Perspective click-and-drag area selections in pictures

Frank NIELSEN
www.informationgeometry.org

Sony Computer Science Laboratories, Inc.

Machine Vision Applications (MVA)
21st May 2013

Traditional click and drag rectangular selection

→ Fails for selecting parts in photos:

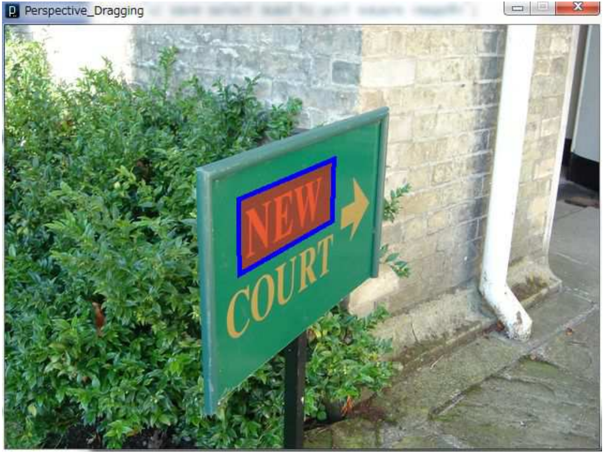
Traditional click and drag rectangular selection

→ Fails for selecting parts in photos:
Cannot capture "New" without part of "Court".

Man-made environments: many perspectively slanted planar parts.

Perspective click'n'drag

Intelligent UI (= computer vision + human computer interface)



→ Image "parsing" of perspective rectangles (automatic/semi-automatic/manual)

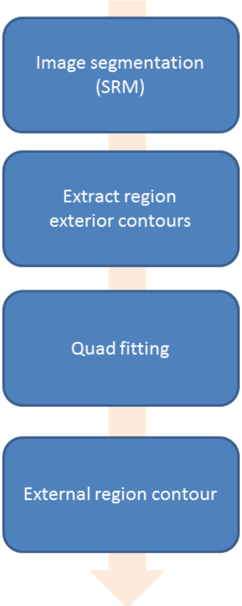
Video demonstrations

Perspective click-and-drag + perspective copy/paste/swap

Perspective click'n'drag: Outline

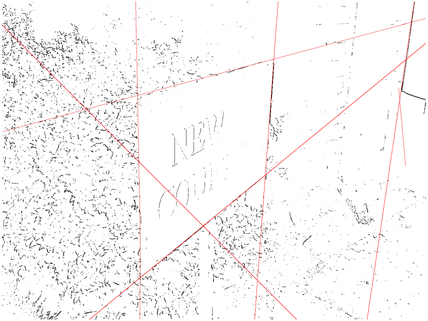
1. Preprocessing:
   Detect & structure perspective parts
   1.1 Quad detector:
       - Image segmentation
       - Outer contour quad fitting
       - Quad recognition
   1.2 Quad homography tree
2. Interactive user interface:
   Perspective quad selection based on click-and-drag UI (=diagonal selection)
3. Application example: Interactive image editing (swap)

Preprocessing workflow



Quad detection: Sobel/Hough transform

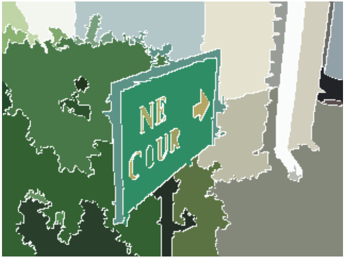
How to detect convex quads in images?
indoor robotics [6] using vanishing point.
Limitations of Hough transform [8] on Sobel image:

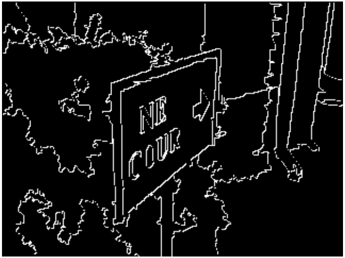


Combinatorial line arrangement $O(n^4)$...
$\rightarrow$ good for limited number of detected lines
(blackboard detection [8], name card detection, etc.)

