EUROGRAPHICS Workshop on Graphics and Cultural Heritage (2022)
R. Pintus and F. Ponchio (Editors)

Semi-Automatic Perspective Lines from Paintings

Yoann Coudert–Osmont[1] and Elmar Eisemann[2] and Ricardo Marroquim[2]

[1]Université de Lorraine, CNRS, Inria, LORIA
[2]Delft University of Technology



Figure 1: *Extracted vanishing point and horizon using the proposed method. An Amsterdam Canal House Garden, Cornelis Troost. Courtesy: Rijksmuseum*

Abstract

Perspective cues play an important role in painting analysis as it may unveil important characteristics about the painter's techniques and creation process. Nevertheless, extracting perspective lines and their corresponding vanishing points is usually a laborious manual task. Moreover, small variations in the lines may lead to large variations in the vanishing points. In this work, we propose a semi-automatic method to extract perspective lines from paintings in order to mitigate the human variability factor and reduce the workload.

CCS Concepts

• ***Computing methodologies*** → ***Image processing;*** • ***Applied computing*** → ***Fine arts;***

© 2022 The Author(s)
Eurographics Proceedings © 2022 The Eurographics Association.

1. Introduction

Since the 15 century, many artists systematically use the mathematically-based rules of perspective to increase realism and bring more dramatic perspective effects to their paintings. For precise perspective constructions, some artists placed pins on the canvas to mark vanishing points. Chalked strings would then be attached to the pin to mark perspective lines on the canvas during the sketching phase.

The pin mark is still visible in some paintings through x-ray images or, occasionally, even with the naked eye. This is the case for some paintings by Johannes Vermeer and other Dutch masters, but the pin technique can be traced back to Italian artists from the previous century [Heu00].

The analysis of the perspective in paintings can give insights into the techniques and the creation process of artists [Sto04, Sto06]. In addition, it can reveal how precise they were in this task, regardless of the author's actual intention of creating perspectively-accurate paintings. Vanishing points can have artistic intent and were sometimes carefully placed to direct the attention of the viewer to specific points. This also holds for some lines used in art, which do not relate to perspective alone but arise from combining different scene elements. In this paper, we focus on perspective but the methodology can be applied in this broader context as well.

Extracting perspective lines and their corresponding vanishing points has mainly been done manually, either by tracing with a pen on a paper replica or using some off-the-shelf image editing software. It is not only a laborious manual job but, more critically, the extracted lines and points may suffer from imprecision due to human interaction and bias. Even the same person might obtain different results when performing the process several times.

The main issue lies in the fact that small deviations when annotating points and lines may lead to large deviations on the resulting vanishing points. Differently from photographs, paintings usually do not have perfect perspective and irregularities in the brushstrokes due to artistic styles are common. Consequently, there is room left for human interpretation of the exact placement of lines. In addition, after extracting a few lines, a human operator might be biased to drawing other lines that agree with the previously extracted ones.

In this paper, we aim at tackling these issues by proposing a semi-automatic method for extracting perspective lines, vanishing points and the vanishing line or horizon line from paintings. Our contribution is twofold. First, we describe a robust method to extract lines from paintings. Second, we couple it to a user-guided process to extract perspective lines. Therefore, we include functionalities to allow users to add/remove lines and select regions of interest. We have extensively tested our method with many paintings that contain different characteristics.

2. Perspective rules

The main perspective rule is that parallel lines in the 3D world when projected under perspective transformation all cross at the same 2D point. This is called the vanishing point of the set of lines. Under one point perspective there is a single vanishing point. For two point perspective there are two vanishing points that define the vanishing line. Even though for one point perspective there is no vanishing line, a horizon can be defined as the horizontal line that passes through the vanishing point. Figure 2 illustrates these concepts.

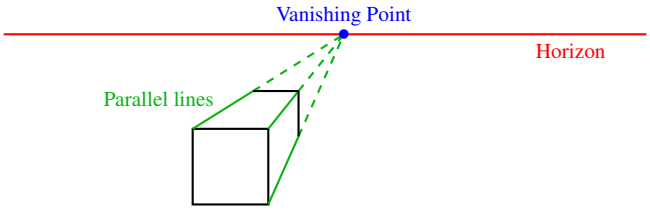


Figure 2: *Illustration of the perspective rules for one point perspective.*

3. Related Work

In the field of computer vision, many works rely on extracting lines to infer spatial information from an image. Apart from detecting vanishing points, these lines are also useful to complete other metrology tasks from images [HZ04]. Methods to reconstruct 3D models from single images have been particularly popular in architecture where many assumptions about the shapes can be reasonably made [vdH98, Rot02, LHK09, VDS14].

Hoiem et al. [HEH05] developed a method to automatically extract 3D structure from a single photo and allow camera movement in a pop-up fashion. They rely heavily on classifying regions and extracting lines between them to cut out objects. In a somewhat opposite direction, Carroll et al. [CAA10] developed a user-based annotation approach to allow for vanishing point modification of photographs.

Clark and Mirmehdi [CM03] recovered vanishing points to rectify images of text. The vanishing point is recovered by assuming properties of margin structure, from which lines can be extracted. Similarly, Baumann et al. [BBS12] extract lines from images of historical documents to rectify the pages.

Vanishing point can typically be accurately detected from photos [KJG09, Tar09, NS11, CDI14, LGvGRM14, CT18, LWP*22]. Simon et al. [SFB18] have also recently proposed a method that takes the opposite route, starting from the horizon towards the vanishing points.

