Automatic Natural and Man-made Scene Differentiation Using Perspective Geometrical Properties of the Scenes

J. Kovač*, P. Peer*, F. Solina*
*University of Ljubljana
Faculty of Computer and Information Science
Tržaška cesta 25, Ljubljana, 1000, Slovenia
E-mail: jure.kovac@email.si

Keywords: automathic scene differentiation, vanishing points, vanishing lines

Abstract – In this paper we are trying to establish a framework for natural and man-made scene differentiation based on perspective geometrical properties of the scenes. Although, we have not jet achieved expected results with built classifier, evidence of distinguishing attributes exists and some of them are introduced in this paper.

1. Introduction

In this work we try to establish a framework for natural and man-made scene differentiation based on perspective geometrical properties of scenes. The idea is based on two geometrical properties that hold when scene follows rules of perspective geometry [1], [5] (Fig. 1):

- Lines in the image that correspond to parallel lines in 3D scene intersect in the same point. This point is called the **vanishing point**.
- Groups of parallel lines which lie in the same or on parallel planes intersect on the same line. The line is called the **vanishing line**. Vanishing line of ground plane is called the **horizont**.

In general, images of both types of scenes, man-made and natural, obey the rules of perspective. However, in man-made scenes, observed perspective properties are expressed to much greater extent. Man-made scenes tend to contain significantly larger amount of parallel lines, perpendicular corners and parallel planes which are all due to the way man builds and arranges objects. Everything seems to have some sort of order, at least when comparing to natural scenes. This order makes man-made scenes much more appropriate for vanishing point and vanishing line detection, which should reflect in certain properties of these two perspective elements. Therefore, we try to distinguish among the two types of scenes by comparing the differences in properties of detected vanishing points and vanishing lines in images of both types.

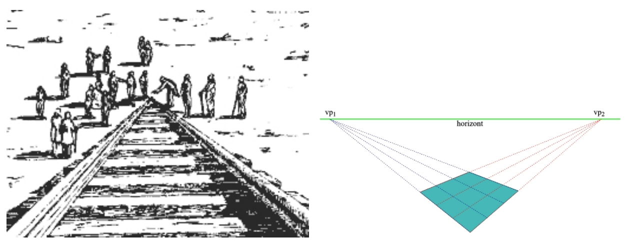

Figure 1: Picture on the left shows an example of the vanishing point and picture on the right shows an example of the vanishing line.

2. Methodology

The proposed algorithm is based mostly on Hough transform [4] and uses some other standard computer vision tools.

2.1 Preprocessing

First we begin by some standard preprocessing steps needed to generate the input image for Hough transform and later for second part of the algorithm:

Resizing: Input image is resized either to 320×200 or 200×320 pixel array depending on the image orientation.

Smoothing: Image is convolved by 9×9 Gaussian kernel [4].

Edge detection: Edges are detected using the Canny edge detector [4].

Such image is then passed to probabilistic Hough transform algorithm provided by OpenCV [2]. The parameters passed to Hough transform are the minimum line length of 15 pixels and maximum gap between lines, also 15. Result is the list of all line segments found on input image, which is actually the input for the other part of the algorithm that tries to look for perspective properties.

2.2 Finding Vanishing Points

The basic element of perspective properties is afore mentioned vanishing point. Therefore, we begin by looking for vanishing point candidates. At the beginning we state, that candidates are all points found by intersecting provided line segments among each other. At this point we already removed all intersections that lie between the endpoints of intersecting segments, as these can not be vanishing points [3].

We than use voting to evaluate the candidates based on the length and number of lines that support that candidate. How much some segment supports some particular candidate is also considered. We use the following voting function ([3]):

$$\mathrm{vote}(\mathbf{v}) = \sum_{\mathbf{d}} \left(u_1 \left(1 - \frac{\mathrm{dist}(\mathbf{v},\mathbf{d})}{t_v} \right) + u_2 \left(\frac{\bar{\mathbf{d}}}{\max \mathbf{d}} \right) \right). \tag{1}$$

