FILTER + CONVOLUTION

FILTERING

$$g(m,n) = \sum_{k,l} I(m+k, n+l) \cdot f(k,l)$$

CONVOLVING

$$g(m,n) = \sum_{k,l} I(m-k, n-l) \cdot f(k,l)$$

Properties:
- Commutative $f \otimes g = g \otimes f$
- Associative
  $f \otimes g \otimes h = f \otimes (g \otimes h)$
- Distributive
  $f \otimes (g+h) = f \otimes g + f \otimes h)$

SHIFT, DILATE, BLUR, SHARPEN BY FILTERING/CONVOLUTION

COMMON FILTERS
average filter: $ones(n \times n)$
Gaussian filter:

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Advantage of Gaussian filter: Rotationally symmetric; Weights nearby pixels more than distant ones
DoG filter:

$$\frac{\partial}{\partial x} h_\sigma(u,v), \frac{\partial}{\partial y} h_\sigma(u,v)$$

Application of Gaussian filter: edge detection, image pyramid etc.
Laplacian filter:

$$\nabla^2 h_\sigma(u,v) = \frac{\partial^2}{\partial x^2} h_\sigma(u,v) + \frac{\partial^2}{\partial y^2} h_\sigma(u,v)$$

Trick: de-compose 2D filters into 1Ds
(reason we do this: reduce computation cost)
Compute the gradient of image using this method:

$$I_x = I \otimes G_x \otimes \delta_x \otimes G_y$$
$$I_y = I \otimes G_y \otimes \delta_y \otimes G_x$$

OUTPUT SIZE: FULL, SAME, VALID
PADDING METHOD
Zero ~; symmetrical (around boundary reflection) ~; reflection ~; replicate ~

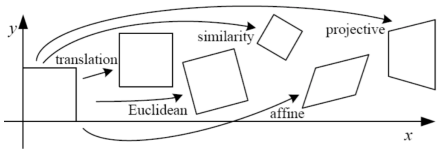
EDGE DETECTION

CRETERIA FOR OPTIMAL DETECTOR
- Good detection
- Good localization
- Single response

CANNY EDGE DETECTION

Canny Edge Algorithm
Input: image I
Output: edge map E
- Convolve I with derivative of Gaussian (Gaussian – filter out noise, then compute derivatives)
- Compute the magnitude and direction of the gradient
- Non-maximal suppression (Keep local maximal in local gradient direction; Use interpolation)
- Hysteresis (Edge linking; Two thresholds; Grow from 'strong' points)

TRANSFORMATION



LINEAR TRANSFORMATION
Combination of scaling (esp. uniform scaling), Rotation, mirror and

Shear $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$.

Properties of linear transformations:
- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

Linear transformation can also be considered as **a change of basis**.

AFFINE TRANSFORMATION
Combination of linear transformation and translation

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of affine transformations:
- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition
- Models change of basis

PROJECTIVE TRANSFORMATION
Combination of affine transformation and projective warps

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$H = \begin{bmatrix} A & vt \\ \mathrm{v} & v \end{bmatrix} = H_S H_A H_P = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ \mathrm{v} & v \end{bmatrix}$$

Properties of projective transformations:
- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis

Transformation	DoF (points)
Translation	1
Rotation	1
Euclidian (T+R)	2
Similarity	2
Affine	3
Projective	4

COMBINATION OF TRANSFORMATION
Pre-multiplying
POINT AND LINE
Point:
$(x_1, x_2, x_3) \Longrightarrow x = x_1/x_3, y = x_2/x_3$
$x_3 = 0$, point is at infinity
Line:
$ax + by + c = 0 \Longrightarrow (a,b,c)$
$(x,y,1)(a,b,c)^T \Longrightarrow ax + by + c = 0$
$a = b = 0$, line passes all point at infinity
Line passing two points x, x':
$l = x \times x' = (x_2x_3' - x_3x_2', x_3x_1' - x_1x_3', x_1x_2' - x_2x_1')$
then regularize.
Intersection point of two lines l, l':
$x = l \times l' = (l_2l_3' - l_3l_2', l_3l_1' - l_1l_3', l_1l_2' - l_2l_1')$
Under projective transformation, all parallel lines intersect at a point at infinity; One point at infinity ⇔ one parallel line direction.
Transformation of point and line:
$x' = Hx, l' = (H^{-1})^T l$
Infinite point/line becomes finite after projective transformation:

