scenes. We do not generate object fitting, but we think that it can be easily obtained if blocks world assumption is used.

First, in our work we automatically find vanishing points and the focal length. Vanishing points are computed using straight lines of objects. To make computation easier we state that the vertical vanishing point is the point at infinity. For the focal length we assume that the principal point is located in the center of an image. Then, we calculate the focal length from vanishing points. This information is needed to recover 3D surfaces. In the second part of this paper we present our new method for surface recovering. Its origin is based on *orientation map* (*omap*) [24]. We collect votes (predicted surfaces) from each surface from *omap* and choose the most appropriate result. To improve our result we use surface approximation based on the Manhattan world assumption. Last, we present experimental validation on two datasets. To evaluate our algorithm we have used Delage et al. [5] and York Urban [6] datasets. Most of the images here are in agreement with the Manhattan world assumption. For surface testing those images were manually labeled with LabelMe tool [27].

4 Finding Camera Parameters

For finding vanishing points we follow [31, 26] approaches with some modifications. Initially, we extract object edges with Canny edge detector. Next, we use the following steps to extract line segments from obtained edge map. First, to speed up computation, we remove all junctions on received binary image. Then, we extract connected components using flood fill algorithm. As the next step we get line segments with Kovesi Matlab toolbox [21]. Each edge of an edge map is divided into straight line segments by splitting it when standard deviation of pixels is larger than one pixel.

Next, we use j-linkage algorithm [32] to get vanishing points from obtained lines. We need a consistency measure to link lines to theirs vanishing points. If we use the distance between a vanishing point and a line, for only a small deviation of a line from a point, we can get large distances, e.g. vanishing points at infinite. To avoid that, we use approximation from [31], it represents the distance between edge ending and a line through a vanishing point and the edge center. You can see obtained lines on Fig. 3a.

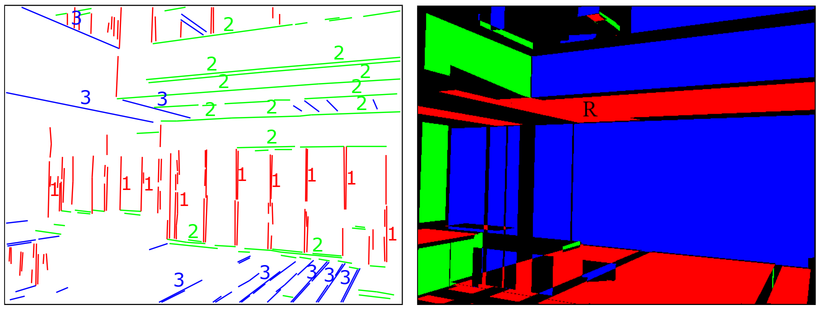
After this step, to get three orthogonal vanishing points we use the Manhattan world assumption and the guess that the vertical vanishing point is the point at infinite. Once vanishing points are found on an image, we get the focal length of the camera by finding a focal length that makes angles 90 degrees with vectors to vanishing points and assuming that principal point location is in the center of an image [4].

4.1 Finding Surface Coordinates

Now, to compute object surfaces we need only a complete orientation map which we obtain in the next part of this article. When we have it, we reconstruct a 3D scene only to the scale factor, because we do not know original metric of a scene. We iteratively compute coordinates for each surface and its neighbors. To minimize an error, we choose surfaces with maximum area and neighbors, then compute them first. Basing on their received coordinates we compute others. Due to the iterative process of 3D model recovering we restore objects that lie on different levels of a scene.

5 Orientation Map

First, to recover the orientation of surfaces we need a starting point. We get it from [24, 23]. We use an orientation map which is described in those articles. We give here only a brief description. An example of orientation map can be seen on Fig. 3b. This map indicates the orientation of some region which is produced by two vanishing points. The main idea is that if you have a region which consists of two groups of lines referred to two different vanishing points, then those lines can produce a surface. On Fig. 3a we can see that the horizontal planes consist of green and blue lines (marked as 2 and 3 respectively), while vertical are from



(a) (b)

Fig. 3. (a) Line segments from Section 4 which are labeled with respect to their vanishing points (only long lines have labels). (b) Initial orientation map from [24]. Color represents orientation of regions and lines.

red and blue (1 and 3) or red and green (1 and 2) lines. Each line segment is extended until it abuts against a line (stopping parameter) which is orthogonal to the surface this line segment generates. This assumption is correct, because a line cannot lie on a plane which is orthogonal to that line. Detailed explanation and formal description can be seen in [24].