$\mathbf{v}$ is a VP candidate, $\mathbf{d}$ is line segment that votes for this candidate. $\mathrm{dist}(\mathbf{v},\mathbf{d})$ returns the angle between line segment $\mathbf{d}$ and line through points $\mathbf{v} \times \mathbf{m_d}$, where $\mathbf{m_d}$ is the center of line segment $\mathbf{d}$ (Fig. 2). Only those segments that have smaller angle than some treshold t_v (e.g. 5° or 10°) vote for candidate $\mathbf{v}$. $\bar{\mathbf{d}}$ is the segment length, $\max \mathbf{d}$ is the length of the longest segment and u_1 and u_2 are additional weights (e.g. 0.3 and 0.7). In the list of VP candidates we than put triplets $(\mathrm{vote}(\mathbf{v}), \mathbf{v}, \mathrm{voters}(\mathbf{v}))$, where $\mathrm{voters}(\mathbf{v})$ is a list of segments that voted for $\mathbf{v}$. The list is then sorted descending by $\mathrm{vote}(\mathbf{v})$.

2.3 Vanishing Line Segmentation

Thus, with weighted intersections we now perform the second Hough transform to find vanishing lines, which also considers calculated weights and has some additional properties. Accumulator space consists of two dimensions that correspond to parametric line parameters θ and r. We want that all intersections that are close to the line (with respect to parameter r), vote for the same line, so accumulator field is not as dense in r dimension as in θ. One accumulator field corresponds to interval $[0, 150]$ pixels with respect to r, while θ uses step 1°.

When voting for a line, the weight of the intersection obtained in section 2.2 is added to the accumulator value. Furthermore, we wanted to encourage those lines that have intersections spread along the line as opposed to the lines, that have all intersections (that voted for this line) close together. So, we first check whether some other intersection in the vicinity of the current has already voted for this line and if so, we diminish it's vote by some factor (e.g. 0.1). This is done when the distance between intersections is less than some treshold (e.g. 150 pixels).

We then filter the accumulator field with N-neighbourhood filter (e.g. 10×1 filter would keep the maximum accumulator field value in angle range of 10° and put all other accumulator fields in range to 0).

2.4 Calculated Attributes

From the vanishing lines found in the previous section we now try to obtain some useful attributes that might later

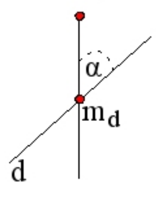

Figure 2: The figure shows an angle between line segment $\mathbf{d}$ and line through points $\mathbf{v}$ and $\mathbf{m_d}$ returned by function dist.

serve as part of the classifier. We calculate the following quantities:

Average residual sum of squares $\bar{S}$: For every VP (x, y) that voted for a line (θ, r) we compute least square difference between the point and fitted line. From parametric line equations:

$$\begin{aligned} x\cos(\theta) + y\sin(\theta) &= r \\ r\cos(\theta) + t\sin(\theta) &= x \\ r\sin(\theta) + t\cos(\theta) &= y \end{aligned} \tag{2}$$

we first compute parameter t:

$$\begin{aligned} t &= (y - r * \sin(\theta))/\cos(\theta); && \text{if } |\sin(\theta)| < 0.5 \\ t &= (-x + r * \cos(\theta))/\sin(\theta); && \text{else,} \end{aligned}$$

then calculate (x', y') according to equation (2), obtain residual from:

$$\begin{aligned} R &= (x' - x)^2 && \text{if } |\sin(\theta)| < 0.5 \\ R &= (y' - y)^2 && \text{else,} \end{aligned}$$

and finally derive the sum $\bar{S}$ by averaging the residuals:

$$\bar{S} = \frac{\sum_{i=0}^{N} R}{N}.$$

This attribute tells us how many intersections that voted for the line fit to the line. The smaller the value, the better.

Average distance: This attribute is very similar to the previous one, except that this time we observe the actual average distance (in pixels) of intersections from the line.

Line strength: The sum of voters that voted for the line.

Line Vote: This is the actual vote of the line as computed by the Hough transform described in section 2.3.

Average intersection weight: This attribute was considered on both, overall image and specific line level.

Standard deviation of weight: Standard deviation with respect to previous attribute.

First three attributes were used in three different ways. First, we computed them by considering **intersection weight** obtained in section 2.2. This was done by extending the list of intersections that voted for the line by duplicating them according to their weights (e.g. intersection with vote 3 was added three times). Second, the attributes were calculated by considering each intersection only once (**one-time**) regardless of its weight. Finally, these attributes were calculated by considering **diminished weights** as described in section 2.3. Line vote was always calculated as described in section 2.3.

