1 Light & color

Light

refraction(transparent obj)
absorbing & reflection (opaque obj)

color perception

dependencies:
1. color of lights: physics of light & reflectance of the surface.
2. percieved colors: visual system receptor, lights..

2 Image filtering

Digital Image formation

1. Formation of image: illumination + scene element + imaging system + image plane
2. Digital camera: sample 2d space on regular grid & quantize each sample (round to nearest integer)

Image noise, filtering, conv

1. **common types of noise**: salt/pepper noise, impulse noise (random occurrences of white pixels), Guassian noise
2. **Cross correlation** filtering $G = H \otimes F$:
$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$
Convolution (cross correlation will flip the img):
$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$
box filter, Gaussian filter:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & ? & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

3. Runtime complexity: $O(C_{in} \cdot C_{out} \cdot N^2 \cdot M^2)$
4. Separability: $1Dgs * 1Dgs = 2Dgs$
$Image * 2Dgs = Image * 1Dgs * 1Dgs$
5. Smoothing: remove high-freq components (low pass filter)
6. Prop of convs:
a. Commutative: f*g = g*f
b. Associative: (f*g)*h = f*(g*h)
c. Distributive over +: l*(f1+f2) = l*f1+l*f2
d. Shift invariant shift(l*f) = shift(l)*f
7. **Sharpening filter** (image - smoothed = details): positive in the middle & negative around
7. **Nonlinear filter**: median filter
9. **hybrid image**: Laplacian filter(identify rapid change) unit impulse - Gaussian

$$\mathbf{K} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

3 Edge detection

Derivatives and Gradients

Image Gradient: Measures the rate and direction of intensity change.
Gradient Magnitude/Direction (Edge Strength/Orientation)

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}, \quad \theta = \tan^{-1}\left(\frac{\partial I/\partial y}{\partial I/\partial x}\right.$$

Edge Detection Filters

Sobel Filter: compute gradients.

$$\text{Horizontal} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Prewitt Filter: Change 2 in Sobel to 1 for noise robustness.

Canny Edge Detection Steps

1. **Smoothing**: w a Gaussian filter.
2. **Gradient Calculation**: Compute gradients w Sobel filters.
3. **Non-Maximum Suppression**: Retain local maxima along the grad direction to thin the edges.
4. **Thresholding**: A **high thres** to start edges and a **low thres** to continue edges.

4 Local Feature Detection

Desired Properties of Local Features

1. **Repeatability**: Detectable across images despite transformations.
2. **Saliency**: Be distinct.
3. **Efficiency**: Lighter than total pixels.
4. **Locality**: Cover small, clutter-resistant regions.

Corners as Interest Points

1. **Flat Region**: No intensity change in any direction.
2. **Edge**: No change along the edge direction.
3. **Corner**: Large change in all directions.

Harris Corner Detector

Invariant to rotation, not scale
1. **Compute** M for each image window:

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

2. **Corner Response Function**:

$$R = \det(M) - \alpha \cdot (\text{trace}(M))^2$$

3. **Thresholding**: Retain points with large R values.
4. **Non-Maximum Suppression**: Keep local maxima to avoid redundant corners.

Blob Detection and Scale Selection

1. Scale-invariant but not rotation.
2. **Laplacian of Gaussian (LoG)**:

$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

3. **Characteristic Scale**: Scale at which the LoG response is maximized.

Feature Matching

1. Extract keypoint features.
2. Compute potential matches between images.
3. Hypothesize transformation T to align related matches.
4. Verify transformation by checking for consistency across matches.

5 Local Feature Description

SIFT Descriptor (Lowe, 2004)

1. Compute gradients within sub-patches –> histograms of gradient orientations.
2. Rotate the patch based on dominant gradient –> rotation invariance.
3. Map pixels to a 128-dimensional vec.
4. Invariance to scale and rotation, partial invariance to illumination changes, capable of handling occlusion.

Blob Detection and Difference of Gaussians (DoG)

1. Key: identifying maxima or minima in both position and scale.
2. The Laplacian of Gaussian (LoG) can be approximated with DoG for better efficiency:

$$\text{DoG}(x,y,\sigma) = G(x,y,k\sigma) - G(x,y,\sigma)$$

3. The characteristic scale corresponds to the peak LoG response, capturing feature blobs across multiple scales.

Matching Techniques

1. Compute candidate matches w Sum of Squared Distances (SSD) etc.
2. Use Nearest Neighbor Search with a thres of nearest to second-nearest descriptor.

$$\text{Ratio} = \frac{\text{Distance to best match}}{\text{Distance to second-best match}}$$

If the ratio is low, the match is reliable; if high, it may indicate ambiguity.