Even though computer-aided painting analysis has become a valuable tool for art and historian scholars [Sto09], there has been limited work on (semi)-automatic retrieval of perspective lines and vanishing points from paintings. A few works describe the extraction of perspective cues from paintings, but were used in specific studies and no systematic method was presented [CKZ02, GS10, Sim21]. Some more recent methods do generalize, but rely on manual tracing of shapes and lines, which again introduce the human variability in the process [CS12, FGV*14].

4. Method

Briefly, our method first performs pre-processing to compute clear gradients. Then, lines are extracted and, via an interface, the user

© 2022 The Author(s)
Eurographics Proceedings © 2022 The Eurographics Association.

can remove/add lines to the visible collection. Finally, the interface allows for grouping lines to extract vanishing points and the vanishing line, or the horizon in the one point perspective case.

4.1. Pre-processing

In a first step, the image is converted from RGB space to CIELab space [CF97]. The main reason is that the Euclidean distance in this space corresponds more closely to the perceptual difference given a human observer. Extracted edges are hereby closer to what a human would identify as edges.

Following, a smoothing operator is employed to remove noise. Instead of the more traditional Gaussian blur, a bilateral filter [TM98] is applied to better preserve edges. We use a spatial standard deviation $\sigma_{space} = 1.5 \cdot 10^{-3} D$ and a color standard deviation $\sigma_{color} = 10$. Here, D is the image diagonal length. Figure 3 depicts the difference. Finally, a Scharr filter [Sch00], a variant of the Sobel filter, is applied to extract the edges of the image.



Figure 3: *From left to right: detail from original image (Figure 9); Gaussian filtering; Bilateral filtering*

At this point, two user thresholds are introduced that can be manipulated to determine which edges are preserved. The first is used to determine the minimum gradient intensity and the second to determine the minimum size of connected components. Figure 4 illustrates the effect of these parameters.

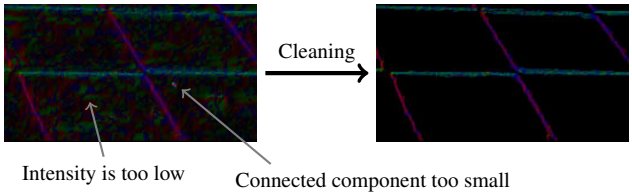


Figure 4: *Detail of cleaning the gradients from Figure 9*

Notwithstanding, the gradient orientation can still be very noisy along edges. Since these orientations guide the subsequent line retrieval step, it is important to smooth them beforehand.

We first compute a weighted average of the gradient angle Θ in a neighborhood of size 3×3 around a pixel (x, y). The weight depends not only on the pixel distance following a Gaussian kernel - for our experiments we used a Gaussian kernel with standard deviation $\sigma = 2$ - but also on the amplitude of the gradient (G) and its direction (Θ). Algorithm 1 details the weight computation process. Note that the factor 2 in the cosine and sine expressions comes from the fact that we consider angles modulo π instead of 2π, since an angle θ and $\theta + \pi$ refer to the same line. The number of smoothing steps depends on the image's resolution and is proportional to the square root of the number of pixels. Figure 5 shows the result of smoothing the gradients using the proposed method.

Algorithm 1 Gradient Smoothing

$$M \leftarrow \begin{bmatrix} 0.0925 & 0.12 & 0.0925 \\ 0.12 & 0.15 & 0.12 \\ 0.0925 & 0.12 & 0.0925 \end{bmatrix} \quad \triangleright\ 3{\times}3 \text{ Gaussian kernel } (\sigma = 2)$$

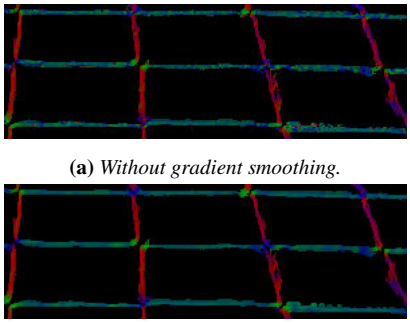
function SMOOTHGRAD(G, Θ)

$a \leftarrow M * (G \cdot \cos(2 \cdot \Theta))$ ▷ $*$: convolution product

$b \leftarrow M * (G \cdot \sin(2 \cdot \Theta))$ ▷ $\cdot$: pointwise product

$\Theta \leftarrow \arg(a + ib) \,/\, 2$

end function



(a) *Without gradient smoothing.*

(b) *With gradient smoothing.*

Figure 5: *Detail of applying gradient smoothing from Figure 9.*

4.2. Line extraction

From the pre-processed image with smoothed gradients we apply a modified version of the Hough Transform to extract the final set of lines. As in the original Hough Transform algorithm, instead of parameterizing a line as:

$$y = ax + b \tag{1}$$

we still parameterized it as:

$$\rho = x\cos(\theta) + y\sin(\theta), \tag{2}$$

where ρ is the shortest distance from the origin to the line and θ is the line direction. We use the domain $(\theta, \rho) \in [-\pi/2; \pi] \times [0; D]$ where D is the diagonal length. This domain covers precisely the entire rectangle of the image, *i.e.* $[0; W) \times [0; H)$ where W and H are the width and the height of the image.

