Vanishing Point Detection with an Intersection Point Neighborhood*

Frank Schmitt and Lutz Priese

Institute for Computational Visualistics, University of Koblenz-Landau, Germany
{fschmitt,priese}@uni-koblenz.de
http://www.uni-koblenz-landau.de/koblenz/fb4/institute/icv/agpriese

Abstract. A new technique to automatically detect the vanishing points in digital images is presented. The proposed method borrows several ideas from various papers on vanishing point detection and segmentation in sparse images and recombines them with a new intersection point neighborhood on $\mathbb{Z}^2$.

Keywords: Image analysis, measurable lines, intersection points, vanishing points.

1 Introduction

There exists a large literature on automatic detection of vanishing points in digital images by an analysis of intersection points of straight lines, using rather different techniques. There are three main problems: i) finding straight lines, ii) construction of an accumulator space of intersection points, and iii) interpretation of the values in the intersection point accumulator space.

We add new ideas to i) in 5.1, where we add a wildness map to reduce noise in the Hough transformation, to ii) in 4.1, by using a finite part of $\mathbb{Z}^2$ with a new intersection point neighborhood as a substitute for an "accumulator" of intersection points, and to iii) in 4.2, where we cluster intersection points to candidates for vanishing points with the AGS algorithm of [1]. The introduction of the intersection point neighborhood is our main contribution. This neighborhood defines a proximity that is proportional to the amount of theoretically possible intersection points of straight lines in an image. This amount depends on the way one measures straight lines. The intersection point neighborhood technique does not require calibrated cameras, does not impose restrictions on the geometry and number of vanishing points, but improves vanishing point detection.

2 Notations

We regard an *image* I as a mapping $I : Loc \to Val$ that maps coordinates $p \in Loc$ to values $I(p)$ in Val. Usually $Loc = [0, N-1] \times [0, M-1]$ in 2-dimensional images

* This work was supported by the DFG under grant PR161/12-1 and PA599/7-1.

S. Brlek, C. Reutenauer, and X. Provençal (Eds.): DGCI 2009, LNCS 5810, pp. 132–143, 2009.

and $Val = [0, 2^n - 1]$ for gray value images or $Val = [0, 2^n - 1]^3$ for color images, where $[N, M]$ denotes the interval of integers between N and M. However, we also allow $Loc \subseteq \mathbb{Z}^2$. I is a *binary* image if $|Val| = 2$. Often the values 0 and 255 are used in binary images.

A *Hough transformation* of I is a function $h_I : H \to \mathbb{N}$. H is called the *accumulator*, an element $b \in H$ a *bin*. Usually H is of some higher dimension and a bin is a formal description of an object or a set of objects in I. The value $h_I(b)$ tells how often the object described by b appears in the image I. It is often helpful to regard h_I as an image with H as its location and values from $\mathbb{N}$. The Hough transformation is frequently used to detect straight lines in a binary image I. For this, a straight line is represented in the *Hesse normal form* and the origin $(0, 0)$ of $\mathbb{R}^2$ is thought to be in the middle of Loc_I. The Hesse normal form describes a straight line l by two parameters: α, the *angle* between the normal of the line and the x-axis, and d, the *distance* between line and image origin. In this representation the accumulator H is 2-dimensional, one coordinate for α and another for d, and a bin (α, d) corresponds to the straight line $l = \{(x, y) \in \mathbb{R}^2 | x \cdot cos\,\alpha + y \cdot sin\,\alpha = d\}$.

For any image I we denote by I^e the binary image of all edges in I that one may compute by transforming I into a greyscale image and applying a Canny edge detector. I^l denotes the binary image of all straight lines in I. An edge point in I^e and a line point in I^l is set to the value 255. By I^i we denote a list of measured intersection points from pairs of straight lines in I^l. I^i is used to compute candidates for the vanishing points.

For a set $M \subseteq \mathbb{R}^n$ of n-dimensional vectors the mean μ_M and standard deviation σ_M are the vectors of the means μ_M^i and standard deviations σ_M^i of all i-th coordinates in M.

3 Previous Work on Vanishing Point Detection

We mention only a few papers from the large literature on vanishing point detection that have influenced our approach.

