Perspective click-and-drag area selections in pictures

Frank Nielsen
Sony Computer Science Laboratories Inc
3-14-13 Higashi Gotanda, 141-0022 Shinagawa-Ku, Tokyo, Japan
Frank.Nielsen@acm.org

Abstract

Pictures of man-made environments often exhibit many perspectively slanted planar parts like buildings, road or shop signages. In this paper, we present a novel user interface for area selections in pictures using perspective click and drag operations. At preprocessing stage, our system first recognizes convex quads from multi-scale segmentations that it interprets as perspective rectangles. It then builds a nested quad hierarchy labeled with quad-to-square homographies and inverse transformations. During the interactive session, whenever a user selects an area by first clicking a corner and then dragging the diagonally opposite corner, the system proposes a perspective sub-rectangle matching the dragged diagonal defined according to the homographies of the selected quad of the hierarchy. We illustrate the perspective click'n'drag prototype with common image editing operations.

1 Introduction

The click-and-drag rectangular area selection[1] is often used when editing images. Many pictures of man-made environments display perspectively slanted planar parts where traditional rectangular area selection fails to appropriately select regions as depicted in Figure 1. To overcome this problem, we propose a *perspective dragging* user interface. The closest work to our perspective dragging UI is the graffiti annotation system [3] in videos that paints on a planar surface such as a taxi door or ground plane by tracking manually input anchor tracks.

The paper is organized as follows: Section 2 describes the preprocessing stage of our system that first recognizes convex quads (interpreted as perspective rectangles) from multi-scale image segmentations, yielding a set of unorganized quads. Section 3 shows how to build from the quad soup a nested quad hierarchy labeled with homographies of quads to the unit square. Section 4 presents the operation work flow during the interactive session: Namely, whenever a user selects an area by first clicking a corner and then dragging the diagonally opposite corner, the system proposes a perspective sub-rectangle exactly matching the dragged diagonal defined according to the selected homographies of the quad hierarchy. Section 5 discusses on some common image operations using the perspective click'n'drag UI. Finally, Section 6 discusses on current limitations and hint at further extensions of the system.

[1]See for example, the on-line demo at `http://odyniec.net/projects/imgareaselect/`

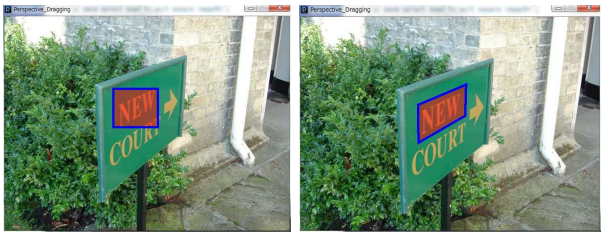

Figure 1. Conventional rectangular (left) versus perspective (right) dragging. Rectangular dragging fails to properly select the `NEW` word part on the panel without portion of `COURT` word because those two parts are not separable by an horizontal line.

2 Convex quad detection

At the preprocessing stage when loading an image, the system first detects convex quads. Each convex quad is interpreted as a potential rectangle imaged by a pinhole camera under perspective projection. In the remainder, we omit to mention the convex property as all considered quads are convex. Those perspective rectangles can be detected automatically using various computer vision techniques. For example, one can apply an edge detection filter followed by a Hough transform, and use additional information like the vanishing point to infer proper quads, as described in [8]. Similarly, Zhang and He [10] proposed a system for capturing with a digital camera the contents of a white-board based again on the Hough transform.

Because, we need to detect potentially many quads in a cluttered environments, we implemented the following detection technique: First, we segment the image using the fast statistical region merging algorithm described in [6]. Then for each *exterior contour* of a segmented region not touching the image boundary, we first compute the contour diameter [5] $[p_1p_3]$, and define the convex quad by choosing point p_2 as the farthest contour point above $[p_1p_3]$, and similarly point p_4 as the farthest contour point below $[p_1p_3]$. We then calculate the Hausdorff distance between the exterior contour to its convex quad approximation, and compare it with a prescribed threshold. When the contour-quad approximation distance falls below a threshold (eg., 5% of diameter) and its area is large enough, we accept the quad as a perspective rectangle. Since perspective rectangles can be nested in pictures[2], we run the segmentation algorithm and detect quads at different scales. More importantly, since segmentation algorithms are usually not perfect!, we can also use various other image segmentation techniques (like mean-shift [11] or normalized cuts [9]) to get a *super-set* of quads. Fi-

[2]For example, think of books and boxes lying on a table.

nally, we add the image rectangular boundary itself to the convex quad soup $\mathcal{Q}$. We next describe how to select a proper subset of quads from this unorganized collection of quads.