The main modification from the original algorithm is that, instead of using a binary input, we use a weighted scheme. The weighted voting function takes into account the gradient intensity and the difference between the gradient direction and the line, as defined below:

$$f(G, \Theta, \theta, x, y) = \left(1 - |\sin(\theta - \Theta[x][y])|^{\frac{2}{3}}\right) \cdot \left(1 + \frac{3}{2}\frac{G[x][y]}{\max(G)}\right), \tag{3}$$

where G and Θ are, as indicated earlier, the gradient intensity and direction, (x, y) is the pixel position and θ is the direction of the line. This function was empirically developed through many tests. The idea behind is that the more θ and $\Theta[x][y]$ differ, the lower the weight should be and the higher the intensity of the gradient the higher the weight should be.

The method builds a new image H^*, of size $R \times T$ where R and T

© 2022 The Author(s)
Eurographics Proceedings © 2022 The Eurographics Association.

are fixed using the dimensions of the original image. Then for each pixel of position (x,y) of the gradient with a non-zero intensity, and for each $t \in [0;T)$ we compute the distance from the origin ρ, of the line of direction θ_t passing through pixel (x,y). Here $\theta_t = 3\pi/2 \cdot t/T - \pi/2$ is the angle associated to the ordinate t in the image H^* such that $t=0$ corresponds to an angle $-\pi/2$ and $t=T$ to an angle π. We then let $r = \lfloor \rho \cdot R/D \rfloor$ and add the positive value defined by Equation (3) to the pixel of coordinates (r,t) in the image H^*. The method is further detailed in Algorithm 2. In practice we choose $R = D/2$ and $T = 720$.

Algorithm 2 Modified Hough Transform

```
function HOUGH(G, Θ)
    H* ← Null image of size R × T
    for (x,y) ∈ [0,W) × [0;H) with G[x][y] > 0 do
        for t ∈ [0;T) do
            θ ← 3π/2 · t/T − π/2
            ρ ← x.cos(θ) + y.sin(θ)
            if ρ ⩾ 0 then
                r ← ⌊ρ · R/D⌋
                H*[r][t] ← H*[r][t] + f(G, Θ, θ, x, y)
            end if
        end for
    end for
    return H*
end function
```

As additional improvements, we also avoid voting for all possible θ values since we are only interested in values near $\Theta[x][y]$. Finally, once a pixel (r,t) in the Hough transform is identified as a peak, that is, an actual line in the image, we readjust the value of the corresponding line (ρ,θ) by using a small neighborhood $\mathcal{N}$ of size 3×3 around (r,t). We first compute the minimum of H^* in the neighbourhood and denote it by $h_\mathcal{N}$. Then, ρ and θ are slightly readjusted. Hereby, the solution will converge to a higher precision than using the original Hough transform grid. The refinement step is as follows:

$$\rho = \left(r + \frac{\sum_{(r',t')\in\mathcal{N}} (r'-r)\cdot(H^*[r'][t'] - h_\mathcal{N})}{\sum_{(r',t')\in\mathcal{N}} (H^*[r'][t'] - h_\mathcal{N})} + \frac{1}{2} \right) \cdot D/R \quad (4)$$

$$\theta = \left(t + \frac{\sum_{(r',t')\in\mathcal{N}} (t'-t)\cdot(H^*[r'][t'] - h_\mathcal{N})}{\sum_{(r',t')\in\mathcal{N}} (H^*[r'][t'] - h_\mathcal{N})} \right) \cdot \frac{3\pi}{2T} - \frac{\pi}{2} \quad (5)$$

4.3. Vanishing points and lines

The vanishing point (x,y) of a group of lines is computed by minimizing the sum of the squared distance of the vanishing point to each line of the group. Representing the i-th line of the group by its Hough transform (θ_i,ρ_i), the vanishing point is obtained via the following optimization:

$$(x,y) = \arg\min_{x,y} \sum_i (\cos(\theta_i)x + \sin(\theta_i)y - \rho_i)^2 \quad (6)$$

Thus, a vanishing point can be computed by inverting a 2×2 matrix:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \left(\sum_i u_i u_i^T \right)^{-1} \left(\sum_i \rho_i u_i \right), \quad u_i = \begin{pmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{pmatrix} \quad (7)$$

Finally, the vanishing line is computed by defining one of its points as the barycenter of the vanishing points and its direction as to the first principal component direction of the PCA [F.R01] of all vanishing points. In the case of an accurate one point perspective, a set of lines will be close to parallel, hence the vanishing point will lie at infinity. We detect this case using a small threshold and place a horizontal line passing through the vanishing point to represent the horizon. Note that in many cases the paintings are not accurate enough and a second vanishing point might be found even under one point perspective.

4.4. User interaction

Given that paintings do not have exact edges, relying on a fully automatic algorithm without priors is not possible. We introduce a few user interactions to guide the process.

Apart from the aforementioned edge extraction thresholds, it is possible to pre-select a region of interest using a lasso tool to define where the gradients are computed, as depicted in Figure 6. After extracting the gradients, it is possible to further edit the region of interest by using a brush tool to remove undesirable gradients, as depicted in Figure 7.