One may regard the list I^i of measured intersection points as an infinite image $I^i : \mathbb{Z}^2 \to \mathbb{N}$, where $I^i(p)$ tells us how many pairs of straight lines from I intersect in $p \in \mathbb{Z}^2$. Seo et al present in [2] a technique where the infinite image I^i is transformed into three finite images I_1^i, I_2^i, and I_3^i. I_1^i is I^i restricted to the locations Loc_I of I. Let Q_i be one of four infinite parts of $\mathbb{R}^2$ between two neighbored infinite diagonals through the origin (0,0) in $\mathbb{R}^2$ with Q_1 left , Q_2 right, Q_3 above, and Q_4 below (0,0). Further, set Q_i' to be the infinite part of I^i inside Q_i but outside Loc. Seo now uses a reciprocal transformation to put Q_1' and Q_2' into a single finite image I_2^i and Q_3' and Q_4' into I_3^i. However, the transformation seems to be ad hoc without a theoretical justification. Also, a cluster of intersection points in I^i near a corner of Loc may lead to three smaller clusters in I_1^i, I_2^i and I_3^i and will complicate the cluster analysis.

Almansa et al use in [3] the idea to build a histogram of I^l. The bins of the histogram are areas in $\mathbb{R}^2$. Positions inside the image location Loc are represented

by a circle c in $\mathbb{R}^2$ with center in (0,0) and diameter equaling the image diagonal. All histogram bins inside c are of the same size. Outside the circle the bins are areas of the shape of a segment of a circle from radius r_1 to r_2 around (0,0) and from angle ω_1 to ω_2. The values of $r_1, r_2, \omega_1, \omega_2$ are chosen in such a way that all bins have the same probability to contain a straight line in $\mathbb{R}^2$ that intersects c. This is a nice idea. However, the distribution of the discrete straight lines in digital images is very different from that idealistic distribution in $\mathbb{R}^2$ and depends on the way one measures discrete straight lines. Also, a histogram of the intersection points in I^i seems to be more adequate than of straight lines in I^l.

In architectural environments with parallel buildings on a plane there are normally two or three vanishing points. Rother [4] tries to use triples of three intersection points in the list I^i as candidates for vanishing points and searches triples fulfilling geometric constraints. However, if there are n straight lines in I^l one gets up to $O(n^2)$ intersection points and thus $O(n^5)$ possible triples of intersection points, leading to an algorithm with a running time that is $O(n^5)$, which is not acceptable in practice.

Seo [2] constructs a histogram of the angles of all found straight lines. The histogram is separated into three intervals. It is argued that intersection points of two straight lines from different intervals cannot become a vanishing point. Thus, only the intersections of straight lines of the same interval are regarded in I^i. This should simplify I^i substantially.

4 Some Theoretical Considerations

4.1 Intersection Point Neighborhood

Measurable Straight Lines. We follow the ideas in [2], [3] and [4]. A first consideration is that the size of the bins for some accumulator for straight lines or intersection points should depend on the number of *measurable* straight lines that intersect an image I. Suppose, one uses an algorithm where the straight lines are detected with a Hough transformation using the Hesse normal form. The accumulator H is usually set to be $H = [0, A-1] \times [0, D-1]$ of size $A \cdot D$ where $[0, A-1]$ represents the discrete angles and $[0, D-1]$ represents the discrete distances of the Hesse normal form. Usually, not all straight lines (α, d) represented in H will intersect with the coordinates $Loc(I)$ of the image. Remove those "false" bins of straight lines not intersecting with the image from H and call the resulting set H_l the *admissible accumulator of size* (A, D). A bin in H_l is called a *measurable line* (for I). If one prefers a technique that does not use a Hough transformation the definition of a measurable straight line must be adopted. In any case, a measurable line must be

- straight,
- intersecting with the image,
- detectable.

For simplicity, we continue our argumentation with a Hough transformation and an admissible accumulator H_l.

(a) Clipping of size $4000x3000$ **(b)** Clipping of size $1600x1200$

Fig. 1. Φ, measurable lines per pixel

Frequency Functions. Let $\Phi(p)$ be the number of measurable lines that run through $p \in \mathbb{Z}^2$. We call $\Phi : \mathbb{Z}^2 \to \mathbb{N}$ the *line frequency* function. As $\Phi(p)$ tells how many measurable lines run through p exactly $\Psi(p) := (\Phi^2(p) - \Phi(p))/2$ many pairs of those lines may intersect in p. Ψ is called the *intersection frequency* function.

