Robot Vision

Lecturer: Prof. Davide Scaramuzza, Note: Yunlong Song (宋运龙)

If I have seen further it is by standing on the shoulders of Giants. — Isaac Newton

Introduction

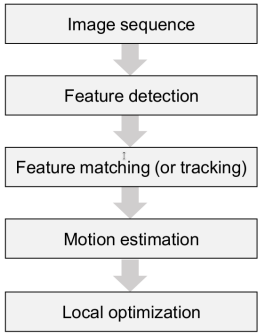

Figure 1: Overview of Visual Odometry.

Image Formation

The Pinhole Approximation $e \approx f$:

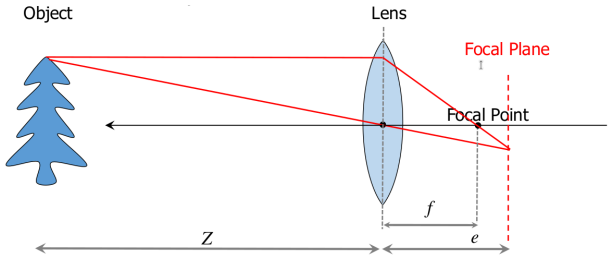

Figure 2: Pinhole Camera Model.

$$-\frac{x}{X} = \frac{f}{Z}$$
$$x = -\frac{fX}{Z}$$

Perspective Projection: Far away objects appear smaller, with size inversely proportional to distance.

Field of View (FoV): The angle of the field of view is determined by the focal length and the sensor size.

$$\text{FoV} = 2\arctan\left(\frac{W}{2f}\right)$$

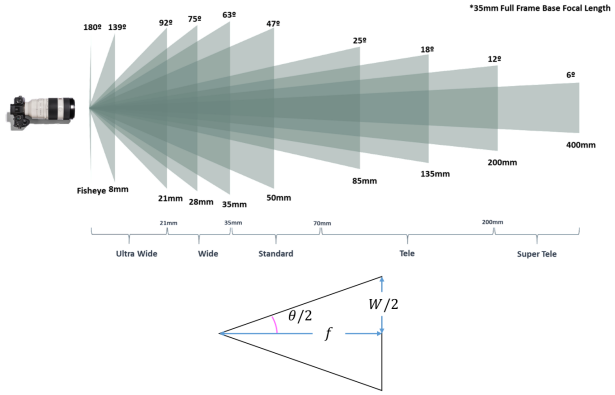

Figure 3: Field of View. W: sensor width, f: focal length.

Rolling Shutter: The image is captured row by row, which can cause distortion in fast moving scenes.
Global Shutter: The entire image is captured at once, which is more suitable for fast moving scenes.
Perspective Projection Equation: A world's 3D points $P_w = [X_w, Y_w, Z_w]^T$ are projected to the image plane as $P = [u, v]^T$. A camera is a mapping between the 3D world and a 2D image plane

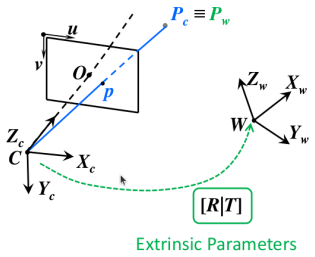

Figure 4: Perspective Projection.

$$\underbrace{x}_{\text{2D image}} = \underbrace{K}_{\text{Intrinsic}} \underbrace{[R|t]}_{\text{Extrinsic}} \underbrace{X}_{\text{3D world}}$$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

where α is the focal length, (u_0, v_0) is the principal point, λ is the scale factor due to the homogeneous coordinates.

Camera Calibration

Camera calibration is the process of estimating the intrinsic K and extrinsic parameters $[R|t]$ of a camera. Sometimes, the distortion parameters are also estimated.

Tsai's Camera Calibration: From 3D objects:

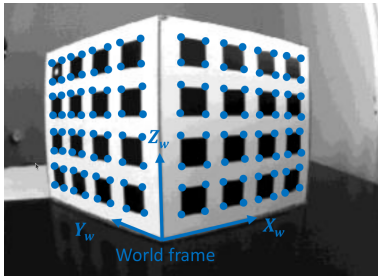

Figure 5: Tsai's Camera Calibration.