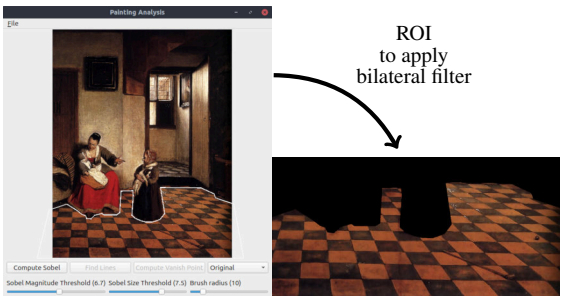


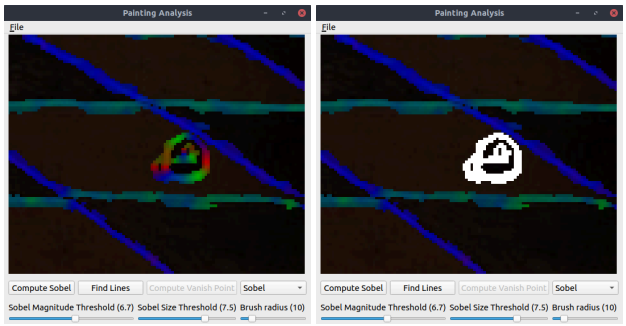
Figure 6: *The lasso tool being used to select a region of interest that contains important perspective cues.*

The benefit of selecting regions is twofold. First, it lowers the computational time and, second, it avoids detecting undesired lines from regions that have no relevant perspective cues.

The lines with highest weights are displayed on top of the image and the user has the possibility to remove some of them. Moreover, by clicking on a pixel all the corresponding lines that traverse this pixel are shown, allowing one or more lines to be included in the selection.

The lines can then be grouped to find corresponding vanishing points, as described in Section 4.3. Once two different vanishing points are identified the algorithm automatically computes the associated vanishing line or the horizon if the second vanishing point cannot be found.

© 2022 The Author(s)
Eurographics Proceedings © 2022 The Eurographics Association.



(a) *Gradients after smoothing.* **(b)** *Selected region to be removed.*

Figure 7: *Example of applying a brush to remove undesired gradient regions from Figure 9.*

5. Results

As a first example, we show the result of applying the method to a photograph which has precise edges. From Figure 8 it is possible to see that the method accurately retrieves the vanishing point and an almost-vertical vanishing line.

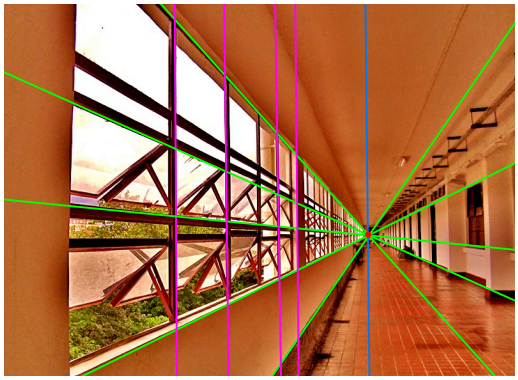


Figure 8: *The method applied to a photo. Photo source: Wikipedia*

We further show results for a series of paintings. All images were taken from the Rijksmuseum and the National Gallery collections that provide, respectively, high and low resolution scans. Only the Last Supper images (Figures 22 and 23), as well as one painting by Pieter de Hooch (Figure 9) were taken from other sources, namely Wikipedia and Wikimedia Common.

For most examples there was need for user interaction since the initial selection contained undesired lines and other lines needed to be added by clicking on points along edges and choosing appropriate perspective lines. For most cases, the entire process including the processing of gradients, line extraction and all user interaction, lasted less than ten minutes.

For all figures, the two sets of lines grouped to extract vanishing points are colored in green and magenta, the red point indicates the calculated vanishing point and the blue line is the associated vanishing line or horizon.

Figures 9, 10, 11 and 12 show classical examples with tiled floors or with regular patterns, where the lines can be clearly extracted. In the latter example (Fig.12), the vanishing points fall outside the canvas in a two point perspective.

In Figure 13, that also contains a tiled floor, no new lines had to be added by the user, only a few lines were removed. In this case, the entire process took less than two minutes.

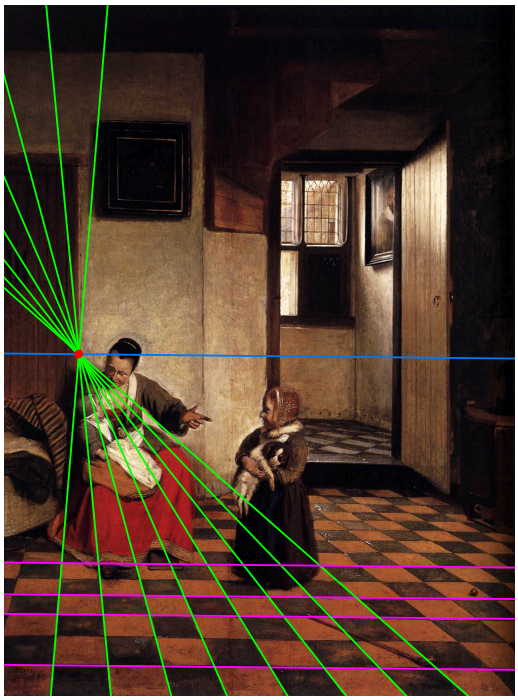


Figure 9: *A Woman With a Baby in Her Lap, Pieter de Hooch. Source: Wikimedia Commons. One-point perspective.*

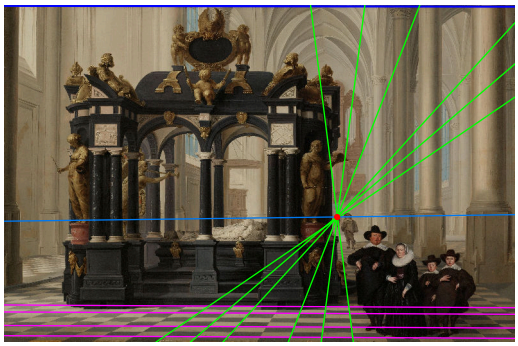


Figure 10: *A Family Beside the Tomb of Prince William I in the Nieuwe Kerk, Dirck van Delen. Courtesy: Rijksmuseum. One-point perspective.*

For The Milkmaid by Vermeer (Figure 14) only the region around the window was selected using the lasso tool. The horizon cannot be extracted since there are no clear horizontal cues. Nevertheless, the window provided enough vertical evidence to extract an almost-vertical vanishing line. Due to the imprecise brushes of the window, it is slightly tilted instead of being precisely vertical.