Figure 1 presents a part of Φ for an admissible accumulator H_l of size $(500, 500)$ for an image of location size 800×600. Φ is restricted to an area in $\mathbb{N}^2$ of size 4000×3000 in (a) respectively of size 1600×1200 in (b), where in both cases Loc_I is in the upper left corner. The values of Φ are transformed into the interval [0,255] to present Φ as a greyscale image. We have calculated the mapping $\Phi : \mathbb{Z}^2 \to \mathbb{N}$ iteratively. Initially $\Phi(p)$ is set to 0 for all $p \in \mathbb{Z}^2$. For each measurable line $l \in H_l$ we virtually draw l into the image Φ by a standard technique for drawing lines in discrete images. Whenever our virtual drawing touches a coordinate $p \in \mathbb{Z}^2$ Φ is accumulated by 1 in p.

It turns out that the frequency function is not constant inside Loc_I, however the line frequencies inside Loc_I are very similar. Outside Loc_I the line frequencies differ heavily. Let $\hat{\Phi}$ denote the mean value of $\Phi(p)$ inside Loc_I. We approximate with $\hat{\Psi} := (\hat{\Phi}^2 - \hat{\Phi})/2$ the mean intersection frequency inside Loc_I.

Intersection Point Neighborhoods. We want to define for any position $p \in \mathbb{Z}^2$ a neighborhood $N(p) \subseteq \mathbb{Z}^2$ in such a way that the aggregated intersection frequencies $\sum_{p' \in N(p)} \Psi(p')$ are independent of p. For the sake of simplicity, a *neighborhood* $N(p)$ *of* p shall become a circle

$$c_r(p) := \{p' \in \mathbb{Z}^2 | d_E(p, p') < r\}$$

of radius r around p. d_E is the Euclidean distance in $\mathbb{R}^2$. For $p \in Loc_I$ we choose some fixed radius $\hat{r}$. The question is how to choose a radius r_p for a point p outside Loc_I. We assume for a moment that the intersection frequency function is constant inside the circle $c_{r_p}(p)$. As the area of a circle of radius r is proportional to r^2 one should choose r_p in such a way that $r_p^2 \cdot \Psi(p) = \hat{r}^2 \cdot \hat{\Psi}$.

Thus,

$$\frac{r_p^2}{\hat{r}^2} = \frac{\hat{\Psi}}{\Psi(p)} = \frac{\hat{\Phi}^2 - \hat{\Phi}}{\Phi^2(p) - \Phi(p)} \approx \frac{\hat{\Phi}^2}{\Phi^2(p)} \text{ and } r_p \approx \hat{r} \cdot \frac{\hat{\Phi}}{\Phi(p)}.$$

We therefore choose the neighborhood of p outside Loc_I as

$$N(p) := c_{r_p}(p) \text{ with } r_p := \hat{r} \cdot \frac{\hat{\Phi}}{\Phi(p)}.$$

Those neighborhoods replace the reciprocal transformation in [2] and the bins in [3]. Although they form no topological space they define a concept of proximity and connectedness: two positions $p, p' \in \mathbb{Z}^2$ are *close* if $p \in N(p')$ and $p' \in N(p)$ and a set $S \subseteq \mathbb{Z}^2$ is *connected* if for any two points $p, p' \in S$ there exist points $p_1, ..., p_n \in S$ s.t. $p \in N(p_1)$, $p' \in N(p_n)$ and $p_{i+1} \in N(p_i)$ for $1 \leq i < n$.

4.2 Cluster Analysis with the Intersection Point Neighborhood

Suppose we have computed the image I^l of all straight lines in I. Then $I^i : \mathbb{Z}^2 \to \mathbb{N}$ is the image of all intersection points, where $I^i(p) = n$ means that n pairs of lines in I^l are intersecting in $p \in Z^2$. We also call n the *multiplicity* of intersection points at p. One may regard I^i as a list or as a very sparse image with the value 0 almost everywhere. A next step is to find clusters in I^i and to take the centers of gravity of those clusters as candidates for vanishing points. Of course, clusters must not be build according to the Euclidean distance in $\mathbb{Z}^2$ but according to the proximity given by the intersection point neighborhood.