3 Nested convex quad hierarchy

We first sort the quads in decreasing order of their area (see Section A). Then we build a nested quad hierarchy $\mathcal{H}$ *greedily* by selecting convex quads that do not partially overlap with the quads already inserted in the hierarchy. That is, we test for a quad Q whether it intersects partially a quad of the hierarchy or not (Section A). It follows that we necessarily pick up the rectangular bounding box of the image as the root convex quad of $\mathcal{H}$. When adding a quad Q to $\mathcal{H}$, we label the node with the projective transformations of the convex quad to a unit square U: Let us assume that we are given a set of n convex quads $\{Q_i = (p_{1,i}, p_{2,i}, p_{3,i}, p_{4,i})\}_{i=1}^{n}$. We label the quad corners so that $p_{1,i}$ is the leftmost point of Q_i, and $(p_{1,i}, p_{2,i}, p_{3,i}, p_{4,i})$ is oriented clockwise (Section A). For each quad Q_i, we compute the 3×3 homography matrix [4] H_i mapping Q_i to the unit square, and its inverse H_i^{-1}. An homography describes the relationships of a plane imaged by perspective projection under two different viewpoints. To map a pixel $p = (x, y)$ belonging to a plane imaged in image I to the corresponding pixel $p' = (x', y')$ in image I', we use homogeneous coordinates [4]. That is, we add an extra coordinate w, and lift $p = (x, y)$ to $\tilde{p} = (x, y, w)$ with $w = 1$. The vector points with a $\tilde{}$ notation are called the homogeneous coordinates of points. The homography [4] relates corresponding pixels as follows:

$$\tilde{p}' = H\tilde{p}.$$

We dehomogeneize $\tilde{p}'$ to find its pixel position p' (inhomogeneous coordinates) in I' by performing the perspective division:

$$\tilde{p}' = (x', y', w') \rightarrow p' = \left(\frac{x'}{w'}, \frac{y'}{w'}\right).$$

We use the normalized DLT technique (described in Section B) for computing a homography given four point correspondences (here, for the quad corners Q_i matching the unit square U). However, note that we *do not explicitly unwarp the quads onto unit images* in contrast to many Augmented Reality (AR) systems [7].

4 Perspective click'n'drag

4.1 Perspective sub-rectangle selection

The regular area dragging selection consists in clicking on a corner p_1 and dragging the opposite corner p_3. The diagonal defines an axis-parallel bounding box $Q = (p_1, p_2, p_3, p_4)$. Perspective dragging similarly relies on the fixed corner p_1 and dragged corner p_3 to define a perspective sub-rectangle with one diagonal coinciding with the line segment $[p_1 p_3]$. We find the deepest quad Q in the hierarchy $\mathcal{H}$ that contains both points p_1 and p_3. We then compute by the attached homography, the points $\tilde{p}'_1$ and $\tilde{p}'_3$ in the normalizing unit square using homogeneous coordinates:

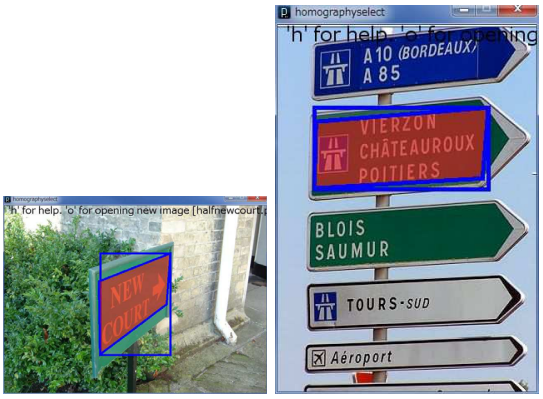

Figure 2. The rectangular quads indicate the traditional area selected by a click-and-drag interaction. The perspective quad indicates the perspective sub-rectangle selection.

$$\tilde{p}'_1 = H\tilde{p}_1 = \begin{bmatrix} x_1 \\ y_1 \\ w_1 \end{bmatrix}, \qquad \tilde{p}'_3 = H\tilde{p}_3 = \begin{bmatrix} x_3 \\ y_3 \\ w_3 \end{bmatrix},$$

where $\tilde{p}_1 = (x_1, y_1, w_1 = 1)$ and $\tilde{p}_3 = (x_3, y_3, w_3 = 1)$. We obtain the points p'_1 and p'_3 in inhomogeneous coordinates as:

$$p'_1 = \left(\frac{x'_1}{w'_1}, \frac{y'_1}{w'_1}\right), \qquad \tilde{p}'_3 = \left(\frac{x'_3}{w'_3}, \frac{y'_3}{w'_3}\right).$$