© 2022 The Author(s)
Eurographics Proceedings © 2022 The Eurographics Association.



Figure 11: *The Courtyard of a House in Delft, Pieter de Hooch. Courtesy: National Gallery of Art, Washington. One-point perspective.*

In order to test the robustness of the method, we ran the process two more times for this same painting varying slightly the threshold parameters. As seen in Figure 15, the vanishing points landed at almost exactly the same location, with a deviation of less than $2mm$. For reference, the painting's dimensions are $41 \times 45.5cm$. Close to the three extracted vanishing points there is a depression attributed to the pin used by Vermeer to employ the chalked string technique. All extracted points are less than $3mm$ from this pin location.

Finally, we ran one more test with the same image. This time the lines were chosen by first clicking on the pin mark and then selecting the lines that passed through the window frame. Figure 16 shows noticeable deviations from the window frame, which explains why it is not possible to achieve perfect accuracy when the vanishing point or pin location is not known beforehand. This example illustrates a major challenge in retrieving perspective information from paintings.

Figure 17 shows a challenging example, where only a couple of non-horizontal lines were available. This is also true for Figure 1, where the painter uses thicker brush strokes and straight lines are not easily identifiable. Figure 18 shows another example of two-point perspective where both aforementioned challenges are present, that is, only a few lines are available and they were very noisy due to the artistic style. Nevertheless, the extracted vanishing line was still very close to the horizontal direction, which was most likely the artist's intention.

Figure 19 illustrates an example where the lines are far from accurate. Even though a vanishing point is computed, it can be noted that no line actually passes through it. However, this is not a flaw of the method but it is rather due to the lines in the painting not being accurate.

Figures 20 and 21 depict a challenging case due to the low image resolution. Albeit the aliased edges the method was still able to extract lines that converge almost in a single point.

As a final example, Figures 22 and 23 show results for two versions of the Last Supper, the original and a reproduction. These paintings were challenging to process, due to the high noise level and imprecise lines in some cases. Nevertheless, the extracted lines are consistent and show that the vanishing point of the reproduction does not perfectly match the one from the original painting by da Vinci.

6. Limitations

The method has a few limitations. First, the threshold parameters still have some influence on the extracted lines. Hence, running the method twice with different thresholds may result in variations of the vanishing points. Second, in some cases, many similar lines may be suggested from a single point and, consequently, it still leaves room for human interpretation. Finally, for obvious reasons, the algorithm only works when there is enough perspective information and gradient information can be sufficiently extracted.

7. Conclusions

We have presented a semi-automatic method to extract perspective information from paintings. More specifically, the method extracts lines that the user can then select and group to retrieve vanishing points and vanishing lines. A few parameters are exposed for the line extraction as well as interactions to add, remove, and group lines.

We have shown results for a diverse set of paintings, ranging in aspects such as image-resolution, scene description, painting style, type of perspective (one or two point) and perspective accuracy. We have also provided comparisons with multiple runs for the same painting.

Since our method can provide a systematic and robust manner to extract perspective information with little user effort, we believe it may serve as a valuable tool to study perspective aspects in paintings. Moreover, the method could be extended to extract radiating lines or other aspects of the composition.

As future work, there are many user interactions that could be improved or added to the system, such as showing the possible lines in a clearer way (with some ranking by significance), allowing the user to indicate rough directions of interest and to backtrack lines from an extracted vanishing point, similarly to what was simulated in Figure 16. Moreover, the method could be extended to handle three point perspective and retrieve multiple horizons, as it is not uncommon for artists to have experimented with this technique in the past [Heu00].

References

[BBS12] BAUMANN R., BLACKWELL C. W., SEALES W. B.: Automatic perspective correction of manuscript images. In *ICADL* (2012). 2

© 2022 The Author(s)
Eurographics Proceedings © 2022 The Eurographics Association.



Figure 12: *The Love Letter, Johannes Vermeer. An example of two point perspective. Courtesy: Rijksmuseum. Two-point perspective.*

[CAA10] CARROLL R., AGARWALA A., AGRAWALA M.: Image warps for artistic perspective manipulation. In *ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, Association for Computing Machinery. URL: https://doi.org/10.1145/1833349.1778864, doi:10.1145/1833349.1778864. 2

[CDI14] CHAUDHURY K., DIVERDI S., IOFFE S.: Auto-rectification of user photos. In *2014 IEEE International Conference on Image Processing (ICIP)* (2014), pp. 3479–3483. doi:10.1109/ICIP.2014.7025706. 2

[CF97] CONNOLLY C., FLIESS T.: A study of efficiency and accuracy in the transformation from RGB to CIELAB color space. *IEEE Trans. Image Processing 6*, 7 (1997), 1046–1048. URL: https://doi.org/10.1109/83.597279, doi:10.1109/83.597279. 3

[CKZ02] CRIMINISI A., KEMP M., ZISSERMAN A.: *Bringing Pictorial Space to Life: computer techniques for the analysis of paintings*. Tech. rep., Microsoft Corporation, 2002. 2