Direct Linear Transform (DLT): The idea of DLT is to rewrite the perspective projection equation as a **homogeneous linear equation** and solve it using standard linear algebra.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$
$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Stacking the equations for n 3D-2D correspondences, we get a linear system

$$\underbrace{Q_{2n\times 12}}_{\text{Know}} \cdot \underbrace{M_{12\times 1}}_{\text{unknown}} = 0$$

where Q is the matrix of 2D image points and 3D world points, and M is the camera matrix P.

Minimum Solution: 6 points, 3D points are not coplanar. $Q_{2n\times 12}$ should have rank 11 to have a unique solution.

Over-determined Solution: More than 6 points, use SVD to solve for M.

Non-linear Calibration Refinement: Use Levenberg-Marquardt to refine the calibration parameters by minimizing the reprojection error.

Zhang's Camera Calibration: From 2D images:

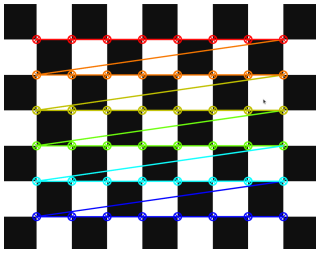

Figure 6: Zhang's Camera Calibration.

Tsai's calibration requires that the world's 3D points are non-coplanar, which is not very practical. Zhang's calibration uses a planar calibration pattern, e.g., $\mathbf{Z_w = 0}$. We start by writing the perspective projection equation for the planar pattern

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 1 & h_1 2 & h_1 3 \\ h_2 1 & h_2 2 & h_2 3 \\ h_3 1 & h_3 2 & h_3 3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

Stacking the equations for n 2D-2D correspondences, we get a linear system

$$\underbrace{Q_{2n\times 9}}_{\text{Know}} \cdot \underbrace{H_{9\times 1}}_{\text{unknown}} = 0$$

Minimum Solution: 4 points, 2D points are not collinear. $Q_{2n\times 9}$ should have rank 8 to have a unique solution.

After solving for H, we can decompose it into K and $[R|t]$.

Camera Localization (PnP)

Camera localization or Perspective-n-Point (PnP) is the problem of estimating the pose of a calibrated camera given the 3D world points and their corresponding 2D image points.

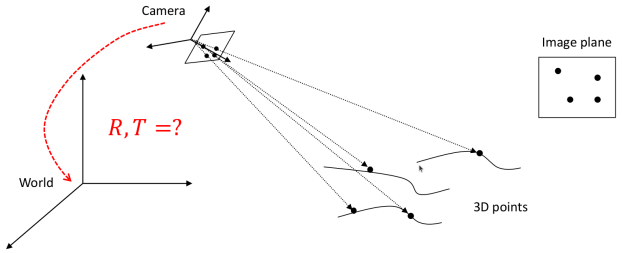

Figure 7: Camera Localization.

How many points are needed to localize a camera?

- 1 point: infinite solutions
- 2 points: infinite solutions (but bounded)
- 3 points (non collinear): up 4 solutions
- 4 points: unique solution

P3P Algorithm: When $n = 3$, the PnP problem is in its minimal form of P3P and can be solved with three point correspondences. However, with just three point correspondences, P3P yields up to four real, geometrically feasible solutions. For low noise levels a fourth correspondence can be used to remove ambiguity.

EPnP Algorithm: This method is based on the notion that each of the n points (which are called reference points) can be expressed as a weighted sum of four virtual control points. Thus, the coordinates of these control points become the unknowns of the problem. It is from these control points that the final pose of the camera is solved for.

EPnP Algorithm

Given n 3D-2D correspondences $\mathbf{P}_i$ and $\mathbf{p}_i$, the EPnP algorithm proceeds as follows:

1. Centroid Computation:

$$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{P}_i, \quad \mathbf{P}'_i = \mathbf{P}_i - \mathbf{C}$$

2. Virtual Control Points:

$$\mathbf{C}_1 = (0,0,0), \mathbf{C}_2 = (1,0,0), \mathbf{C}_3 = (0,1,0), \mathbf{C}_4 = (0,0,1)$$

3. Weights Representation:

$$\mathbf{P}'_i = \sum_{j=1}^{4} w_{ij}\mathbf{C}_j$$

4. Projection Equations:

$$s_i \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \mathbf{K}\left(\mathbf{R}\sum_{j=1}^{4} w_{ij}\mathbf{C}_j + \mathbf{t}\right)$$

5. Solve for Weights: Form and solve a linear system for w_{ij}.
6. Determine Pose: Solve for $\mathbf{R}$ and $\mathbf{t}$ using w_{ij}.
7. Optimization:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{n} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2$$

Image Filtering

Convolution: A convolution is a mathematical operation on two functions f and g that produces a third function expressing how the shape of one is modified by the other.