6 Fitting and Hough Transform

Hough Transform for Line Detection

1. Mapping to Hough Space:
- **Image space** (x,y): Holds edge points.
- **Hough space** (m,b): Lines $y = mx + b$.
- A point in the image space –> a line in Hough space.
2. Advantages: robust to noise, tolerance to disconnected segments

Hough Transform Algorithm (Line Detection)

1. Initialize accumulator arr $H[m,b] = 0$.
2. For each edge point (x,y) in the image:
   For each possible slope m:
      Calculate $b = y - mx$.
      Increment $H[m,b]$ by 1.
3. Find the (m,b) with the highest votes in H.

Polar Representation of Lines

1. To avoid infinite slopes for vertical lines, use the polar equation:

$$x\cos\theta + y\sin\theta = d$$

2. Each edge point votes for a sinusoid in (d,θ) space, and intersections correspond to lines in the original image space.

Extensions of Hough Transform

- Use gradient direction to reduce param search.
- Assign higher weights to stronger edges during voting.
- Adapt the transform for circles etc.

Pros and Cons of Hough Transform

1. **Pros:**
- Can handle occlusion, noise, and gaps in features.
- Detects multiple instances of a model in a single pass.
2. **Cons:**
- Search time increases exponentially with more parameters.
- Non-target shapes may produce spurious peaks.
- Requires careful choice of grid size for parameter space.

7 Fitting a 2D Transformation

Parametric Warping

1. Transformation T can take different forms:
- Translation, Rotation, Scaling, Affine, and Perspective.
2. These transformations can be represented as matrix operations, such as:

$$\mathbf{p}' = \mathbf{T} \cdot \mathbf{p}$$

Basic Transformations as Matrices

1. **Scaling**, **Rotation**:

$$\mathbf{S} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

2. **Shear**, **Translation**:

$$\mathbf{S} = \begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Affine Transformations

1. Combine linear transformations (scaling, rotation, shear) and translation:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2. Parallel lines remain parallel under affine transformations.

Using RANSAC for Robust Fitting

1. RANSAC (Random Sample Consensus):
- Randomly select a set of pts, estimate the transformation.
- Compute the transformation and identify inliers.
- If enough inliers, recompute the transformation with all inliers.
2. Keep the transformation with the most inliers across multiple trials.

Summary of RANSAC

1. **Pros**:
- Robust to noise & outliers(extremely x).
2. **Cons**:
- Requires careful tuning of hyperparams.
- Perf drops with low inlier ratios.
- Can struggle with poor initialization based on minimum samples.

8 Homography and Image Mosaics

1. **Goal:** Use homography to align and stitch images into a seamless mosaic.
2. **Homography:** A projective transformation that maps points from one image plane to another.
- Preserve straight lines, not necessarily parallel lines/length/angle.
- Represented by a 3x3 matrix H:

$$\mathbf{p}' = H \cdot \mathbf{p}$$

Generating Image Mosaics

1. Capture a sequence of images from the same camera position.
2. Compute the transformation between consecutive images w feature-based alignment.
3. Use homography to transform and align images.
4. Blend aligned images to create the final mosaic.

Image Warping and Reprojection

1. **Forward Warping:**

$$(x', y') = T(x, y)$$

- If a pixel lands between two pixels in the target image, distribute its color among neighbors (splatting).
2. **Inverse Warping:**

$$(x, y) = T^{-1}(x', y')$$

- Interpolate color values from neighbors (e.g., nearest neighbor, bilinear interpolation).

RANSAC for Homography Estimation

1. Randomly select 4 pairs of corresponding pts.
2. Compute the homography matrix H.
3. Identify inliers—pairs that satisfy:

$$\text{SSD}(\mathbf{p}', H \cdot \mathbf{p}) < \epsilon$$

4. Keep the set of inliers w the largest size.
5. Recompute H using all inliers w least squares.

9 Image Formation

Physical Parameters

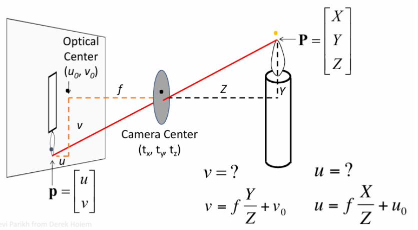
1. **Photometric:** - Type, direction, and intensity of light reaching the sensor.
- Surface reflectance properties.
2. **Geometric:** - Type of projection (e.g., perspective, orthographic).
- Camera pose and position.
3. **Optical:** - Lens type, focal length, aperture, and shutter speed.

