Normalized 8-point Algorithm (DLT at the end)

- Given 8 or more corresponding points, we can estimate the fundamental matrix using the 8-point.
- The DLT algorithm involves solving a linear system of equations to estimate the fundamental matrix.

$$\begin{bmatrix} x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx'_m & x_my'_m & x_m & y_mx'_m & y_my'_m & y_m & x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = 0$$

$$\mathbf{AX} = \mathbf{0}$$

Algorithm Steps

- Normalize the points (since we are working in image coordinates)
  - Let the matrix used for normalizing the points in both cameras be $\mathbf{T}$ and $\mathbf{T}'$
- Construct the matrix $\mathbf{A}$. Find SVD of $\mathbf{A}$, and using that, find approximate value $\hat{\mathbf{F}}$
- To find $\mathbf{F}$, enforce the rank-2 constraint on $\hat{\mathbf{F}}$ using SVD

$$\min_F ||F - \hat{F}||_F, \text{subject to } \det F = 0$$

$$F = U \begin{bmatrix} \Sigma_1 & 0 & 0 \\ 0 & \Sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^\top$$

- Denormalize the fundamental matrix to get the final fundamental matrix $F_{\text{final}} = (\mathbf{T}')^\top F \mathbf{T}$
- For DLT, every point generates 2 pairs $\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -x'_1y'_1 & -x'_1 \end{bmatrix}$, $\begin{bmatrix} 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1x'_1 & -y'_1y'_1 & -y'_1 \end{bmatrix}$ in $\mathbf{A}$, so you only need 4 corresp. points. Baki same algo only (except for rank 2 constraint)

Image Processing (just do convolution ez)

- Filtering: Form a new image whose pixel values are a combination of the original pixel values, to get useful info out of them.
- **Linear Filter**: Replace each pixel by a linear combination (a weighted sum) of its neighbors, the linear combination is the kernel.
- Cross Correlation:
  - Let F be the image, H be the kernel (of size $2k+1 \times 2k+1$), and G be the output image
  - $G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$ $(G = F \otimes H)$
  - Can think of it as a "dot product" between local neighborhood and kernel for each pixel
- Convolution:
  - Kernel is flipped (horizontally and vertically) before applying cross-correlation
  - $G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$ $(G = F * H)$
  - It is commutative, and associative.
- Averaging/Blur Filter: $1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, Sharpening Filter: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - 1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
- **Gaussian Filter**: Used for blurring, noise reduction, etc. Removes high frequency components from the image (low-pass filter)
  - Convolution with self is another Gaussian
  - Convolving twice with Gaussian kernel of width σ = convolving once with kernel of width $\sigma\sqrt{2}$
- Original - Blurred = High-pass filter, Original + α High-pass = Sharpened

Edge Detection

Convert a 2D image into a set of curves - Extracts salient features of the scene, More compact than pixels. - Edge is a place of rapid change in the image intensity function

How to take derivative for digital image? take discrete derivative $\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$. Gradient of an image is ∇f . Edge strength is $||\nabla f||$ and gradient direction is θ

$$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}] \text{ , } ||\nabla f|| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} \text{ , } \theta = \arctan(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x})$$

Noise in the image might cause too many rapid changes, so we smooth the image before taking the derivative. $\frac{d}{dx}(f * h) = f * \frac{d}{dx}h$. For 1D Gaussian,

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma}}, G'_\sigma(x) = -\frac{1}{\sigma}(\frac{x}{\sigma})G_\sigma(x)$$

Harris corner detection for feature detection

- Key Idea: Small shifts in the window should cause large changes in the intensity
- Compute image gradients for each pixel over a small region W. Let the image gradients be I_x, I_y.
- Subtract mean from each image gradient to remove the constant intensity
- Compute the covariance

$$H = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

This comes from second order taylor expansion of $E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$, where x, y are the points in the window, and $w(x,y)$ is the window function.

- Compute eigenvectors and eigenvalues $Hx = \lambda x$
  - $x_{\max}$ is the direction of the largest change in intensity
  - $x_{\min}$ is the direction of the smallest change in intensity
  - $\lambda_{\max}$, $\lambda_{\min}$ are the amount of increase in direction of $x_{\max}$ and $x_{\min}$ respectively
- Depending on the value of $\lambda_{\max}$ and $\lambda_{\min}$, we can classify the point as a corner, edge, or flat region
  - $\lambda_{\max}$ is large, $\lambda_{\min}$ is small $\rightarrow$ Corners. $\lambda_{\max}$ and $\lambda_{\min}$ are large $\rightarrow$ Edge. $\lambda_{\max}$ and $\lambda_{\min}$ are small $\rightarrow$ Flat region
- Choose points where $\lambda_{\min}$ is local maxima. Normally, we just approximate $\lambda_{\min}$ with Harris operator, which is usually much easier to compute. one common choice is $\frac{\det(H)}{\text{trace}(H)} >$ threshold

Feature Description

- Come up with a descriptor for each point, find similar descriptors in another image, and match them
- Invariace: Descriptor shouldn't change even if image is transformed
- Discriminability: Descriptor should be highly unique for each point
- Harris Detector is not invariant to scale, orientation, and illumination. SIFT is invariant to all of these.
- Automatic scale selection: Find scale that gives local maxima of some function f in both position and scale.
- From edges to blobs: Convolve image with Gaussian of different scales, and find local maxima in scale space

SIFT (Scale Invariant Feature Transform) for feature description

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations
- Can handle changes in viewpoint (up to about 60 degree out of plane rotation)
- Can handle significant changes in illumination (sometimes even day vs. night)
- Pretty fast—hard to make real-time, but can run in <1s for moderate image sizes

RANSAC

- Randomly sample a subset of points, fit a model to that subset (Take 4 points, compute homography using DLT)
- Find the number of inliers for that model [$d(p'i, Hpi) < \epsilon$]
- Repeat for a large number of iterations, and choose the model with the most inliers (Re-compute least-squares H estimate using all of the inliers)
- Robustness : If there are G proportion of inliers, and our model need P pairs, Probability of picking P inliners = G^P, Probability of not picking set of inliners after N iterations = $(1 - G^P)^N$
- If there are multiple structures, RANSAC won't out of the box due to unknown number of structures, inlier set for each structure, and noise affecting the structures.