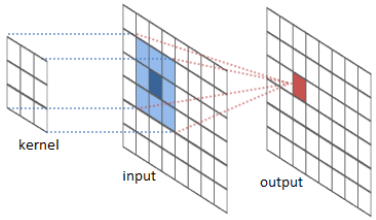

Figure 8: Convolution.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

Discrete Convolution:

$$(f * g)[n] = \sum_{m=-k}^{k} f[m]g[n-m]$$

2D Convolution:

$$(f * g)[i,j] = \sum_{m=-k}^{k} \sum_{n=-k}^{k} f[m,n]g[i-m, j-n]$$

Linear Filter:

- **Mean Filter**: Replace each pixel with the average of its neighbors.

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} = \frac{1}{3}\begin{bmatrix}1\\ 1\\ 1\end{bmatrix}\frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix}$$

- **Gaussian Filter**: Replace each pixel with the weighted average of its neighbors.

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} = \frac{1}{4}\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}\frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

- **Sobel Filter**: Detect edges in an image.

$$\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \quad \text{or} \quad \begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}$$

- **Laplacian Filter**: Detect edges in an image.

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

2D convolution can be sped up if the filter is separable. For example, a 2D Gaussian filter can be decomposed into two 1D Gaussian filters.

$$I \star G = (I \star G_x) \star G_y$$

Non-linear Filter:

- **Median Filter**: Replace each pixel with the median of its neighbors.
- **Bilateral Filter**: Replace each pixel with the weighted average of its neighbors, where the weights depend on both the spatial distance and the intensity difference.

Edge Detection:

Canny Edge Detector

The Canny edge detector is a multi-step algorithm that involves:
1. Take a grayscale imagee.
2. Convolve the image I with x and y derivatives of a Gaussian filter.

$$\frac{\partial I}{\partial x} = I \star \frac{\partial G_\sigma}{\partial x} \quad \text{and} \quad \frac{\partial I}{\partial y} = I \star \frac{\partial G_\sigma}{\partial y}$$

and compute the edge strength and orientation:

$$\|\nabla I\| = \sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2}$$

$$\theta = \arctan 2\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$$

3. Thresholding: set to zero all pixels of $\|\nabla I\|$ whose value is below a given threshold.
4. Thinning: remove pixels that are not local maxima along the gradient direction.

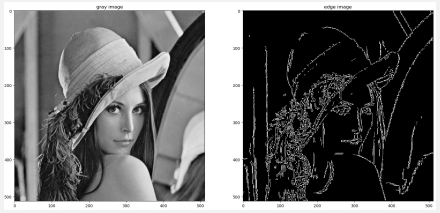

Figure 9: Canny Edge Detector.

Feature Detection

Cross-Correlation: A measure of similarity between two signals.

$$(f \star g)[i,j] = \sum_{m=-k}^{k} \sum_{n=-k}^{k} f[m,n]g[i+m, j+n]$$

Similarity Measures:

- **Sum of Squared Differences (SSD)**:

$$\text{SSD}(f,g) = \sum_{i,j}(f[i,j] - g[i,j])^2$$

- **Normalized Cross-Correlation (NCC)**:

$$\text{NCC}(f,g) = \frac{\sum_{i,j} f[i,j]g[i,j]}{\sqrt{\sum_{i,j}(f[i,j])^2}\sqrt{\sum_{i,j} g[i,j]^2}}$$

- Census Transform and Hamming Distance: The Census Transform is a non-parametric feature descriptor that encodes the local neighborhood of a pixel. The Hamming Distance is used to compare two Census Transforms.

Moravec Corner Detector: Key idea: Corners are points where the image intensity changes in all directions.