6 Surface Recovering

There are some problems with original orientation map (*omap*) from Section 5: 1) incorrect surfaces, which are formed due to the wrong lines, 2) *omap* approach cannot restore a surface without lines, as you can see there are a lot of free space on Fig. 3b, 3) in addition to the previous point, *omap* method does not restore occluded surfaces, e.g. walls of a building behind a tree.

To overcome problems that were mentioned in the previous paragraph we propose our new *surface prediction* (*SP*) approach. First, we remove all surfaces whose area is lower than the threshold. In that case we do not loose much information, because we will fill this free space during next steps. Second, our algorithm recovers missing planes by predicting them using surfaces that were restored in Section 5. Finally, our method reconstruct occluded surfaces. For example, if one part of a building wall was occluded, then a recovered segment of the wall will vote for the occluded part.

Our method extends the idea of line continuation from [24] to plane continuation. Moreover, we introduce a voting scheme to get final orientation of surfaces. Next, we propose a surface approximation step to further improve the result. We have found that it is especially useful for building 3D models.

6.1 Surface Prediction

Suppose, we know one surface of an object, in our case it is an initial orientation map from Section 5, then we can recover others. For example, if we reconstruct

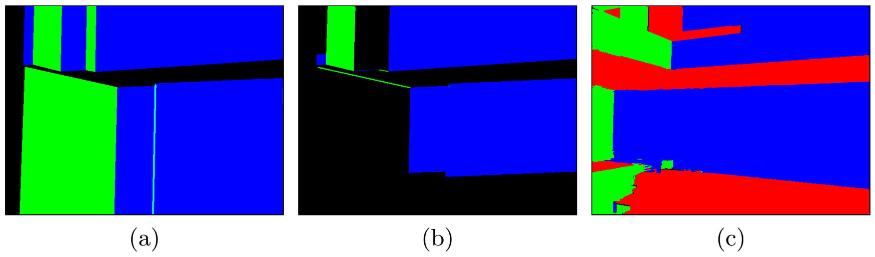


Fig. 4. The first two figures show predicted surfaces for one red plane from Fig. 3b (marked as R). (a) shows surfaces without a stopping parameter, on (b) for the stopping parameter we use neighbor planes from Section 5, on (c) you can see the final result which is obtained after adding together all predicted surfaces.

only one cube facet, then we can say that the cube can have other facets or it lies on a different object (there are no flying objects on an image or they are supposed to be a noise). Following this assumption we define a position and an orientation of other surfaces, see Fig. 4 for an example of predicted surfaces for a red plane. Basically, we continue boundaries of planes from Section 5 to vanishing points. Then, we calculate votes for each orientation from all predicted surfaces and receive full orientation map.

The formal description of the method: let $S_x = \{s_{x,1}, s_{x,2}, ..., s_{x,n_x}\}$ be the set of surfaces (or planes), which are orthogonal to the x, where $x \in \{1, 2, 3\}$ denotes one of the three orientations and n_x is the number of planes with corresponding orientation. Remember that in Manhattan world, we have only 3 vanishing points (2 horizontal, 1 vertical) and each surface is produced by 2 of them. Consequently, we have only 3 orientations for surfaces. Next, the set of points which are produced by Ramer–Douglas–Peucker algorithm [25, 7] for each surface boundary is:

$$P_{x,n} = \{p_{x,1,1}, ..., p_{x,n_x,m_x}\} \tag{1}$$

where m_x is a number of points which are obtained by this approximation and $n \in \{1 : n_x\}, m \in \{1 : m_x\}$.