Pinhole Camera Model

1. A camera model where light passes through a pinhole onto a film/sensor.
- Reduces blurring by blocking most light rays and allows sharp image formation.

Projective Geometry

1. Projection maps 3D world coordinates to 2D image coordinates.



2. **Vanishing Points:** 3D Parallel lines -> converge at a point in the image. (Vertical vanishing point: infinity)
Vanishing Line: 3D Parallel lines -> converge at a line in the image.

Homogeneous Coordinates and Camera Matrix

1. Homogeneous coordinates: add another axis, translation –> mat mul.
2. **Intrinsic Matrix** (K): Encodes internal camera properties. Deg of freedom: 5

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- f_x, f_y: Focal length.
- s: Skew factor (usually 0) to account for non-rectangular pixels.
- c_x, c_y: Optical center coordinates (principal point).

3. **Extrinsic Matrix** ($[R|t]$): External camera params. Deg of freedom: 6

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma)$$

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

- R, t: from world's coordinate to cam's

Orthographic and Weak Perspective Proj

1. **Orthographic Projection:** The distance to the image plane is infinite.
2. **Weak Perspective:** An approximation where object dimensions are small relative to the camera distance.

10 Stereo and Camera Calibration

Stereo Vision

1. **Stereo Vision Setup:** Two or more cams capture the scene from slightly different viewpoints.
2. **Disparity:** The difference in the position of corresponding pts between the two images. Depth is inversely proportional to disparity:

$$\text{Depth} = \frac{f \cdot B}{\text{Disparity}}$$

where f is the focal length, and B is the baseline (distance between the cameras).

Projection Matrix and Calibration Process

Projection Model:

$$\mathbf{p} = \lambda M \mathbf{X}$$

where $M = K[R|t]$, 3×4 projection matrix
1. **Camera calibration:** Estimates M given p, X.
- **Calibration Targets:** Objects with known dimensions (like checkerboards)
- **Linear Calibration Method:** Uses correspondences between 3D points and their projections to estimate M.
- **Linearization via Cross-Product:** The projection model is linearized as:

$$\mathbf{p} \times (M\mathbf{X}) = 0.$$

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \times \begin{bmatrix} m_1^T \mathbf{X}_i \\ m_2^T \mathbf{X}_i \\ m_3^T \mathbf{X}_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

$$\begin{bmatrix} 0^T & -\mathbf{X}_i^T & v_i \mathbf{X}_i^T \\ \mathbf{X}_i^T & 0^T & -u_i \mathbf{X}_i^T \\ -v_i \mathbf{X}_i^T & u_i \mathbf{X}_i^T & 0^T \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Note: 2 linearly independent eqs per correspondence, M (3x4) 11 deg of freedom

- **Stacking Equations:** For n correspondences:

$$A\mathbf{m} = 0,$$

where A is a $2n \times 12$ matrix and $\mathbf{m}$ contains 12 unknown entries of M.
- **Solving the System:** Use SVD to find the eigenvector of $A^T A$ corresponding to the smallest eigenvalue, providing an initial estimate of M.
- Note: Can use a **non-linear optimizer** (e.g., Levenberg-Marquardt) to handle noise and improve accuracy.

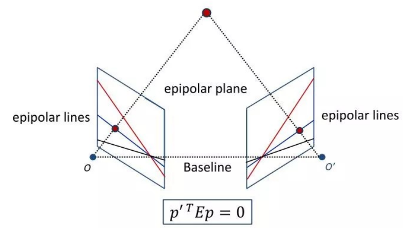
2. **Triangulation:** Given M,p –> X
- **Method 1: Geometric Approach** Find the shortest segment between viewing rays and select the midpoint of this segment.
- **Method 2: Non-linear Optimization**

$$X = argmin(\sum_i d(\mathbf{p}_i, M_i X)^2),$$

11 Epipolar Geometry

Epipolar Geometry:

Describes the geometric relationship between two cameras and the corresponding points in their images.
1. **Epipoles:** The intersections of the baseline (the line connecting the two camera centers) with the image planes. Each epipole is the projection of the other camera center.
- All epipolar lines for a given image pass through the epipole.
- Image plane parallel: Epipoles infinitely far away, epipolar lines parallel



Fundamental and Essential Matrices:

1. **Fundamental Matrix** F relates corresponding points in two *uncalibrated* images, singular(r=2), deg free=7:

$$\mathbf{p}'^T F \mathbf{p} = 0.$$

It maps a point p in the first image to its epipolar line in the second image.
2. **Essential Matrix** E is a special case for *calibrated* cameras:

$$E = [t_x]R,$$

R : rotation matrix, $[t_x]$: skew-symmetric matrix representing translation.