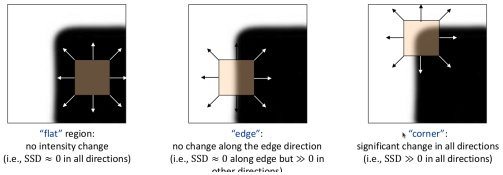

Figure 10: Moravec Corner Detector.

$$SSD(\Delta x, \Delta y) = \sum_{i,j}(I[i,j] - I[i+\Delta x, j+\Delta y])^2 \qquad (1)$$

Harris Corner Detector

1.**Computer Image Gradients:** Calculate the partial derivatives of the Image with resepct to x and y, resulting in the gradient images I_x and I_y.
2.**Construct Second Moment Matrix:** For each pixel, compute the elements of the second moment matrix M:

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

where $w(x,y)$ is a window function.
3. **Compute Corner Response Function:** Compute the corner response function R:

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

where k is an empirically determined constant.
4. **Non-maximum Suppression:** Apply non-maximum suppression to the corner response function.
5. **Thresholding:** Apply a threshold to the corner response function to obtain the final corner detections.

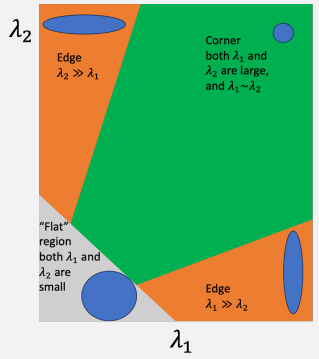

Figure 11: Harris Corner Detector.

Scale-Invariant Feature Transform (SIFT)

Building a Space-Scale Pyramid: The initial image is incrementally convolved with Gaussian $G(k^i\sigma)$ to produce blurred images separated by a constant factor k.

Adjacent blurred images are then subtracted to produce Difference of Gaussians (DoG) images.

Detecting Extrema in Scale-Space: Extrema are detected in the DoG images by comparing each pixel to its 26 neighbors in the current image and the two adjacent images at the same scale.

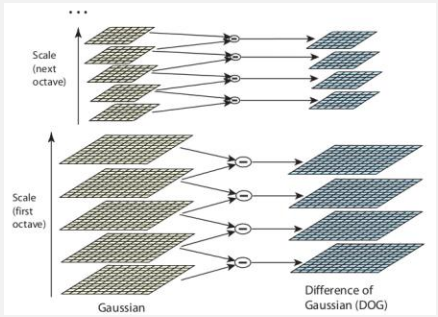

Figure 12: SIFT.

Multi-View Geometry

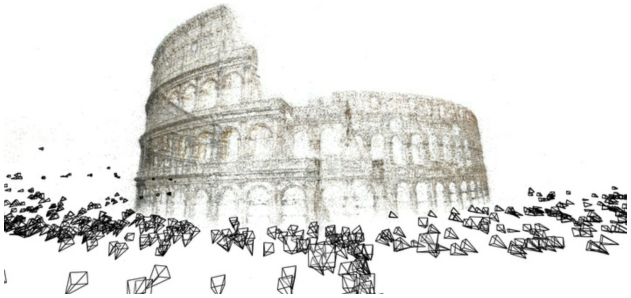

Figure 13: Building Rome in a Day.

Depth Estimation

Depth from Stereo (Simple Case):

- Assumptions: The cameras are calibrated and the baseline B is known.
- Goal: Given two images of the same scene, estimate the depth.

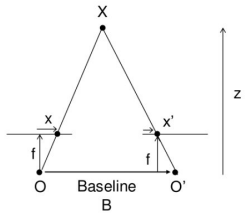

Figure 14: Stereo Vision.

Disparity: The difference in horizontal position of the same point in the two images.

$$d = x - x' = \frac{Bf}{Z}$$

Depth from Stereo (General Case):

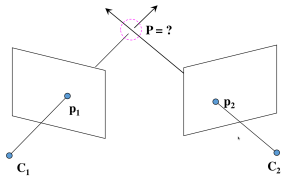

Figure 15: Triangulation.

Least Square Approximation: We construct the system of equations of the left and right cameras and solve it using the least square method.
Nonlinear Refinement: We can refine the depth estimate by minimizing the reprojection error.

Epipolar Geometry:

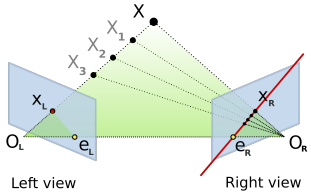

Figure 16: Epipolar Geometry.

- **Epipolar Line**: The line on which the projection of a point in one image lies.
- **Epipole**: The point where the line connecting the two camera centers intersects the image plane.
- **Epipolar Constraint**: The epipolar line in one image is the projection of the corresponding point in the other image.