Algorithm 1 *Predict neighbors for surface*

```
Require: Points from approximation $P_{x,n}$, vanishing points $v_d$, where $d \in \{1,2,3\}$.
Ensure: Orientation map for each surface $O_{x,n}$.
1: $p \leftarrow P_{x,n}[1]$
2: for $i \leftarrow 2$ to $m_x$ do
3:     $g \leftarrow linesegment(p, P_{x,n}[i])$, a line segment between two points.
4:     $O_{x,n} \leftarrow O_{x,n} + getomap(g, v_d)$ ($getomap$ returns orientation map $\widehat{O}$ from Section 5 for $g$ and $v_d$)
5:     $p \leftarrow P_{x,n}[i]$
6: end for
```

After that, we get orientation maps $O_{x,n}$ by Algorithm 1, you can see the example for the vertical vanishing point on Fig. 4. It is important that Fig. 4b presents the result which uses the same idea for a line segment continuation as the one in Section 5, the only difference is that continuation stops when a line segment abuts a surface, not a line. Then, we sum up all surfaces:

$$O_x = \sum_{1 \leq n \leq n_x} O_{x,n} \tag{2}$$

This addition represents voting for surface prediction. During algorithm evaluation we realized that horizontal surfaces performed poorly compared to vertical, due to the fact that ground and floor parts of an image usually contain many incorrect surfaces from original *omap*. Therefore, we decided to use surfaces from [17] for horizontal surfaces in Eq. (2). In the testing section we show results with and without this information. The final equation for the whole orientation map O calculates as:

$$O = \widehat{O} \vee max_x(O_x) \tag{3}$$

where $x \in \{1, 2, 3\}$ and $\widehat{O}$ is an orientation map from Section 5, the function max_x denotes orientation of a pixel on the base of votes for each orientation, which we obtain when we get the sum from Eq. (2). The result can be seen in Fig. 4c.

6.2 Approximation of Surface Edges

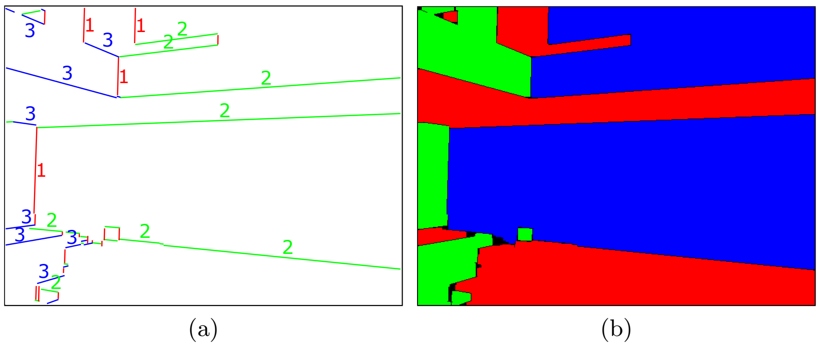


Fig. 5. Smoothed boundaries and a final orientation map. Boundaries (a) divided into three groups of line segments. The final orientation map (b).

Orientation map which is obtained in the previous part, see Fig. 4c, is not good. It has a lot of noise among edges of different channels. Due to the iterative process of building a 3D model, see Section 4.1, this may lead to gross errors at those surfaces that are computed in the last instance. Thus, we need smooth boundaries for surfaces. To get a better result we must use lines which belong to vanishing points for those boundaries. We propose the following method to do that.

First of all, we use erosion for each channel of orientation map. Next, we add all channels together and get a binary image (after voting, channels do not intersect)

$$B = \sum_{1 \leq x \leq 3} O_x \tag{4}$$

Then, we reverse image $\neg B$ and use thinning operation on it. We break obtained edge map into line segments as we did in Section 4. After that, we link all line segments to their vanishing points through consistency measure, see Section 4. We need to align each line segment to its vanishing point. For $g = linesegment(t_1, t_2)$, where t_1 and t_2 are endpoints of g, let m be midpoint of g and l be the line which passes through m and v (vanishing point), then aligned line segment is:

$$r = (proj(t_1, l), proj(t_2, l)) \tag{5}$$

where $proj(t, l)$ function returns the projection point of a point t on a line l. The result can be seen on Fig. 5a.