As our neighborhood does not define a metric space we must carefully choose a cluster anaysis technique. Therefore we apply the AGS (**A**utomatic **G**rouping of **S**emantics) algorithm that was introduced by Priese, Schmitt, Hering in [1] for an automatic grouping of locations of a similar semantics. The advantage of the AGS is that it can find groupings in spaces with a concept of a neighborhood but without a topology. The AGS follows ideas of the CSC (Color Structure Code) segmentation technique of Priese and Rehrmann [5] and works as follows:

One starts with $\mathcal{N} := [N(p)|p \in I^i]$, a list of the neighborhoods $N(p)$ of all intersection points p in I^i, as an initial grouping. AGS will merge overlapping groups G, G' in $\mathcal{N}$ if they are similar enough as shown in the following pseudo code. Similarity is measured by the overlap rate $O(G, G') := |G \cap G'|/min(|G|, |G'|)$ and some threshold t_{ov}.

```
  H := N;
* G:= empty list;
  for 0 ≤ i < |H| do G := H[i];
      for 0 ≤ j < |H|, i ≠ j do
          if G = H[j]
              then remove H[j] from H
              else if O(G, H[j]) > t_ov then G := G ∪ H[j]
      end for;
      insert G into G
```

end for;
if $\mathcal{H} \neq \mathcal{G}$
then $\mathcal{H} := \mathcal{G}$; goto line *
else end.

Each computed group G in $\mathcal{G}$ consists of connected intersection points with a multiplicity. The *center* c_G of such a group is the weighted mean μ_G of all intersection points in G according to their multiplicity. The center represents the coordinate in $\mathbb{R}^2$ of a possible vanishing point. The *size* $|G|$ of G is the sum of the multiplicities of all points in G. It tells how many intersection points have contributed to c_G. The *weight* w_G of G is the number of measured lines that have produced the intersection points in G. There are two extreme cases: G may result from a single line that intersects with n more or less parallel lines, leading to $w_G = n + 1$ and $|G| = n$, or from n' lines all intersecting with each other within G, leading to $w_G = n'$ and $|G| = (n'^2 - n')/2$. Thus, in any case we have

$$w_G - 1 \leq |G| \leq \frac{w_G^2 - w_G}{2}.$$

5 An Application

We present some details how we detect vanishing points with the intersection point neighborhood. In a current project we have to find the position and direction of a camera from the camera image and a 3d model of the environment. For this, we must detect the facades of buildings. If there is no perspective projection distortion, as in Figure 2a, we can do so without a knowledge of vanishing points. However, in images as in Figure 2b the vanishing points become important. Thus, we apply some simple geometrical considerations inspired by [4].

Basically, we expect to see two to three vanishing points, namely an upper, a left and a right vanishing point where either the left or right one might be missing (see, e.g., figure 2c). Thus, we need two or three groups of measured lines, where the lines of one group lead to one vanishing point.

We transform an image I into a greyscale image and apply a Canny edge detector to get the edge image I^e. Now, a Hough transformation is applied to I^e to get the line image I^l. However, even in such rather simple architectural environments standard implementations of a Hough transformation do not lead to satisfactory line images I^l. We thus have been forced to turn our attention to improve the Hough transformation, a step that is more or less independent from our application scenario.

5.1 Noise Reduction in the Hough Transformation

We apply two filters on the Hough accumulator to improve I^l. The first one from [6] removes accumulator maxima resulting from edge points which are not part of longer lines in the image (in outdoor images, such edge points typically occur, e.g., in trees) by increasing accumulator values in the middle of

(a) No perspective distortion, three vanishing points: top, center, infinity

(b) Perspective distortion, three vanishing points: top, left, right

(c) Perspective distortion, two vanishing points: top, right

Fig. 2. Vanishing points in facades

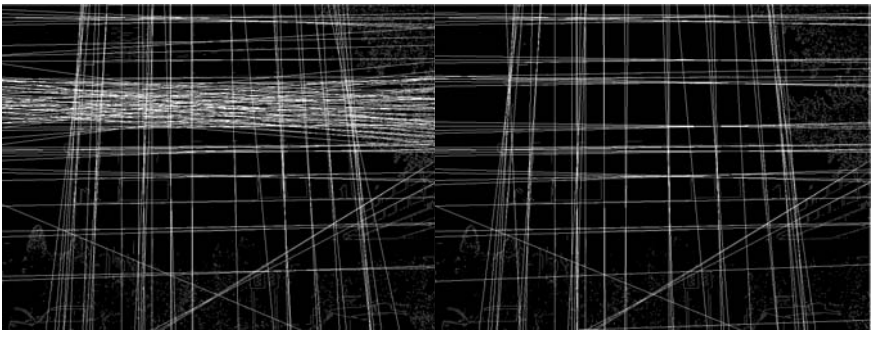
the typical butterfly formed structures resulting from true lines while lessening other accumulator values. When no such filter is applied, the numerous edges in strongly textured structures often yield artificial straight lines in the Hough transformation. The second filter, developed by us, thins out straight lines closely neighbored in the accumulator.