Let Q' denote the axis-parallel bounding box of p'_1 and p'_3, we map back this orthogonal quad onto the image by using the inverse homography H^{-1}:

$$S \leftarrow H^{-1}Q',$$

where S denotes the perspective selection quad, and MQ the quad obtained by mapping by some homography M the four points of Q. Figure 2 shows some quad selection obtained with this method. Note that since we shall select perspective sub-rectangles, slight errors in the homography estimation does not impact significantly the area selection procedure. Since the diagonal $[p_1 p_3]$ is a diagonal of the selection quad S, we described a more efficient implementation.

4.2 An efficient implementation

Without loss generality, let us assume that p_1 is located to the left of p_3 (i.e., $x_1 \leq x_3$, or otherwise relabel those points by swapping them). We need to define the two remaining corners p_2 and p_3 of the perspective sub-rectangle such that the quad $S = (p_1, p_2, p_3, p_4)$ is oriented clockwise. First, we calculate the mapping of p_1 and p_3 onto the unit square by the homography H:

$$p'_1 = (x'_1, y'_1) \xleftarrow{\text{dehomogeneize}} \tilde{p}'_1 = H\tilde{p}_1,$$
$$p'_3 = (x'_3, y'_3) \xleftarrow{\text{dehomogeneize}} \tilde{p}'_3 = H\tilde{p}_3,$$

Second, we define points p'_2 and p'_4 such the quad (p'_1, p'_2, p'_3, p'_4) forms a rectangular bounding box with diagonal $[p'_1 p'_3]$. To express those point coordinates, let us introduce the maxima/minima coordinates:

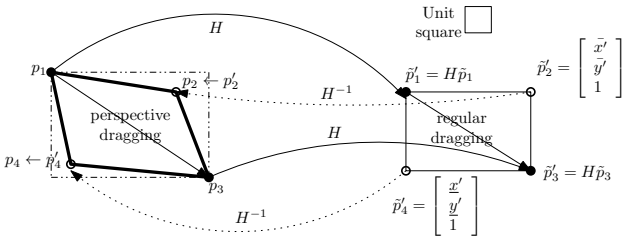

Figure 3. Perspective dragging: The user selects p_1 and drag p_3. The system maps p_1 and p_3 to p'_1 and p'_3 using the predefined convex quad-to-square homography H. It then makes a rectangular bounding box (p'_1, p'_2, p'_3, p'_4) with opposite corners p'_1 and p'_3, and maps back p'_2 and p'_4 to the image coordinate system using the inverse square-to-convex quad homography H^{-1}.

$$\begin{aligned} \underline{x'} &= \min\{x'_1, x'_3\}, \bar{x'} = \max\{x'_1, x'_3\}, \\ \underline{y'} &= \min\{y'_1, y'_3\}, \bar{y'} = \max\{y'_1, y'_3\} \end{aligned}$$

If $y'_3 < y'_1$ (see Figure 3), points p'_2 and p'_4 have coordinates:

$$\begin{aligned} p'_2 &= (\bar{x'}, \bar{y'}), \quad \tilde{p}'_2 = (\bar{x'}, \bar{y'}, 1) \\ p'_4 &= (\underline{x'}, \underline{y'}), \quad \tilde{p}'_4 = (\underline{x'}, \underline{y'}, 1). \end{aligned}$$

Otherwise (i.e., $y'_3 > y'_1$), points p'_2 and p'_4 have coordinates:

$$\begin{aligned} p'_2 &= (\underline{x'}, \bar{y'}), \quad \tilde{p}'_2 = (\underline{x'}, \bar{y'}, 1) \\ p'_4 &= (\bar{x'}, \underline{y'}), \quad \tilde{p}'_4 = (\bar{x'}, \underline{y'}, 1). \end{aligned}$$

We get back the corresponding points p_2 and p_4 for the source image by applying the inverse homography transformation:

$$\begin{aligned} p_2 &\xleftarrow{\text{dehomogeneize}} H^{-1}\tilde{p}'_2, \\ p_4 &\xleftarrow{\text{dehomogeneize}} H^{-1}\tilde{p}'_4. \end{aligned}$$

Finally, we draw the perspective sub-rectangle area selection $S = (p_1, p_2, p_3, p_4)$. This work flow is illustrated in Figure 3. Note that the perspective dragging operation does not require to perform explicit image unwarping.