Obtained line segments are added together and imposed to an orientation map dividing it to distinct components. After noise removing we get smoothed orientation map, see Fig. 5b.

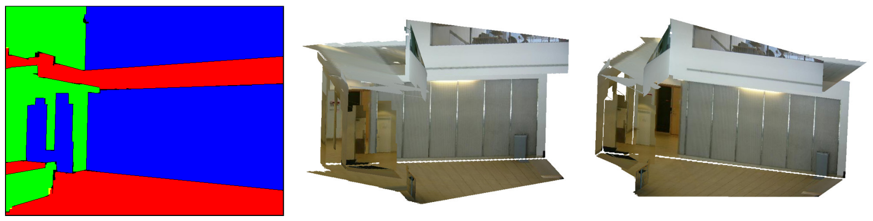


Fig. 6. From left to right: orientation map after multiscaling, two projections of the reconstructed 3D model.

To get a better result we can use different scales of an image or *multiscaling*. However, the algorithm execution time grows a lot. The idea is pretty simple: we choose several scales of an image, compute an orientation map for each scale, then we add all orientation maps from all scales together and use voting to choose the best orientation for each pixel. The result can be seen in Fig. 6.

7 Experiments and Results

We have tested our algorithm on Delage et al. [5] dataset (48 images) of indoor scenes and we have also labeled its images with LabelMe tool [27]. In particular,

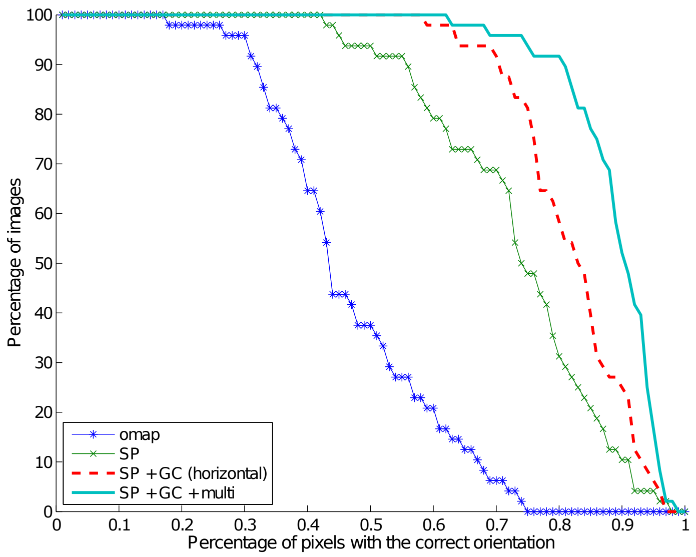


Fig. 7. Percentage of pixels with the correct orientation. Compared methods: *omap* — raw orientation map from [24], *SP* — Surface Prediction approach (Section 6), *GC* (Geometric Context) — [17] for all channels, *GC*(*horizontal*) — [17] to vote for horizontal surfaces (red channel on images), *multi* — *multiscaling*, see Section 6.2.

we have manually marked ground truth orientation for each surface formed by vanishing points. You can see the percentage of pixels that have the correct orientation on Fig. 7.

Here we present a comparison of the average percentage of correct pixels among all images in Delage et al. dataset. First, we have tested our *SP* method and got 73.2%. With multiscaling this result was improved to 73.6%. Multiscaling for *SP* added less than a percent for an output result, so it is not so useful due to the additional time cost. We applied it only to the final result to slightly improve it and to obtain better 3D models (boundaries which are produced after multiscaling are much smoother). After that, to improve the result we combined *SP* with [17] (*Geometric Context*) approach for horizontal surfaces *GC*(*horizontal*) and in that case this combination outperformed method that was shown in [24] (81.9% versus 80%). Next, as reported in [24], they got 87% for [14] approach on Delage et al. dataset, we have tested this with *GC* for all orientations and got nearly the same result. Interestingly, $GC + multiscaling$ produced worse result (86%) than *GC* itself. Finally, we combined *GC* for all surfaces with our Surface Prediction method and got 88.9%, that outperformed *GC* for indoor scenes. The average percentage for each method can be seen in Table 1.