[CM03] CLARK P., MIRMEHDI M.: Rectifying perspective views of text in 3d scenes using vanishing points. *Pattern Recognition 36*, 11 (2003), 2673–2686. doi:https://doi.org/10.1016/S0031-3203(03)00132-8. 2

[CS12] CHANG Y., STORK D.: Warping realist art to ensure consitent perspective: a new software tool for art investigations. In *Human vision and electronic, imaging* (2012). 2

[CT18] CHANG H., TSAI F.: Vanishing point extraction and refinement for robust camera calibration. *Sensors 18*, 1 (2018). URL: https://www.mdpi.com/1424-8220/18/1/63, doi:10.3390/s18010063. 2

[FGV*14] FURFERI R., GOVERNI L., VOLPE Y., PUGGELLI L., VANNI N., CARFAGNI M.: From 2d to 2.5d i.e. from painting to tactile model. *Graphical Models 76*, 6 (2014), 706–723. URL: https://www.sciencedirect.com/science/article/pii/S1524070314000526, doi:https://doi.org/10.1016/j.gmod.2014.10.001. 2

[F.R01] F.R.S. K. P.: Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2*, 11 (1901), 559–572. doi:10.1080/14786440109462720. 4

[GS10] GUTRUF G., STACHEL H.: The hidden geometry in vermeer's 'the art of painting'. *Journal for Geometry and Graphics 14* (2010), 187–202. 2

[HEH05] HOIEM D., EFROS A. A., HEBERT M.: Automatic photo pop-up. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, Association for Computing Machinery, p. 577–584. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/1186822.1073232, doi:10.1145/1186822.1073232. 2

[Heu00] HEUER C.: Perspective as process in vermeer. *Res: Anthropology and aesthetics 38* (2000), 82–99. doi:10.1086/RESv38n1ms20167509. 2, 6

[HZ04] HARTLEY R. I., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, 2004. 2

[KJG09] KALANTARI M., JUNG F., GUEDON J.: Precise, automatic and fast method for vanishing point detection. *The Photogrammetric Record 24*, 127 (2009), 246–263. doi:https://doi.org/10.1111/j.1477-9730.2009.00542.x. 2

[LGvGRM14] LEZAMA J., GROMPONE VON GIOI R., RANDALL G., MOREL J.-M.: Finding vanishing points via point alignments in image primal and dual domains. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 509–515. doi:10.1109/CVPR.2014.72. 2

[LHK09] LEE D. C., HEBERT M., KANADE T.: Geometric reasoning for single image structure recovery. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 2136–2143. doi:10.1109/CVPR.2009.5206872. 2

[LWP*22] LIN Y., WIERSMA R., PINTEA S.-L., HILDEBRANDT K., EISEMANN E., GEMERT J. V.: Deep vanishing point detection: Geometric priors make dataset variations vanish. *CVPR* (2022). https://arxiv.org/abs/2203.08586. URL: http://graphics.tudelft.nl/Publications-new/2022/LWPHEG22. 2

[NS11] NIETO M., SALGADO L.: Simultaneous estimation of vanishing points and their converging lines using the em algorithm. *Pattern Recogn. Lett. 32*, 14 (oct 2011), 1691–1700. URL: https://doi.org/10.1016/j.patrec.2011.07.018, doi:10.1016/j.patrec.2011.07.018. 2

[Rot02] ROTHER C.: A new approach to vanishing point detection in architectural environments. *Image and Vision Computing 20*, 9 (2002), 647–655. doi:https://doi.org/10.1016/S0262-8856(02)00054-9. 2

[Sch00] SCHARR H.: *Optimal operators in digital image processing*. PhD thesis, University of Heidelberg, Germany, 2000. URL: http://archiv.ub.uni-heidelberg.de/volltextserver/volltexte/2000/962/pdf/Diss.pdf. 3

[SFB18] SIMON G., FOND A., BERGER M.-O.: A-contrario horizon-first vanishing point detection using second-order grouping laws. In *Computer Vision – ECCV 2018* (Cham, 2018), Ferrari V., Hebert M., Sminchisescu C., Weiss Y., (Eds.), Springer International Publishing, pp. 323–338. 2

[Sim21] SIMON G.: Jan van eyck's perspectival system elucidated through computer vision. *Proc. ACM Comput. Graph. Interact. Tech. 4*, 2 (aug 2021). doi:10.1145/3465623. 2

[Sto04] STORK D. G.: Were optical projections used in early Renaissance painting? A geometric image analysis of Jan van Eyck's "Arnolfini

© 2022 The Author(s)
Eurographics Proceedings © 2022 The Eurographics Association.