Quad detection: Image segmentation (SRM)

→ Fast Statistical Region Merging [4] (SRM)



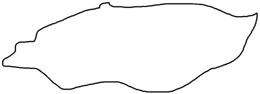


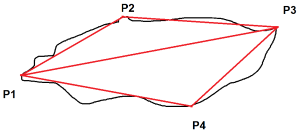
Source codes in Java™, Matlab®, Python®, C, etc.

Quad detection: Image segmentation (SRM)

Quad detector

- For each segmented region, consider its exterior contour C (polygon),
- Compute the contour diameter, $P_1 P_3$,
- Compute the upper most P_2 and bottom most P_4 extremal points
- Calculate the symmetric Haussdorf distance between quad $Q = (P_1, P_2, P_3, P_4)$ and contour C,
- **Accept** region as quad when distance falls below as prescribed threshold.





All quads convex and **clockwise oriented**.

Quad detection: Image segmentation (SRM)

... any closed contour image segmentation,

$\rightarrow$ run at different scales (eg., parameter Q in SRM).

Alternatively, can also use mean-shift [9], normalized cuts [7], etc.

Why? To increase the chance of detecting for some parameter tuning quads.

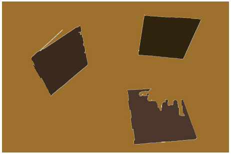
$\rightarrow$ We end up with a **quad soup**

Multi-segmentation

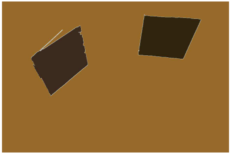
Increases the chance of recognizing quads, but get a quad soup.

$Q = 128$

$Q = 10$



$Q = 0.3$



$Q = 0.25$

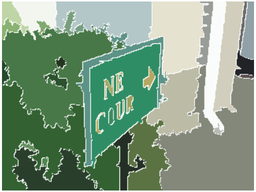
Nested convex quad hierarchy

- From a quad soup, sort the quads in *decreasing order* of their **area** in a priority queue.
- Add image boundary quad Q_0 as the **quad root** of the quad tree $\mathcal{Q}$.
- Greedy selection: Add a quad of the queue if and only if it is fully contained in another quad of $\mathcal{Q}$.
- When adding a quad Q_i, compute the homographies [2] H_i and H_i^{-1} of the quad to the unit square.

Do not explicit unwarp perspective rectangles

Many existing systems first unwarp...

source



segmented

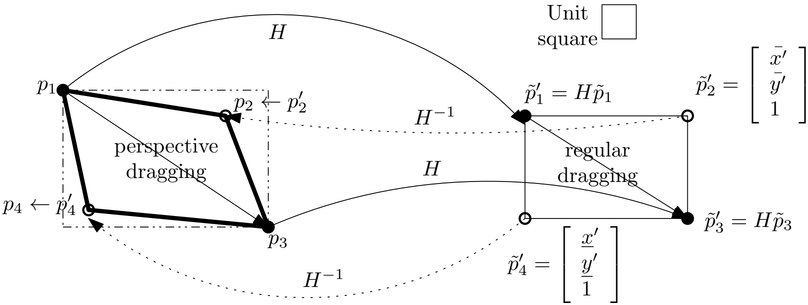
unwarped

Mobile cell phone signage recognition [5], AR systems, etc.

Perspective click'n'drag: User interaction

Perspective sub-rectangle selection:
Clicking on a corner p_1 and **dragging** the opposite corner p_3.

find the deepest quad Q in the quad hierarchy $\mathcal{Q}$ that contains **both** points p_1 and p_3.



Some examples of perspective click-and-drag selections

Regular vs. perspective rectangle UI selection

Implementation details: Primitives on convex quads

By convention, order quads clockwise.
Positive determinant for the two quad-induced triangles:

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

- Predicate $p \in Q = (p_1, p_2, p_3, p_4)$?:
  Two queries: $p \in (p_1, p_2, p_3)$ and $p \in (p_3, p_4, p_1)$.
- Area of a quad:
  One half of the absolute value of the determinant of the two quad triangles.