For Delage et al. dataset the number of images with $\geq 50\%$ of correct surfaces is near 94% for *SP* and 100% for its combinations with *GC*.

Table 1. Results that were obtained on Delage et al. [5] (second column) and York Urban [6] (third column) datasets. The first column shows used methods for surface recovering, two last columns denote an average percentage of correct pixels for surface orientation.

Method components	Avg Delage	Avg York
$omap$	46.7	51.8
SP	73.2	73
Lee et al. [24]	80	–
$SP + GC(horizontal)$	81.9	81.1
GC [17, 14]	87	–
$SP + GC + multi$	88.9	–

We have also tested our approach on York Urban [6] dataset (102 images), which consists of indoor and outdoor urban scenes. On this dataset we got 81.1% correct surfaces in average with SP+$GC(horizontal)$ params, which corresponds to the average percentage of Delage et al. dataset. This means that our algorithm is robust to different types of Manhattan world scenes. We do not provide a result for [24] approach, because it is suitable only for indoor scenes. In addition, original GC cannot be easily applied for images of outdoor scenes for vertical surfaces which belong to vanishing points, so we did not use it in our experiments on York Urban dataset.

Eventually, 3D models have been tested by four people who were not a part of our project. They have tested if obtained 3D model was plausible interpretation or not, we have got around 55% correct models for all images.

Some examples of 3D scene reconstruction can be seen on Fig. 8. Typical errors are shown on Fig. 9 and Fig. 10. For each photo there are four images: original image, our final orientation map, two images of an obtained 3D model. It is worth noting that there are two main types of errors:

1. Wrong output 3D model due to the large amount of noise on an image Fig. 9.
2. Irregular surface position while the orientation of a surface is obtained correctly Fig. 10.

8 Conclusion

In this article, we introduced a new method for 3D surface reconstruction. It was shown that it works on both urban scenes and indoor apartments. Moreover, we addressed the problem of restoring objects that are not connected to only one base plane (ground or floor). By evaluation on popular image datasets, we showed that the presented algorithm restores plausible 3D models from different images. In addition, due to the voting process this approach is easily combined with other algorithms and obtained results outperformed several state-of-the-art techniques. The proposed concept of *surface prediction* can be useful for many computer vision applications as well.