3. **Epipolar Constraint (Calibrated Case):** With known intrinsic matrices K_1 and K_2, the projection matrices are:

$$M_1 = K_1[I \mid 0], \quad M_2 = K_2[R \mid t].$$

$$\hat{\mathbf{p}}'^T E \hat{\mathbf{p}} = 0.$$

Estimating the Fundamental Matrix

1. **Linear Estimation of** F**:** From the epipolar constraint and Stacking:

$$u'uf_{11} + u'vf_{12} + u'f_{13} + v'uf_{21}$$

$$+ v'vf_{22} + v'f_{23} + uf_{31} + vf_{32} + f_{33} = 0.$$

$$A\mathbf{f} = 0,$$

where A is an 8×9 matrix constructed from the point correspondences, and $\mathbf{f}$ is the 9-vector representing the entries of F.

2. **Eight-Point Algorithm:** - Solve the system $A\mathbf{f} = 0$ using Singular Value Decomposition (SVD) to estimate F. - F rank 2, set the smallest singular value of F to zero.

3. **Normalization:** - For better numerical stability:

$$\mathbf{p} \to \frac{\mathbf{p}}{\|\mathbf{p}\|}, \quad \mathbf{p}' \to \frac{\mathbf{p}'}{\|\mathbf{p}'\|}.$$

- After the estimation, the fundamental matrix should be denormalized.

4. **RANSAC for Robust Estimation:** - RANSAC randomly samples minimal sets of correspondences (8) to estimate F and selects the solution that maximizes the number of inliers (pts that satisfy the epipolar constraint within a threshold).

12 Depth from Stereo & Structure from Motion

Depth from Stereo:

1. **Goal:** Recover depth by finding corresponding points in two images (stereo matching). Depth is inversely related to disparity.
2. **Disparity:** - Disparity $d = x - x'$, where x and x' are corresponding image coordinates in two views.

$$d = \frac{fB}{d},$$

d: depth, f : focal length, B : baseline distance between cameras.

3. **Stereo Matching Algorithm:** - For each pixel in imgA, find corresponding epipolar line in imgB. - Search along epipolar line and select best match based on a similarity measure (e.g., SSD, normalized correlation). - Triangulate to obtain depth.
4. **Rectification:** Reproject image planes onto a common plane to transform epipolar lines into horizontal scanlines, simplifying correspondence search.
5. **Basic Challenges:** - Textureless regions: Hard to find unique correspondences. - Repeated patterns: Ambiguity in matching points. - Specular surfaces: Appearance changes with viewpoint.

Structure from Motion (SfM):

1. **Goal:** Recover 3D structure and camera motion from multiple views.
2. **Projection Model:** - For a 3D point X_j and camera M_i, the image point p_{ij} is given by:

$$p_{ij} \equiv M_i X_j, \quad p_{ij} = M_i X_j.$$

- $M_i = K_i[R_i|t_i]$, K_i : intrinsic matrix, R_i, t_i: extrinsic parameters.
3. **Ambiguities in SfM:** - **Scale ambiguity:** Cannot determine absolute scale from motion alone. - **Projective ambiguity:** Without constraints, reconstruction is only determined up to a projective transformation. - **Affine ambiguity:** If parallel lines are known, ambiguity reduces to affine. Additional constraints can reduce ambiguity to similarity.
4. **Affine Structure from Motion:** - Use affine cameras for a simplified SfM approach. For m cameras and n points:

$$p_{ij} \equiv A_i X_j + b_i,$$

A_i : affine projection matrix. b_i : translation.
- Factorize the measurement matrix to recover motion and structure:

$$D = \begin{bmatrix} A_1 \\ \vdots \\ A_m \end{bmatrix} [X_1 \quad \dots \quad X_n].$$

5. **Triangulation in SfM:** - Estimate the 3D points X_j by minimizing the reprojection error across all views:

$$\sum_i \|p_{ij} - M_i X_j\|^2.$$

- Use optimization techniques such as bundle adjustment to refine the structure and motion estimates.