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = H \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix}, \begin{bmatrix} l_1' \\ l_2' \\ l_3' \end{bmatrix} = H^{-T} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

WARPING

FORWARD WARPING
$(x', y') = T * (x, y)$
Splatting: if pixel lands "between" two pixels, distribute color among neighboring pixels of (x', y')

INVERSE WARPING
$(x, y) = T^{-1} * (x', y')$
Bilinear interpolation: if pixel comes from "between" two pixels, do linear interpolation

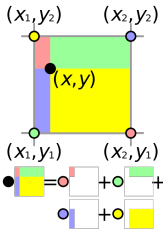


IMAGE MORPHING

TRIANGULATION

- Delaunay Triangulation (dual of Voronoi graph $O(nlogn)$) (On middle shape)
- Find intermediate image

$$tp_0 + (1-t)p_1$$

- Barycentric coordinates (BC)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

- For each point (in intermediate image), first decide which triangle it is in, then calculate its BC, use this BC to find its corresponding point in two source images.

TPS

- $x' = f(x,y) = a_1 + a_x x + a_y y +$

$\sum_i w_i U(\|(x_i, y_i) - (x,y)\|)$

(similarly for y')

where $U(r) = -r^2 \log(r^2)$ (more robust)

- Estimate TPS:

$$\begin{bmatrix} K & P \\ P^T & O \end{bmatrix} \begin{bmatrix} w \\ a \end{bmatrix} = \begin{bmatrix} v \\ o \end{bmatrix}$$

where:

$$K_{ij} = U(\|(x_i, y_i) - (x,y)\|)$$
$$P(i,:) = (1, x_i, y_i), v(i) = x_i' \text{ or } y_i'$$

- First estimate TPS parameters from current image to two source images using control points. Then for each point in intermediate image, apply TPS to find its correspondent points in two source images.

FEATURES

HARRIS CORNER DETECTOR

Advantages of local features:

Locality; Distinctiveness; Quantity; Efficiency; Extensibility

Application:

Image alignment; 3D reconstruction; Motion tracking; Object recognition; Indexing and database retrieval; Robot navigation

Properties of Harris Detection:

- Invariant to intensity shift
- Invariant to intensity scaling
- Non-invariant to image scaling

Harris Corner Detection Algorithm

Input: image

Output: measure of corner response map

Calculate change of intensity:

Choose the scale of different corner

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$w(x,y) > 0$ inside window, $w(x,y) = 0$ outside of window

$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+$

$v)$-$I(x,y)] \approx [u,v] M [u,v]^T$

Calculate the eigenvalues of M

$$\begin{cases} edge, & \lambda_2 \gg \lambda_1 or\ \lambda_1 \gg \lambda_2 \\ corner, both\ \lambda s\ are\ large, & \lambda_1 \sim \lambda_2 \\ flat\ region, & both\ \lambda s\ are\ small \end{cases}$$

Calculate measure of corner responses:

$$\boldsymbol{R} = detM / TraceM = \lambda_1 \lambda_2 / (\lambda_1 + \lambda_2)$$

ANMS

For each point p, calculate radius r. Every pixel inside the circle centered with p with radius r has smaller corner response than that of p.