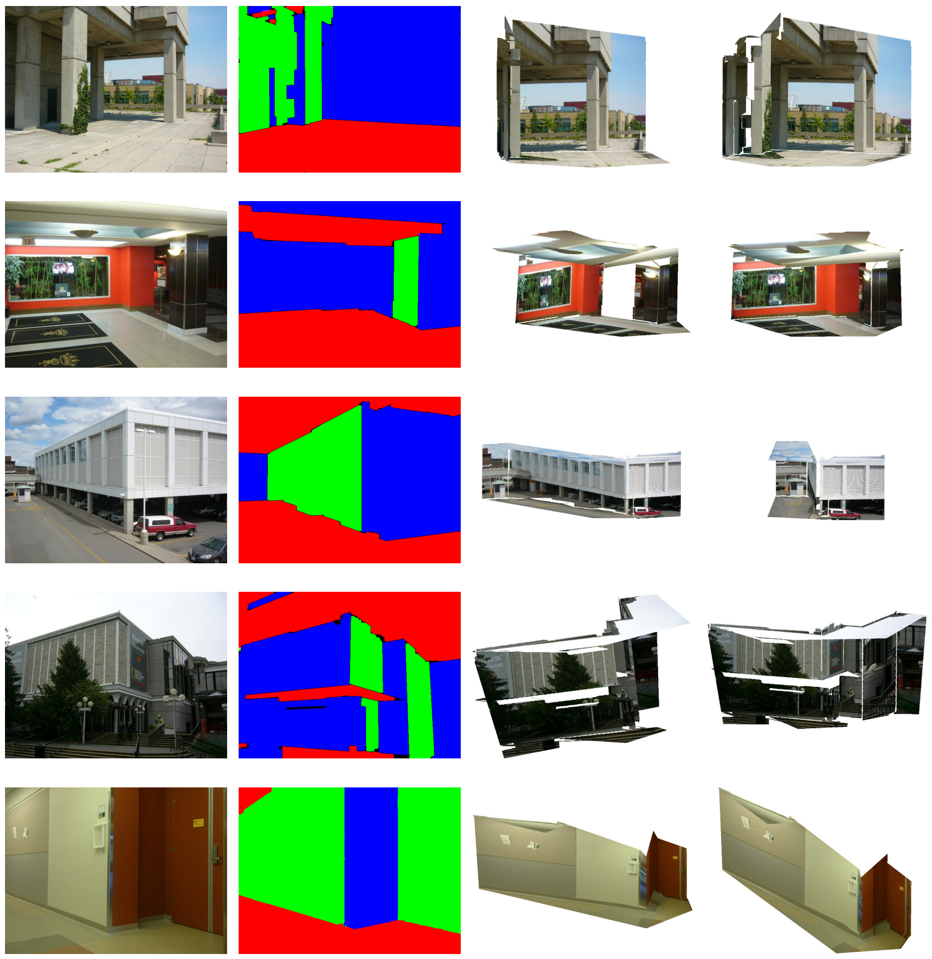


Fig. 8. Scene reconstruction examples. From left to right: original image, obtained orientation map, two views of a 3D model.

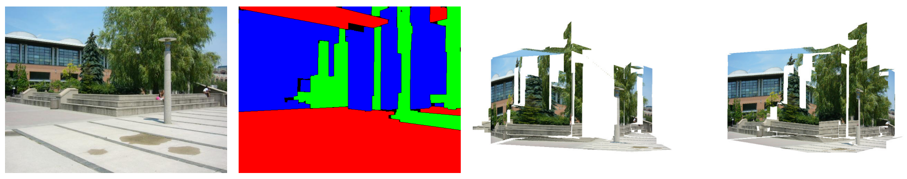


Fig. 9. An example of failure in the presence of significant noise (trees).

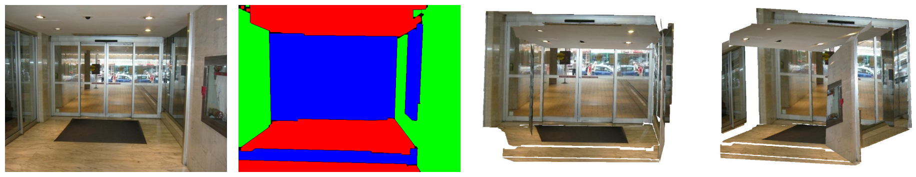


Fig. 10. Another example of errors when surface positions are irregular while most of surface orientation is obtained correctly.

Acknowledgements. To Jiri Matas for valuable comments on the paper and to Evgeny Stolov for the help during the research.

References

1. Barinova, O., Konushin, V., Yakubenko, A., Lee, K., Lim, H., Konushin, A.: Fast automatic single-view 3-d reconstruction of urban scenes. Computer Vision–ECCV 2008 (2008) 100–113
2. Boulanger, K., Bouatouch, K., Pattanaik, S.: Atip: A tool for 3d navigation inside a single image with automatic camera calibration. EG UK theory and practice of computer graphics **15** (2006)
3. Coughlan, J.M., Yuille, A.L.: Manhattan world: Compass direction from a single image by bayesian inference. In: Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. Volume 2., IEEE (1999) 941–947
4. Criminisi, A., Reid, I., Zisserman, A.: Single view metrology. International Journal of Computer Vision **40** (2000) 123–148
5. Delage, E., Lee, H., Ng, A.: A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. Computer Vision and Pattern Recognition **2** (2006) 2418–2428
6. Denis, P., Elder, J.H., Estrada, F.J.: Efficient edge-based methods for estimating manhattan frames in urban imagery. Computer Vision–ECCV 2008 (2008) 197–210
7. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization **10** (1973) 112–122
8. Guo, R., Hoiem, D.: Beyond the line of sight: labeling the underlying surfaces. In: Computer Vision–ECCV 2012. Springer (2012) 761–774
9. Gupta, A., Efros, A.A., Hebert, M.: Blocks world revisited: Image understanding using qualitative geometry and mechanics. Computer Vision–ECCV 2010 (2010) 482–496
10. Hedau, V., Hoiem, D., Forsyth, D.: Recovering the spatial layout of cluttered rooms. In: Computer vision, 2009 IEEE 12th international conference on, IEEE (2009) 1849–1856
11. Hedau, V., Hoiem, D., Forsyth, D.: Thinking inside the box: Using appearance models and context based on room geometry. Computer Vision–ECCV 2010 (2010) 224–237
12. Hedau, V., Hoiem, D., Forsyth, D.: Recovering free space of indoor scenes from a single image. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 2807–2814