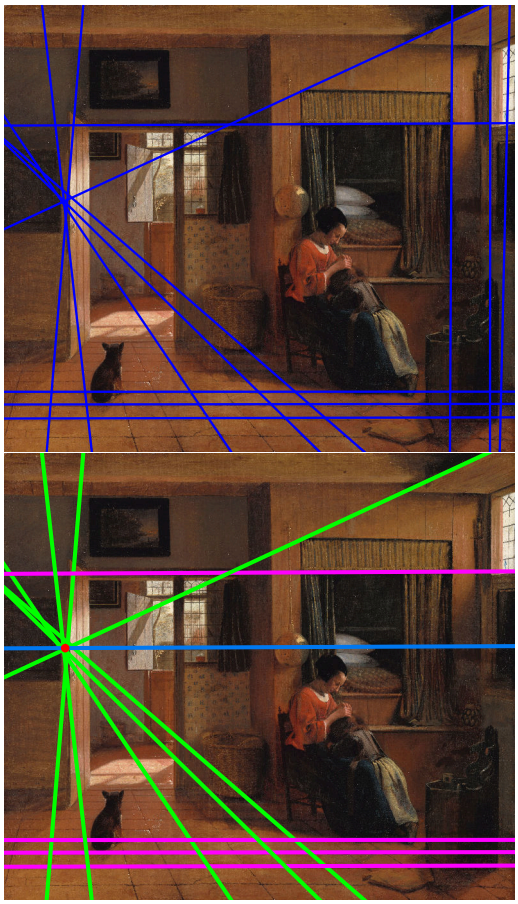


Figure 13: *A Mother Delousing her Child's Hair, Pieter de Hooch. On top the initial set of lines displayed by the method. On the bottom, the final result after removing some vertical lines and grouping the remaining lines to find the vanishing points. Courtesy: Rijksmuseum*



Figure 14: *The Milkmaid, Johannes Vermeer. Courtesy: Rijksmuseum. One-point perspective without a horizon.*

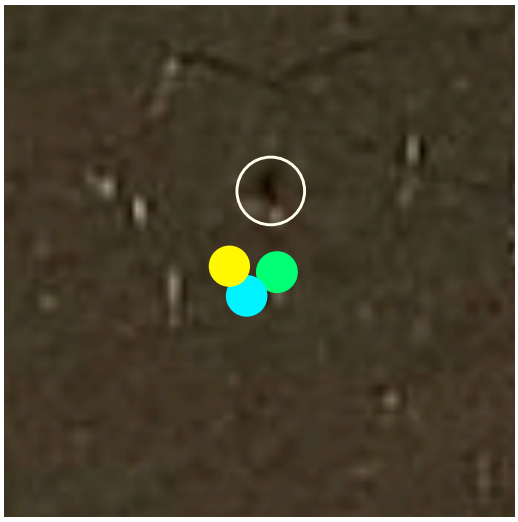


Figure 15: *Comparison of vanishing point found in three different runs, represented by the yellow, green and cyan circles. The white circle indicates the probable depression caused by the pin. The Milkmaid, Johannes Vermeer. Courtesy: Rijksmuseum*

Portrait" and Robert Campin's "Mérode Altarpiece". In *Vision Geometry XII* (2004), Latecki L. J., Mount D. M., Wu A. Y., (Eds.), vol. 5300, International Society for Optics and Photonics, SPIE, pp. 23 – 30. URL: https://doi.org/10.1117/12.524193, doi:10.1117/12.524193. 2

[Sto06] STORK D.: Computer vision, image analysis, and master art: part 1. *IEEE MultiMedia 13*, 3 (2006), 16–20. doi:10.1109/MMUL.2006.50. 2

[Sto09] STORK D. G.: Computer vision and computer graphics analysis of paintings and drawings: An introduction to the literature. In *Computer Analysis of Images and Patterns* (Berlin, Heidelberg, 2009), Jiang X., Petkov N., (Eds.), Springer Berlin Heidelberg, pp. 9–24. 2

[Tar09] TARDIF J.-P.: Non-iterative approach for fast and accurate vanishing point detection. In *2009 IEEE 12th International Conference on Computer Vision* (2009), pp. 1250–1257. doi:10.1109/ICCV.2009.5459328. 2

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *ICCV* (1998), pp. 839–846. URL: https://doi.org/10.1109/ICCV.1998.710815, doi:10.1109/ICCV.1998.710815. 3

[vdH98] VAN DEN HEUVEL F. A.: Vanishing point detection for architectural photogrammetry. In *International Archives of Photogrammetry and Remote Sensing* (1998). 2

[VDS14] VOUZOUNARAS G., DARAS P., STRINTZIS M. G.: Automatic generation of 3d outdoor and indoor building scenes from a single image. *Multimedia Tools and Applications 70* (2014), 361–378. doi:https://doi-org.tudelft.idm.oclc.org/10.1007/s11042-011-0823-. 2

© 2022 The Author(s)
Eurographics Proceedings © 2022 The Eurographics Association.



Figure 16: *Selecting lines by clicking on the pin mark. The detailed image on the right illustrates that not all lines pass exactly through the window frame. The Milkmaid, Johannes Vermeer. Courtesy: Rijksmuseum*

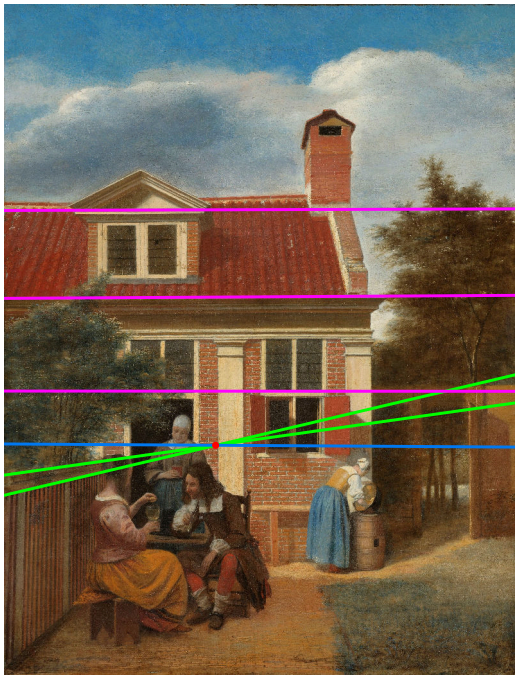


Figure 17: *Figures in a Courtyard behind a House, Pieter de Hooch. Courtesy: Rijksmuseum. Challenging case with only a few lines available.*