Unfortunately, even this is insufficient as often Moir lines in window shutters or straight structures in roofs introduce many "false" lines. We therefore have developed a "wildness map" which characterizes "wild" parts of an image and masks them in I^e. The remaining straight lines are the *measured lines*.

Wildness at an image position $p \in Loc_I$ is measured by calculating two values in a window w_p of size 11×11 around p:

- the standard deviation σ of all values in w_p, and
- the sum of the differences between a pixel $p' \in w_p$ and its four direct neighbor pixels is averaged across all $p' \in w_p$ and gives $\mu_D(w_p)$.

The value of the wildness map at p is calculated as $c_1 \cdot \mu_D(w_p) - c_2 \cdot \sigma$ with constants $c_1, c_2 \in \mathbb{R}$ assuring that both values are in the same number range. Figure 3 shows the effect of removing wild structures in the image (a) of figure 2. Lines found by the Hough transform are shown in white, edge points in I^l in grey.



(a) Hough of building (a) **(b)** Hough after removing wild structures

Fig. 3. Removing wild textures for better line extraction

5.2 Construction of I^i

The previous step is rather independent from our application but the following steps follow our application scenario. We expect to find two or three vanishing points with certain geometrical restrictions. Thus, we first follow the idea of Seo [2] and group all measured lines according to their angle α. For this we use two independent k-harmonic-means clustering [7] runs, one time with two clusters and the other time with three clusters. Only when the intra-cluster variance drops substantially the result from the clustering in three groups is used. Otherwise we use only two groups and the group of vertical lines must have a width less than 30°.

For each pair of measured lines in one group we compute their intersection point and store it in a list I^i. It remains to handle vanishing points in infinity that may result from parallel lines. We expect two vanishing points on the horizon line and a third orthogonal to and above that line. The maximal distance Δ_{max} of a measurable intersection point from Loc_I depends on the size of Loc_I and the admissible accumulator H_l and is easily computed. Two measured lines $l_1, l_2 \in I^l$ with parameters α_i, d_i are parallel if $\alpha_1 = \alpha_2$ holds. In this case we regard the middle line $l_{1,2}$ between l_1 and l_2 with the parameters α_1 and $(d_1 + d_2)/2$. If $l_{1,2}$ is roughly a vertical line we chose as an point the point on $l_{1,2}$ in distance Δ_{max} above $loc(I)$. In case of a horizontal line we choose two intersection points in distance Δ_{max} on $l_{1,2}$ right and left from $loc(I)$. Those intersection points are the substitute of the intersection of the two parallel lines in infinity and they are added to I^i. If one just changes the angle of one of the two parallel lines by

the smallest angle possible in H_l the resulting intersection point would be in a distance less than Δ_{max}.

An intersection point with a multilicity n will thus occur exactly n times in our list I^i.

5.3 Grouping in I^i

We now cluster the intersection points in I^i according to their neighborhoods with the AGS algorithm. The threshold t_{ov} is set to 0.6. The output of this step is a list $\mathcal{C} = [(c_G, |G|, w_G)|G \in \mathcal{G}]$ of all centers c_G of groups G in $\mathcal{G}$ together with the size and weight of G.

5.4 Construction of Vanishing Points

The list $\mathcal{C}$ contains weighted candidates for vanishing points. Each of the vanishing point candidates in $\mathcal{C}$ results from intersections of lines in one group of lines with similar directions. If three line direction groups have been found we assume that each of them represents lines to one of the three possible vanishing points. If only two line direction groups have been found the vanishing point candidates from one group are assigned to upper and the candidates from the other direction group are split into left and right. In our application a bird's eye view and an against the horizon line rotated camera are quite unlikely. This helps to use some geometrical restrictions. We firstly remove all vanishing point candidates c_G in $\mathcal{C}$ which are located inside the upper third of Loc_I or with $|G| < 3$, i.e., that result from less than three intersection points.