3. Preliminary Results

The images of intersections that were obtained by described procedure show some promising properties, which might be used for building a classifier. Solely by observing the result images it can be seen that quite some distinctions are present when comparing man-made and natural scenes,

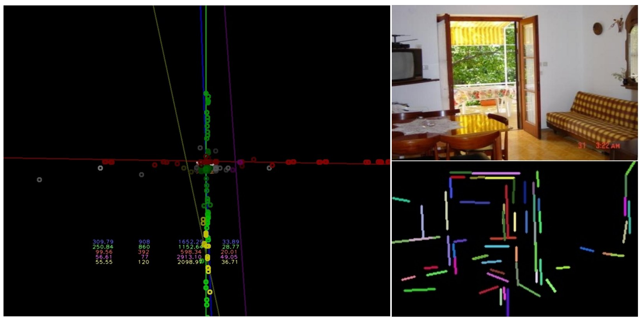
Figure 3: The figure shows an example image of a man-made scene processed as described in the text. On the right is the input image (top) and detected line segments in it (bottom), while on the left is an image of intersections with first five vanishing lines.

however, defining those differences seems to be a bit of a hassle.

The example images obtained by described algorithm are shown in figures Fig. 3 and Fig. 4. The right side contains input image and image of detected line segments. The input image was reduced in size and put into the center of left side image, which also shows all line intersections marked by circles of different color. The color corresponds to the color of vanishing line that intersections voted for. The brighter the intersection color, the bigger is its weight. The first five obtained vanishing lines are shown. Also, the brightness of the line corresponds to the total line vote. Grey color marks those intersections that did not vote for any lines shown in the image. The calculated attributes for both examples are shown in tables 1 and 2.

Another test was performed considering 38 man-made and 50 natural scene images. Some statistics were computed on these images. The average intersection vote for man-made scenes was 3.7131 with standard deviation of 1.6016 and the average vote for natural scenes was 3.3660 with st. dev. of 1.2831. Also, the attributes of all 88 images were put into Weka's "Random Tree" algorithm [6] and the results show only a slight improvement of apriori probability (see table 3).

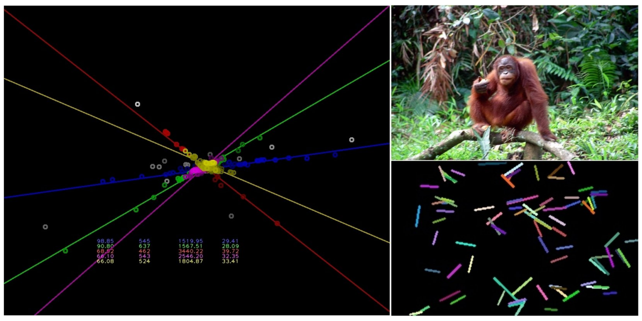
Figure 4: The figure shows an example image of a natural scene processed with the described algorithm. On the right is the input image (top) and detected line segments in it (bottom), while on the left is an image of intersections with first five vanishing lines.

4. Discussion

Although we did not achieve very good results with the classification jet, the preliminary results still look somehow promising. The following observations hold for most of the images and comply with hypotheses stated in the introduction:

- The image of detected line segments seems to preserve larger amount of contents information within the man-made scenes than within the natural scenes. From lines image of man-made scene it is easier to guess what is on the original image.
- In the natural scenes usually only one vanishing line shows intersections arranged through the whole line length, while on the man-made scenes two such lines exists. This supports the proposition that man-made scenes more evidently reveal vanishing lines, while in the natural scenes only the horizont is evident.
- In the man-made scenes the vanishing lines show more compact (dense) arrangement of intersections along the whole line.
- In the natural scenes the density of intersections is much larger in the center of the image than in the man-made scenes.
- In the man-made scenes the important (the brighter) intersections seem to lie far away from the image center, which more reliably indicates the positions of the vanishing points. In case of close, man-made objects, parallel lines should intersect far in the distance. Although this also depends on camera external parameters (angle, position), it seems that most of man-made scene images are taken in certain "right" way.
- The vanishing lines in the man-made scenes are more aligned to vertical and horizontal direction, while in the natural scenes these angle seems to be more random. As in the previous case this also depends on external camera parameters.
- The average image vote is larger in the man-made then in the natural scenes. This supports the hypothesis that vanishing points are more supported in the man-made scenes, since they contain more parallel lines lying on parallel planes than it is the case in the natural scenes.
- The vanishing line vote is generally higher in the man-made scenes.