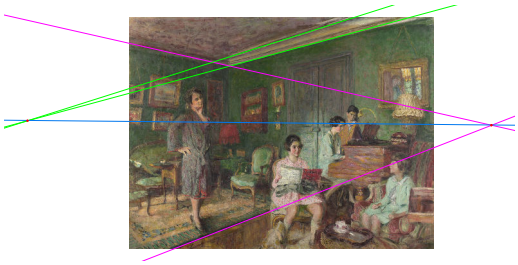


Figure 18: *Madame André Wormser and her Children, Edouard Vuillard. Courtesy: National Gallery of Art, Washington. Challenging case with only a few lines available and noisy edges.*

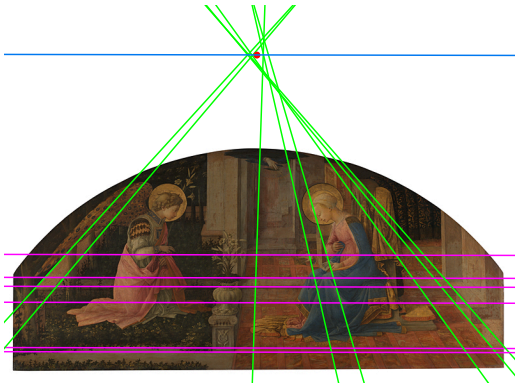


Figure 19: *The annunciation, Fra Filippo Lippi. Courtesy: National Gallery of Art, Washington. Example of painting with imprecise perspective lines.*

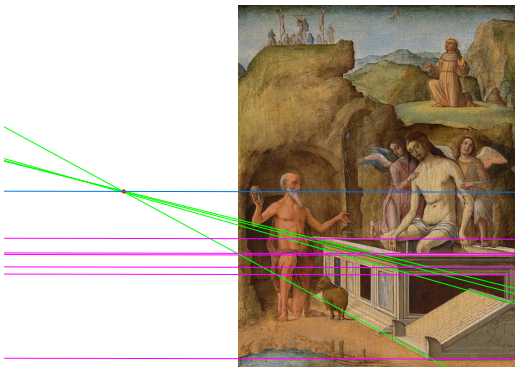


Figure 20: *The Dead Christ, Ercole de' Roberti. Courtesy: National Gallery of Art, Washington. Example of method working with a low-resolution image.*

© 2022 The Author(s)
Eurographics Proceedings © 2022 The Eurographics Association.

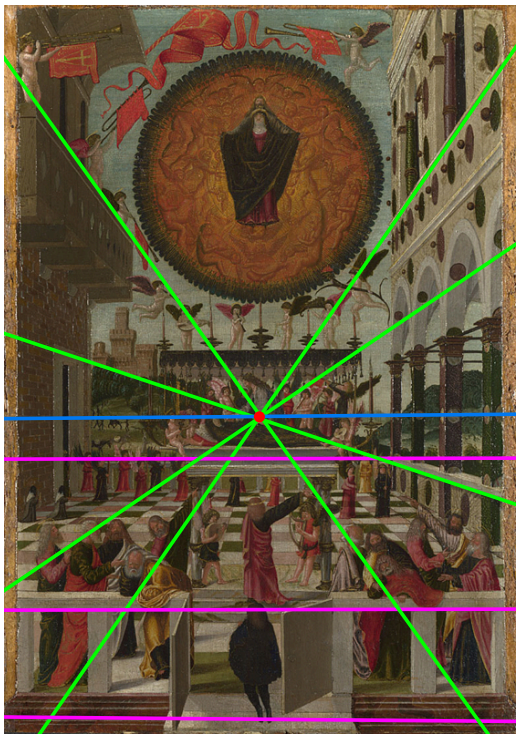


Figure 21: *The Dormition and Assumption of the Virgin, Gerolamo da Vicenza. Courtesy: National Gallery of Art, Washington. Example of method working with a low-resolution image.*

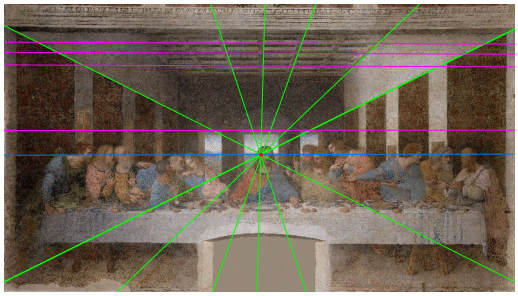


Figure 22: *Last Supper, Leonardo da Vinci. Source Wikipedia. Challenging example due to noisy edges.*

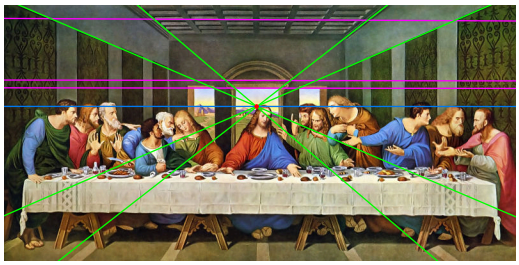


Figure 23: *Last Supper, reproduction by unknown artist. Source Wikimedia Commons. Challenging example due to noisy edges. The vanishing point does not precisely match the one from the original painting.*

© 2022 The Author(s)
Eurographics Proceedings © 2022 The Eurographics Association.