FEATURE DESCRIPTOR

40x40 patch (dominate direction of corner response)

blur (local deformation invariant)

subsample: 5x5

FEATURE MATCHING

MATCHING

$SSD(1NN)/SSD(2NN) < 0.6$

K-D tree

RANSAC

RANSAC Algorithm

Input: corresponding feature point pairs

Output: *H,* set of inliers

For each iteration:

Randomly select s feature pairs

Compute homography *H*

Compute inliers where

$\boldsymbol{SSD(p', p'_{ets}) < threshold}$

Keep largest set of inliers

Re-compute *H* on all inliers

How to fit the model:

Linear regression

Way to decide number of iterations *N*:

s – number of feature pairs in a sample

e – probability that a point is an outlier

N – number of iterations

p – desired probability of getting good *H*

$$1 - (1 - (1-e)^s)^N = p$$
$$N = \log(1-p) / \log(1 - (1-e)^s)$$

4 points algorithm:

since $\mathrm{x}_i' = H\mathrm{x}_i$, $x_i' \times H\mathrm{x}_i = 0$

Let $\mathrm{x}_i' = (x_i', y_i', w_i')$

Then

$$\mathrm{x}_i' \times \mathrm{H}\mathrm{x}_i = \begin{bmatrix} y_i' h^{3T}\mathrm{x}_i - w_i' h^{2T}\mathrm{x}_i \\ w_i' h^{1T}\mathrm{x}_i - x_i' h^{3T}\mathrm{x}_i \\ x_i' h^{2T}\mathrm{x}_i - y_i' h^{1T}\mathrm{x}_i \end{bmatrix}$$

$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' & x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y_1' & y_1y_1' & y_1' \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' & x_2' \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y_2' & y_2y_2' & y_2' \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3x_3' & y_3x_3' & x_3' \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3y_3' & y_3y_3' & y_3' \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4x_4' & y_4x_4' & x_4' \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4y_4' & y_4y_4' & y_4' \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ h8 \end{bmatrix} = 0$$

$$A = USV, h = V(:,9)$$

GENERATE PANORAMA

Image warping:

$$X_i = H_{i+1} X_{i+1}$$

To represent all images one world frame:

Let $H_1 = I_{3\times3}$,

$X_1 = H_1 X_1$, $X_1 = H_2 X_2$, $X_2 = H_3 X_3$, $\cdots$, $X_{n-1} = H_n X_n$

$$\therefore X_1 = (\prod_{i=1}^{n} H_i) X_n = T_n X_n$$

Choose another world frame c:

$$X_1 = T_c X_c = T_i X_i$$
$$X_c = T_c^{-1} T_i X_i = T_i^c X_i$$

BLENDING

ALPHA CHANNEL

Feathering; Averaging; Center weighting:

$alpha = dist1/(dist1 + dist2)$

The influence of window size: avoid seam & ghosting

LAPLACIAN PYRAMID

General Approach:

1. Build Laplacian pyramids LA and LB from images A and B
2. Build a Gaussian pyramid GR from selected region R
3. Form a combined pyramid LS from LA and LB using nodes of GR as weights:

$$LS(i,j) = GR(I,j,)LA(I,j) + (1 - GR(I,j))LB(I,j)$$

4. Collapse the LS pyramid to get the final blended image

Esp. 2-band blending (low freq. hight freq.)

PYRAMID

Represent a $N \times N$ image as a pyramid of size $1\times1, 2\times2, 4\times4, \ldots, 2^k \times 2^k \ldots (2^k \le N)$

Blur the image (filter with Gaussian), then subsample

GAUSSIAN PYRAMID

Filter size double <> image size /2

Each pixel containing a local average that corresponds to a pixel neighborhood on a lower level of the pyramid.

Use: texture synthesis

LAPLACIAN PYRAMID

Laplacian pyramid saves the difference image of the blurred versions between each levels. Only the smallest level is not a difference image to enable reconstruction of the high resolution image.

Use: image compression

IMAGE ENCODEING & DECODEING

Coding book of DCT (discrete Cosine)

APPENDIX

CROSS PRODUCT

$$\mathbf{c} = \mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} =$$

$$= (a_2 b_3 - a_3 b_2)\mathbf{i} + (a_3 b_1 - a_1 b_3)\mathbf{j} + (a_1 b_2 - a_2 b_1)\mathbf{l}$$