13. Hoiem, D.: Seeing the world behind the image: Spatial layout for three-dimensional scene understanding. (2007)
14. Hoiem, D., Efros, A., Hebert, M.: Putting objects in perspective. International Journal of Computer Vision (2008)
15. Hoiem, D., Efros, A.A., Hebert, M.: Automatic photo pop-up. In: ACM Transactions on Graphics (TOG). Volume 24., ACM (2005) 577–584
16. Hoiem, D., Efros, A.A., Hebert, M.: Geometric context from a single image. In: Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. Volume 1., IEEE (2005) 654–661
17. Hoiem, D., Efros, A.A., Hebert, M.: Recovering surface layout from an image. International Journal of Computer Vision **75** (2007) 151–172
18. Hoiem, D., Efros, A.A., Hebert, M.: Closing the loop in scene interpretation. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE (2008) 1–8
19. Hoiem, D., Efros, A.A., Hebert, M.: Recovering occlusion boundaries from an image. International Journal of Computer Vision **91** (2011) 328–346
20. Karsch, K., Hedau, V., Forsyth, D., Hoiem, D.: Rendering synthetic objects into legacy photographs. ACM Transactions on Graphics (TOG) **30** (2011) 157
21. Kovesi, P.D.: Matlab and octave functions for computer vision and image processing. Online: http://www.csse.uwa.edu.au/ pk/research/matlabfns/ (2000)
22. Lalonde, J.F., Hoiem, D., Efros, A.A., Rother, C., Winn, J., Criminisi, A.: Photo clip art. In: ACM Transactions on Graphics (TOG). Volume 26., ACM (2007) 3
23. Lee, D.C., Gupta, A., Hebert, M., Kanade, T.: Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In: NIPS. Volume 1., Vancouver, BC (2010) 3
24. Lee, D.C., Hebert, M., Kanade, T.: Geometric reasoning for single image structure recovery. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE (2009) 2136–2143
25. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing **1** (1972) 244–256
26. Rother, C.: A new approach to vanishing point detection in architectural environments. Image and Vision Computing **20** (2002) 647–655
27. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: Labelme: a database and web-based tool for image annotation. International journal of computer vision **77** (2008) 157–173
28. Saxena, A., Chung, S.H., Ng, A.Y.: 3-d depth reconstruction from a single still image. International Journal of Computer Vision **76** (2008) 53–69
29. Saxena, A., Sun, M., Ng, A.Y.: Make3d: Learning 3d scene structure from a single still image. Pattern Analysis and Machine Intelligence, IEEE Transactions on **31** (2009) 824–840
30. Stella, X.Y., Zhang, H., Malik, J.: Inferring spatial layout from a single image via depth-ordered grouping. In: In CVPR Workshop. (2008)
31. Tardif, J.P.: Non-iterative approach for fast and accurate vanishing point detection. In: Computer Vision, 2009 IEEE 12th International Conference on, IEEE (2009) 1250–1257
32. Toldo, R., Fusiello, A.: Robust multiple structures estimation with j-linkage. Computer Vision–ECCV 2008 (2008) 537–547
33. Tretyak, E., Barinova, O., Kohli, P., Lempitsky, V.: Geometric image parsing in man-made environments. International journal of computer vision **97** (2012) 305–321