5 Some image editing applications

Perspective dragging is the natural extension of dragging for perspectively slanted portions in pictures. Indeed, when the deepest quad containing both points p_1 and p_3 is the root quad, we find the usual rectangular area selection procedure. This novel UI finds broad applications for annotating planar areas in social picture networks, and for performing sub-image internet queries, etc. We describe below the convenient swapping image editing operation (beyond the cut/copy/paste/inpaint operations). A swap operation is the exchange of the contents of two convex quads. Once, the first convex quad Q_1 is selected, it is swapped with another selected quad Q_2. We need an extra buffer image to keep the pixel contents of Q_1 while rewriting it with the pixel contents of Q_2. The backward mapping proceeds by inspecting all pixels of Q_1. Let H_1 denote the homography from Q_1 to U (the unit square), and H_2 the homography from Q_2 to U. We define the homography H_{12} from Q_1 to Q_2 by composition $H_{12} = H_1 H_2^{-1}$, and its inverse $H_{21} = H_{12}^{-1} = H_2 H_1^{-1}$. Figure 4 displays the result of a swap operation in a city image.

Figure 4. Swap operation between two quads.

6 Conclusion and perspectives

We proposed a novel interactive system for perspective area selection in pictures. At preprocessing stage, the system detects a super-set of convex quads and builds a hierarchical nested quad structure with quad-to-square homography and inverse transformations attached to each node. Then during the interactive session, from the 2-point click-and-drag events, we localize the deepest quad in the hierarchy and based on its attached homography transformations we propose a corresponding sub-rectangle selection. Perspective click'n'dragging is useful for image editing and annotations. One limitation of the system is related to the performance of the perspective rectangle detection algorithms. Ongoing work consists in setting a benchmark like BSDS500 did for image segmentation [1]. However, the main difference with [1] to our advantage is that we allow several recognition algorithms to run at preprocessing stage to get a quad soup (preprocessing can be operated remotely on a cloud infrastructure). It remains for the system to cope with quads touching the image boundary: In that case we need to estimate the homography not from its bounding corners but rather from texture deformation information, for example. Finally, let us conclude by mentioning two extensions: First, we would like to compute simultaneously more robustly the convex quad-to-square homographies by using the homography manifold constraints [2] (and also find the best "unit" for the unit square). Second, we would like to consider other perspective click'n'drag operations like detecting ellipsoids on the images and reinterpreted them as perspective balls (click-and-drag user selection then can be diameter-based, or center-radius-based, etc.).

References

[1] Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.

[2] Anders Eriksson and Anton van den Hengel. Optimization on the manifold of multiple homographies.

pages 24 –249, 2009.

[3] Dan B. Goldman, Chris Gonterman, Brian Curless, David Salesin, and Steven M. Seitz. Video object annotation, navigation, and composition. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 3–12, New York, NY, USA, 2008. ACM.

[4] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[5] Grégoire Malandain and Jean-Daniel Boissonnat. Computing the diameter of a point set. In *Discrete Geometry for Computer Imagery (DGCI)*, pages 197–208, 2002.

[6] Richard Nock and Frank Nielsen. Statistical region merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1452–1458, 2004.

[7] Michael Rohs and Christof Roduner. Camera phones with pen input as annotation devices. In *Pervasive workshop on Pervasive Mobile Interaction Devices (PERMID)*, pages 23–26, Munich, Germany, 2005.

[8] David Shaw and Nick Barnes. Perspective rectangle detection. *Proceedings of the Workshop of the Application of*, pages 1–152, 2006.

[9] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, pages 731–737, Washington, DC, USA, 1997. IEEE Computer Society.

[10] Zhengyou Zhang and Li wei He. Whiteboard scanning and image enhancement. *Digital Signal Processing*, 17(2):414–432, 2007.

[11] Huiyu Zhou, Xun Wang, and Gerald Schaefer. Mean shift and its application in image segmentation. In Halina Kwasnicka and Lakhmi Jain, editors, *Innovations in Intelligent Image Analysis*, volume 339 of *Studies in Computational Intelligence*, pages 291–312. Springer Berlin / Heidelberg, 2011.