5. Conclusion

With all these observations and findings we can see that the attributes that distinguish those two types of scenes exist. Although we probably don't know all of them jet and also cannot tell which of them are the most important, we are confident that based on this information a useful classifier for the man-made and the natural scene differentiation can be found, so we will further pursue this idea.

gavgvt	gvtstd
4.0044	2.4166

θ	r	avgvt	vtstd	lvote	lenx	sumres	avgres	avgdst
178	-150	4.0384	2.5708	309.7933	226	351896.3902	1557.0637	32.5077
0	150	3.8258	2.4526	250.8435	226	319301.0000	1412.8363	31.3496
91	0	3.4685	1.0723	99.5630	112	65393.4119	583.8697	19.5501
176	-600	6.8436	2.4832	56.6059	11	33626.5794	3056.9618	50.8035
168	150	6.9786	2.2226	55.5512	17	37991.5240	2234.7955	38.7819
	lenxW	**sumresW**	**avgresW**	**avgdstW**	**lenxC**	**sumresC**	**avgresC**	**avgdstC**
	908	1500282.8	1652.29	33.89	285	489014.05	1715.84	35.7
	860	991272	1152.64	28.77	225	179202	796.45	24.19
	392	234551.03	598.34	20.01	81	100051.23	1235.2	30.37
	77	224308.68	2913.1	49.05	58	150735.22	2598.88	44.94
	120	251876.07	2098.97	36.71	59	131579.71	2230.16	39.66

Table 1: The table shows derived attributes of example in figure 3. On the top, global average vote and standard deviation of image intersections ("gavgvt" and "gvtstd") are given. θ and r are line parameters, "avgvt" is line's average intersection vote, "vtstd" is line's intersection standard deviation and "lvote" is line's vote. The rest are "lenx": the line strengt, "sumres": residual sum, "avgres": average residual sum and "avgdst": average distance. The last four are repeated three times, for three different ways of derivations; without suffix: one-time, with suffix "W": intersection weights and with suffix "C": diminished weights (see section 2.4 for description).

gavgvt	gvtstd
2.3608	0.682

θ	r	avgvt	vtstd	lvote	lenx	sumres	avgres	avgdst
82	150	2.33	0.7	98.85	233	384461.65	1650.05	31.73
60	150	2.45	0.69	90.8	257	441252.31	1716.94	29.48
128	0	2.28	0.61	68.52	202	665732.84	3295.71	38.6
49	150	2.47	0.72	66.1	217	567326.73	2614.41	32.77
113	0	2.26	0.64	6.08	229	380002.78	1659.4	31.63
	lenxW	**sumresW**	**avgresW**	**avgdstW**	**lenxC**	**sumresC**	**avgresC**	**avgdstC**
	545	828371.7	1519.95	29.41	49	67613.6	1379.87	27.71
	637	998505.14	1567.51	28.09	31	43252.86	1395.25	26.89
	462	1589383.26	3440.22	39.72	25	85392.67	3415.71	38.3
	543	1382588.25	2546.2	32.35	14	61899.37	4421.38	47.97
	524	945750.61	1804.87	33.41	16	67286.27	4205.39	57.38

Table 2: The table shows derived attributes of example from figure 4. The meaning of table fields is the same as in table 1.

	apr	rt
manmade	43.1818	0.6050
natural	56.8182	0.6000

Table 3: The table shows the results of Weka's "Random Tree" algorithm. In the first column apriori probabilities of both classes are stated and in the second column the percentage of true positives as classified by "Random Tree" is shown.

REFERENCES

[1] J. Kovač, *3D Object Reconstruction from Single View*, University in Ljubljana, Faculty of computer and information science, Master thesis (2007).

[2] —. Open Source Computer Vision Library homepage. [Online]. Available: http://www.intel.com/technology/computing/opencv

[3] C. Rother, *A New Approach to Vanishing Point Detection in Architectural Environments*, In 11th British Machine Vision Conference (BMVC2000) (2000) 382–391.

[4] J. C. Russ, *The image processing handbook (2nd edition)*, IEEE Press, (1995).

[5] D. G. Stork, *Did Hans Memling Employ Optical Projections When Painting Flower Still-Life*, In Journal of the International Society for the Arts, Sciences and Technology, **2** 38 (2005) 155–160.

[6] —.Weka: A Java data mining package. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/