Often, inside Loc_I many intersection points are computed between lines pointing in very different directions. This intersections shall not contribute to a vanishing point. The introduction of two or three groups for different line angles helps to avoid this problem but will not always solve it. However, outside Loc_I this problem almost never arises. We therefore try to solve this problem by two actions:

Firstly, we use a different reference radius $\hat{r}$ inside and outside Loc_I where the value outside is multiplied by 1.5. In the following evaluation $\hat{r} = 6$ is used outside and $\hat{r} = 4$ inside Loc_I.

Secondly, we regard the situation that a single measured line crosses a bundle of almost parallel measured lines, resulting in many neighbored intersection points. To prevent that they form a vanishing point we compare the weight w_G with the size $|G|$ for any group G with its center c_G inside Loc_I. $|G|$ becomes close to $max_G := (w_G^2 - w_G)/2$, the maximal possible number of intersection points in G, if most lines of G accumulate into a neighbored set of intersection points. The size $|G|$ becomes rather small if one or two lines intersect a bundle of almost parallel lines. Thus, we also drop c_G if $|G| < 0.5 \cdot max_G$ holds.

Now, we calculate for each triple $(c_{G^l}, c_{G^r}, c_{G^u})$ of left, right and upper vanishing point candidates the size $S := \sqrt{2 \cdot |G^l|} \cdot \sqrt{2 \cdot |G^r|} \cdot \sqrt{2 \cdot |G^u|}$ and search for the triple with the biggest size fulfilling the following conditions:

- the angle between x-axis and the horizon line l_h between left and right vanishing point must be below 5°,

- l_h must be below the image center,
- the difference between the mean angle of lines to the upper vanishing point and the angle of the normal to l_h must be below 7.5°,
- the left vanishing point must be left of the image center and the right must be right of the image center.
- let d_l be the distance from image center to left vanishing point, d_r the distance from image center to right vanishing point and $diag$ the length of the image diagonal.
  - At least one of d_l and d_r must be bigger than $\frac{diag}{2}$
  - If $d_l > 2 \cdot diag$ than d_r must be smaller than $diag$ and vice versa

6 Evaluation

We tested our algorithm on 70 images taken at the campus of our university for which the "true" vanishing points have been annotated manually. This has been done by another group, not by the authors, by manually annotating several straight lines in an image that run to the same vanishing point. However, one has to note that in many images the location of those "true" vanishing points can only be estimated vaguely. Due to imperfect parallelism in 3d, lens distortions in the camera, and limited angular accurateness, straight lines that should intersect in one vanishing point in fact show intersections scattering in a rather large area.

Nevertheless, we attempt a quantitative evaluation where we compare each calculated vanishing point $c = (c_x, c_y)$ against a true vanishing point $t = (t_x, t_y)$. Such a measure of similarity can not be the Euclidean distance. Suppose we have an imgae $I : [0, 799] \times [0, 599] \rightarrow [0, 255]^3$ with a horizon line on $y = 400$. A computed vanishing point $(-20000, 390)$ far outside the image location is obviously similar to a true vanishing point $(-18000, 400)$ in spite of their rather large Euclidean distance. If the true vanishing point is $t = (350, 400)$ a computed vanishing point $c_1 = (380, 400)$ is better than $c_2 = (350, 420)$, although $d_E(t, c_2) < d_E(t, c_1)$. It turns out that the 3d angle between t and c measured from some height on a tower on the origin reflects very well the quality of the computed vanishing points: the lower this angle the better. In our images with a resolution of 1024×768 pixels the height of 40 to 90 pixels of this tower gives the best angles for an evaluation. As a suitable similarity measure for c and t we therefore calculate the angle between $\tilde{c} = (c_x, c_y, 40)$ and $\tilde{t} = (t_x, t_y, 40)$. This is $\alpha_{\tilde{c},\tilde{t}} = acos\left(\frac{\tilde{c} \cdot \tilde{t}}{|\tilde{c}| \cdot |\tilde{t}|}\right)$.

We say t is *successfully detected* if a vanishing point c is computed with $\alpha_{\tilde{c},\tilde{t}} \leq 3°$, *unsuccessfully detected* if $\alpha_{\tilde{c},\tilde{t}} > 3°$, and *undetected* if no vanishing point for t is computed at all. Figure 4 shows the distribution of $\alpha_{\tilde{c},\tilde{t}}$ across all detected vanishing points.