In class Quadrangle

```
double area(Feature p1, Feature p2, Feature p3)
{
  double res;
  res=(p1.x-p3.x)*(p2.y-p1.y)-(p1.x-p2.x)*(p3.y-p1.y);
  return 0.5*Math.abs(res); // half of determinant
}

double area()
{
return (area(p1,p2,p3)+area(p1,p3,p4));
}

//
// Clockwise or aligned order predicate
//
boolean CW(Feature a, Feature b, Feature c)
{
double det=(a.x-c.x)*(b.y-c.y)-(b.x-c.x)*(a.y-c.y);
if (det>=0.0)
    {return true;}
  else
    {return false;}
}

// Determine if a pixel falls inside the quadrangle or not
boolean inside(int x, int y)
{
Feature p=new Feature(x,y,1.0);

if ( CW(p1,p2,p) && CW(p2,p3,p) && CW(p3,p4,p) && CW(p4,p1,p) )
  {return true;}
else
  {return false;}
}
```

Homography estimation

Projective geometry, homogeneous and inhomogeneous coordinates.

$$\tilde{p}'_i = \begin{bmatrix} \tilde{x}'_i \\ \tilde{y}'_i \\ w'_i \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ w_i \end{bmatrix} = H\tilde{p}_i,$$

$$w'_i = h_{31}x_i + h_{32}y_i + h_{33}w_i$$

$x'_i = \frac{h_{11}x_i+h_{12}y_i+h_{13}w_i}{h_{31}x_i+h_{32}y_i+h_{33}w_i}$, $y'_i = \frac{h_{21}x_i+h_{22}y_i+h_{23}w_i}{h_{31}x_i+h_{32}y_i+h_{33}w_i}$.
A_i block matrix:

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

Solve for $A_i h = 0$

Homography estimation using inhomogeneous system

Assume $h_{33} \neq 0$ (and set $h_{33} = 1$).

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix} \times \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix}}_{h'} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix}$$

Linear system written:

$$Bh' = b.$$

For four pairs

$$h' = B^{-1}b$$

Homography estimation using the normalized DLT

$$H = UDV^T = \sum_{i=1}^{9} \lambda_i u_i v_i^\top,$$

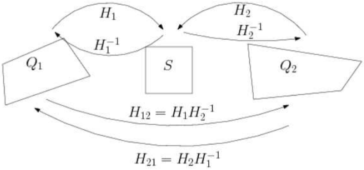
Right eigenvector of V corresponding to the smallest eigenvalue. (last column vector v_9 of V)

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 $h = v_9$.
In practice, this estimation procedure is *highly* unstable numerically[2].
Points need to be first *normalized* to that their centroid defines the origin, and the diameter is set to $\sqrt{2}$.

Image editing: Selection swaps

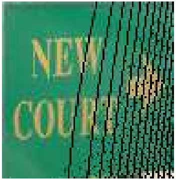
H_{12} from Q_1 to Q_2 by composition:

$$H_{12} = H_1 H_2^{-1}$$

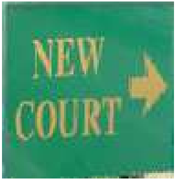


$$H_{21} = H_{12}^{-1} = H_2 H_1^{-1}$$

$\rightarrow$ backward pixel mapping [3] (avoid holes)



forward mapping
SRC$\rightarrow$ DEST (H)



backward mapping
DEST$\rightarrow$SRC (H^{-1})

Image editing: Selection swaps

Image editing: Selection swaps

Image editing: Selection swaps

Image editing: Selection swaps

Perspective Click-and-Drag UI: Conclusion

- Simple **UI** system relying on **computer vision**.
- Extend to other input formats: Stereo pairs, RGBZ images, etc.
- Implemented using `processing.org` (2500+ lines)

Ongoing work:

- Rely on efficient *quad detection*: extensive benchmarking (BSDS500, Corel, ImageNet, etc. databases)
- Extend to various **perspectively slanted shapes** (like ball $\rightarrow$ ellipsoids, etc.)
- Robust **multiple** quad-to-square homography estimations [1]?

`www.informationgeometry.org`

Bibliographic references I

Anders Eriksson and Anton van den Hengel.
Optimization on the manifold of multiple homographies.
pages 24 –249, 2009.

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

Frank Nielsen.
Visual Computing: Geometry, Graphics, and Vision.
Charles River Media / Thomson Delmar Learning, 2005.

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

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.

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

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.

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

Bibliographic references II

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.