Stereo Rectification: Stereo rectification is the process of transforming the two images so that the epipolar lines are aligned.

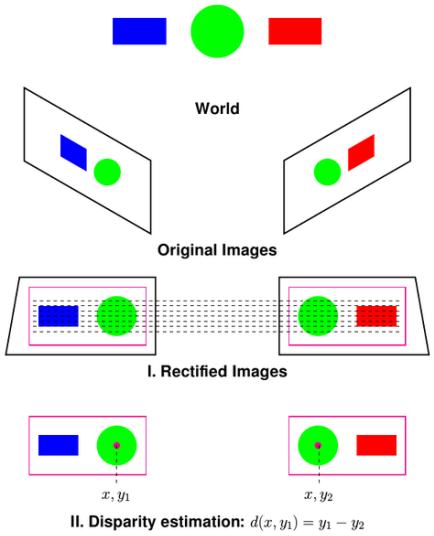

Figure 17: Stereo Rectification.

Structure from Motion (SfM)

Problem Formulation: Given a set of n point correspondences between two images, $p_1^i = (u_1^i, v_1^i)$ and $p_2^i = (u_2^i, v_2^i)$, where p_1^i and p_2^i are the projections of the same 3D point $P^i = (X^i, Y^i, Z^i)$,

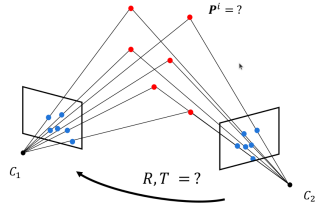

Figure 18: Structure from Motion.

- estimate the 3D points P^i,
- estimate the camera poses $[R|t]$,
- and estimate the camera intrinsics K_1, K_2 that satisfy:

$$\lambda_1 \begin{bmatrix} u_1^i \\ v_1^i \\ 1 \end{bmatrix} = K_1[I|0] \begin{bmatrix} X^i \\ Y^i \\ Z^i \\ 1 \end{bmatrix}, \quad \lambda_2 \begin{bmatrix} u_2^i \\ v_2^i \\ 1 \end{bmatrix} = K_2[R|T] \begin{bmatrix} X^i \\ Y^i \\ Z^i \\ 1 \end{bmatrix}$$

Case 1: Calibrated Camerars (K_1, K_2 known):

- $4n$ knowns: n correspondences, each one $(u_1^i, v_1^i), (u_2^i, v_2^i)$.
- $5 + 3n$ unknowns:$3n$ is the number of coordinates of the 3D points, and 5 is for motion up to scale (3 for rotation, 2 for translation).

Epipolar Constraint: The epipolar constraint is the relationship between the two images of a 3D point.

$$p_2^T [T]_x R p_1 = 0$$

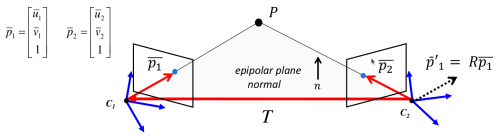

Figure 19: Epipolar Constraint.

Essential Matrix: The essential matrix E is a 3×3 matrix that encapsulates the relative pose between two cameras. It is defined up to scale.

$$E = [T]_x R$$

8-Point Algorithm

Given n point correspondences between two images, the 8-point algorithm proceeds as follows:
1. **Construct the Linear System:** For each correspondence i, construct the equation $p_2^T E p_1 = 0$.

$$\begin{bmatrix} u_1^i u_2^i & u_1^i v_2^i & u_1^i & v_1^i u_2^i & v_1^i v_2^i & v_1^i & u_2^i & v_2^i & 1 \end{bmatrix} \begin{bmatrix} e_{11} \\ e_{12} \\ e_{13} \\ e_{21} \\ e_{22} \\ e_{23} \\ e_{31} \\ e_{32} \\ e_{33} \end{bmatrix} = 0$$

For n correspondences, we get n Equations:

$$Q \cdot E = 0$$

2. **Solve for E:** Solve the linear system using the least squares method.
3. **Enforce Rank-2 Constraint:** Enforce the rank-2 constraint on E by performing a singular value decomposition and setting the smallest singular value to zero.
4. **Recover R and T:** Recover the rotation matrix R and translation vector T from E.

