RoboCup 2010

5. Line Detection / Perspective Transform

Content

Finding a Line in an Image (1)

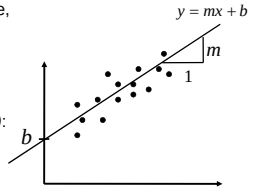
- If a set of points (x_i, y_i) forms a line, we can use simple linear regression for fitting the optimal line.
- The slope *m* and the offset *b* can be computed from all points (x_i, y_i):

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$b = \bar{y} - m\bar{x}$$

$\bar{x}$ = average value of all x_i

$\bar{y}$ = average value of all y_i



- Vertical lines → use vector representation
- The slope *m* and the offset *b* can be computed from all points (x_i, y_i):

Finding a Line in an Image (2)

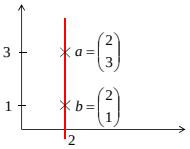
Challenges

- Unknown number of lines
- Outliers (single uncorrelated pixels)
- Lines may be interrupted
- Vertical lines: $y = mx + n$

Finding a Line in an Image (3)

Vertical lines

- Use vector representation: $r = a + \lambda(b - a)$
- a, b: two vectors on line (from origin of coordinate system)
- r: pixel on line, λ line parameter



$$r = a + \lambda(b - a)$$
$$= \begin{pmatrix}2\\3\end{pmatrix} + \lambda\begin{pmatrix}2-2\\1-3\end{pmatrix}$$
$$= \begin{pmatrix}2\\3\end{pmatrix} + \lambda\begin{pmatrix}0\\-2\end{pmatrix}$$

RANSAC

RANSAC: Random Sample Consensus

Preprocessing: Create list of candidate line pixels (e.g., canny or use a threshold)

1. Repeat ***R*** iterations:
   a) Select 2 candidate line pixels at random
   b) Calculate parameters of line based on these pixels
   c) Count the number of pixels ***N*** that fit to the line
   d) If ($N > N_{max}$): keep line parameters and update N_{max}
2. Store line parameters, remove pixels on line, and continue with 1

The Hough Transform (1)

In **Hough space**, a line, described by

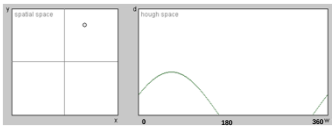
$$x \sin \alpha + y \cos \alpha = r$$

is defined by a **single point** where

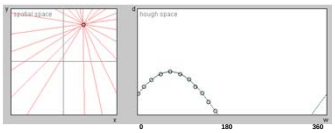
- the vertical axis defines the **distance** of the line from the origin (the length of the normal r on the line),
- the horizontal axis defines the **angle** of the normal r with the x axis.

The Hough Transform (2)

Transformation of a *point* in the spatial domain into Hough space



- Point in the spatial domain does not define a line by itself
  → any line passing through the point is a candidate line

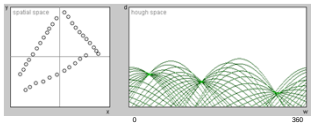


- All candidate lines correspond to the sine-shaped trajectory in the Hough space

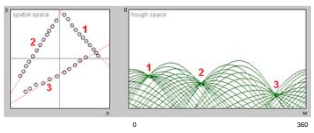
Classify a Line (1)

Example:

- Three roughly defined lines in the spatial domain
- Each point defines a sine-shaped trajectory in the Hough space.
- All trajectories meet in three distinct locations.

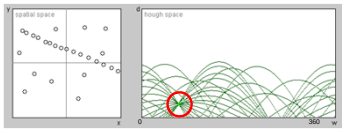


- Mark each intersection in the Hough space with a point
  → we get an approximation for the actual lines (shown in red)

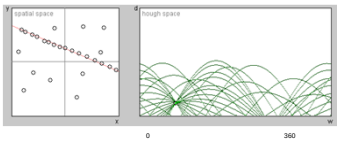


Classify a Line (2)

- Count line density ***D***: number of marked lines in the Hough space within a local neighborhood (the red circle on the right)



- IF (D > threshold): define its center of gravity as the representing line



Line Detection Algorithm

1. Calculate the gradient image (binary edge image)
2. Apply the Hough Transform to each edge pixel

```
maxAngle=359
maxDist=100
houghImage[0...maxAngle][0...maxDist]:= 0
foreach edgePixel p do
   for α:=0 to maxAngle do
      d:= pX * cos(α) + pY * sin(α)
      houghImage[α][d]++
   endfor
endfor
```

3. Identify local maxima in the `houghImage` array and map the local maxima back into the image (spatial space). Each maximum represents a line in the image.

Image Registration (1)

Definition: Image Registration

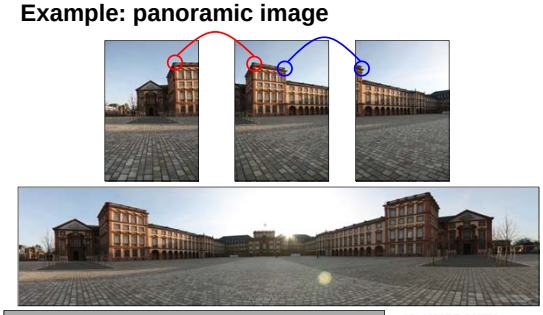
- Input: 2 images of the same scene but taken from different perspectives
- Goal: Identify transformation to present both images in the same coordinate system
- The alignment of images is called *Image Registration*

Image Registration techniques are used

- to detect camera motion
- to generate panoramic images
- to transform robots / the ball into a soccer field coordinate system

Image Registration (2)

Example: panoramic image



Affine Transform

Assumption

- Image is painted on a flat sheet:
  If we move the camera, which transformation of the captured image is possible?

2D motion

- 6 parameters define *affine motion*
  (translation, rotation, scaling, [mirroring, shear]):

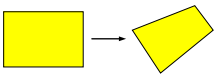
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Rotation, Scaling, [Shear], [Mirroring] — Translation

Perspective Transform (1)

3D motion

- Perspective transformation



- The parameters $a_{i,j}$ *and* t_x, t_y describe affine motion
- 2 additional parameters b_1 *and* b_2 describe perspective deformations

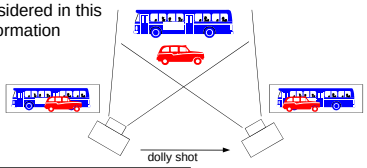
$$x' = \frac{a_{11}x + a_{12}y + t_x}{b_1 x + b_2 y + 1} \qquad y' = \frac{a_{21}x + a_{22}y + t_y}{b_1 x + b_2 y + 1}$$

Perspective Transform (2)

- Horizontal / vertical shift (horizontal pan or vertical tilt): t_x, t_y
- Zoom and rotation along the optical axis: $a_{i,j}$
- Perspective transformation: b_1, b_2
- Dolly shots (camera changes position) are not considered in this model (3D scene information would be required)

$$x' = \frac{a_{11}x + a_{12}y + t_x}{b_1 x + b_2 y + 1}$$

$$y' = \frac{a_{21}x + a_{22}y + t_y}{b_1 x + b_2 y + 1}$$



Mapping frames to a court model

- Detect 4 corresponding pixel positions
- The exact shift (x / y coordinate) of 4 blocks are sufficient to calculate the 8 parameters of the camera model