A Primitive on convex quads

Perspective dragging requires to detect whether a point falls inside a convex quad $Q = (p_1, p_2, p_3, p_4)$, or not. Let the quad Q be ordered in the clockwise order. To determine whether a query point p (pixel) is inside Q or not, we check that the following four triples of points are all ordered clockwise or not:(p_1, p_2, p), (p_2, p_3, p), (p_3, p_4, p), (p_4, p_1, p). We check that that a triple of points (p_1, p_2, p_3) are oriented clockwise by computing the sign of a 2×2 determinant:

$$\det = \left| \begin{bmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{bmatrix} \right| \quad (1)$$

The triple is oriented clockwise if and only if $\det > 0$.

The area of a convex quad (p_1, p_2, p_3, p_4) is required when sorting detected quads for building the hierarchical quad tree. This area is calculated by summing up the area of the two triangles (p_1, p_2, p_3) and (p_3, p_4, p_1) of its partition. The area of a triangle (p_1, p_2, p_3) can be calculated as one half of the absolute value of the determinant of the 2×2 matrix of Eq. 1.

When building the hierarchical tree, we need to test whether a convex quad q is fully inside another convex quad q'. This amounts to test whether the 4 vertices of q is inside q' using the former predicate.

B Homography via normalized DLT

We recall the most common technique for estimating a homography from a set of pairs of point correspondences using the *Direct Linear Transform* (DLT) algorithm [4]. We assume that we are given at least $n \geq 4$ pairs of point correspondences $p_i \leftrightarrow p'_i$, and that the points are in non-degenerate position. That is, there are no three collinear points. We write the n correspondence constraints $p_i \leftrightarrow p'_i$ using the 3×3 homography matrix:

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

with the homogeneous vectors $\tilde{p}_i = \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ w_i \end{bmatrix}$ and $\tilde{p}'_i = \begin{bmatrix} \tilde{x}'_i \\ \tilde{y}'_i \\ w'_i \end{bmatrix}$: $\tilde{p}'_i = H\tilde{p}_i$. Using the inhomogeneous coordinates (with $w'_i = h_{31}x_i + h_{32}y_i + h_{33}w_i$), we get the following constraints: $x'_i = \frac{h_{11}x_i + h_{12}y_i + h_{13}w_i}{h_{31}x_i + h_{32}y_i + h_{33}w_i}$, $y'_i = \frac{h_{21}x_i + h_{22}y_i + h_{23}w_i}{h_{31}x_i + h_{32}y_i + h_{33}w_i}$. Setting $w_i = 1$, and rearranging those equations, we end up with:

$$\begin{aligned} x'_i(h_{31}x_i + h_{32}y_i + h_{33}) &= h_{11}x_i + h_{12}y_i + h_{13}, \\ y'_i(h_{31}x_i + h_{32}y_i + h_{33}) &= h_{21}x_i + h_{22}y_i + h_{23}. \end{aligned}$$

Those equations are linear in the homography coefficients h_i, and can be written compactly by vectorizing the coefficients as $A_i h = 0$, with $A_i =:$

$$\begin{bmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & x_i x'_i & x'_i y_i & x'_i \\ 0 & 0 & 0 & -x_i & -y_i & -1 & x_i y'_i & y_i y'_i & y'_i \end{bmatrix}$$

and $h = [h_{11}\ h_{12}\ h_{13}\ h_{21}\ h_{22}\ h_{23}\ h_{31}\ h_{32}\ h_{33}]^\top = [h_1\ h_2\ h_3\ h_4\ h_5\ h_6\ h_7\ h_8\ h_9]\top$.

By stacking the n two row constraints A_i, we obtain a matrix A of size $2n \times 9$. Solving the homogeneous least square equation $Ax = 0$ amounts to find the null space vector of A. Thus we perform the singular value decomposition (SVD) [4]: $A = UDV^T = \sum_{i=1}^{9} \lambda_i u_i v_i^T$, and choose the right eigenvector of V corresponding to the smallest eigenvalue. Since eigenvalues are usually sorted in decreasing order in the diagonal matrix D, that means that we choose the last column vector v_9 of V. When $\lambda_9 = 0$, the system is exactly determined. When $\lambda_9 > 0$, the system is over-determined and λ_9 is an indicator of the goodness of fit of the solution $h = v_9$. Finally, we rearrange $h = v_9$ into a 3×3 matrix H. In practice, this estimation procedure is *highly* unstable numerically. Therefore the points need to be first *normalized* to that their centroid defines the origin, and the diameter is set to $\sqrt{2}$. This is the *normalized DLT* procedure [4].