Table 1 gives the number of successfully detected, unsuccessfully detected, and undetected annotated vanishing points as well as the mean and standard deviation σ of $\alpha_{c,t}$ in degrees for all successfully detected vanishing points. Our method has detected 154 of 189 annotated vanishing points, resulting in a success rate of

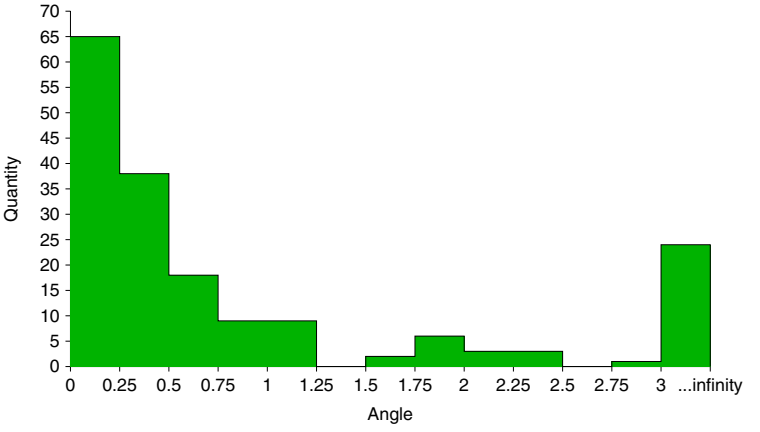


Fig. 4. Histogram of angles between vector to true and calculated vanishing point

Table 1. Evaluation of vanishing point detection

	total number	successfully detected	unsuccessfully detected	undetected	mean of $\alpha_{c,t}$ for successful t	σ of $\alpha_{c,t}$ for successful t
left	64	49	9	6	0.62	0.66
right	55	37	13	5	0.57	0.59
upper	70	68	2	0	0.45	0.51
total	189	154	24	11	0.54	0.58

81.48%. In most cases we find very small angles but a few rather large ones increase the mean. This is why σ becomes 0.58 with a smaller mean of just 0.54.

Only 7 out of the 189 calculated vanishing points do not correspond to a true vanishing point at all. This gives an error rate of 3.79%.

A visual inspection of the calculated vanishing points shows even less errors as the annotated "true" vanishing points are sometimes questionable. Further, in most of the images where vanishing points are undetected or unsuccessfully detected the Hough transform has not produced reasonable straight lines to operate with.

7 Conclusion

We have presented a new neighborhood on line intersection points and its application in detection of vanishing points in architectural environments. Our algorithm combines ideas from several algorithms in literature in order to achieve reliable results in reasonable time. In the future, we want to apply our new algorithm for facade detection and matching between a 3d model and camera

images. However, we believe that the intersection point neighborhood can improve vanishing point detection in very different application scenarios.

We would like to encourage the community to use our improved algorithms for Hough and vanishing points detection in further research. The algorithms are freely available under `http://www.uni-koblenz-landau.de/koblenz/fb4/institute/icv/agpriese/downloads/vpoints`

Acknowledgment

We would like to thank the unknown reviewers for their valuable comments.

References

1. Priese, L., Schmitt, F., Hering, N.: Grouping of semantically similar image positions. In: Salberg, A.-B., Hardeberg, J.Y., Jenssen, R. (eds.) SCIA 2009. LNCS, vol. 5575, pp. 726–734. Springer, Heidelberg (2009)
2. Seo, K.S., Lee, J.H., Choi, H.M.: An efficient detection of vanishing points using inverted coordinates image space. Pattern Recogn. Lett. 27(2), 102–108 (2006)
3. Almansa, A., Desolneux, A., Vamech, S.: Vanishing point detection without any a priori information. IEEE Trans. Pattern Anal. Mach. Intell. 25(4), 502–507 (2003)
4. Rother, C.: A new approach for vanishing point detection in architectural environments. In: Proc. 11th British Machine Vision Conference, pp. 382–391 (2000)
5. Rehrmann, V., Priese, L.: Fast and robust segmentation of natural color scenes. In: Chin, R., Pong, T.-C. (eds.) ACCV 1998. LNCS, vol. 1351, pp. 598–606. Springer, Heidelberg (1997)
6. Leavers, V.F., Boyce, J.F.: The radon transform and its application to shape parametrization in machine vision. Image Vision Comput. 5(2), 161–166 (1987)
7. Zhang, B., Hsu, M., Dayal, U.: K-harmonic means - a spatial clustering algorithm with boosting. In: Roddick, J., Hornsby, K.S. (eds.) TSDM 2000. LNCS (LNAI), vol. 2007, pp. 31–45. Springer, Heidelberg (2001)