Case 2: Uncalibrated Cameras (K_1, K_2 unknown):
Fundamental Matrix: The fundamental matrix F is a 3×3 matrix that encapsulates the relationship between the two images of a 3D point. It is defined up to scale.

$$\begin{bmatrix} u_2^i \\ v_2^i \\ 1 \end{bmatrix}^T K_2^{-T} E K_1^{-1} \begin{bmatrix} u_1^i \\ v_1^i \\ 1 \end{bmatrix} = 0$$

$$p_2^T F p_1 = 0$$

$$\text{where } \underbrace{F = K_2^{-T} E K_1^{-1}}_{\text{Fundamental Matrix}}$$

The same 8-point algorithm can be used to estimate the fundamental matrix F.

Robust Structure from Motion

RANSAC

RANSAC (Random Sample Consensus) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers. Commond models include lines, planes, and fundamental matrices used in stereo vision.
1. **Random Sampling:** Randomly select the minimum number of data points required to estimate the model parameters.
2. **Model Fitting:** Fit the model to the selected data points.
3. **Model Evaluation:** Determine how well the model fits the data points not used to estimate the model.
4. **Model Refinement:** If the model fits the data well, re-estimate the model using all the inliers.
5. **Model Selection:** Repeat the process and select the model that best fits the data.

- 8-point RANSAC: Estimate the fundamental matrix.
- 5-point RANSAC: Estimate the essential matrix.
- 2-point RANSAC: Line fitting.

Bundle Adjustment

Bundle Adjustment is the process of refining the estimated camera poses and 3D points to minimize the reprojection error. It is an optimization problem that minimizes the difference between the observed image points and the projected 3D points.

$$P^i, C_2, \cdots, C_n = \sum_{k=1}^{n} \sum_{i=1}^{N} \|p_k^i - \pi(P^i, K_k, C_k)\|^2$$

where P^i are the 3D points, C_k are the camera poses, p_k^i are the observed image points, and π is the projection function.

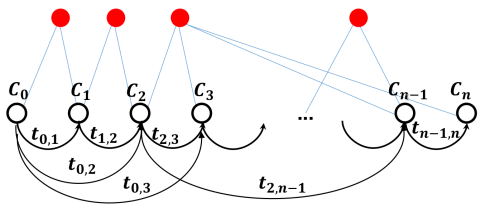

Figure 20: Bundle Adjustment.

Sequential SfM (Visual Odometry)

VO computes the camera path incrementally by estimating the camera poses between consecutive frames.

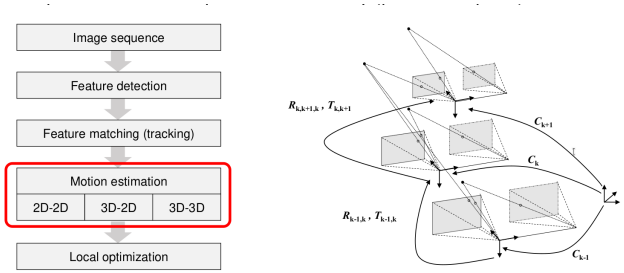

Figure 21: Visual Odometry.

KLT Tracking

The KLT (Kanade-Lucas-Tomasi) algorithm is a feature tracking algorithm that tracks a set of feature points across frames.
1. **Feature Detection:** Detect feature points in the first frame.
2. **Feature Tracking:** Track the feature points across frames using the Lucas-Kanade algorithm.
3. **Outlier Rejection:** Remove outliers using RANSAC.
4. **Pose Estimation:** Estimate the camera pose using the tracked feature points.

Dense 3D Reconstruction

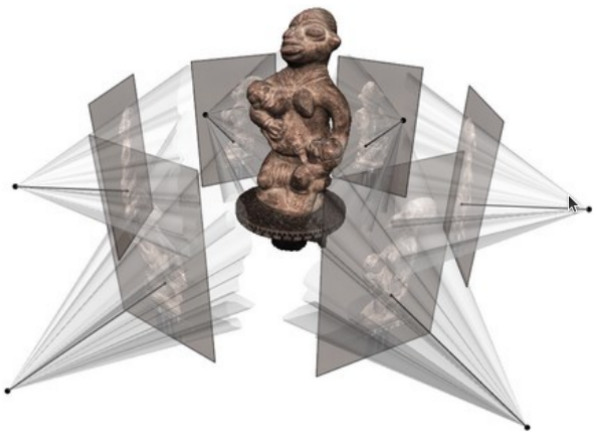

Figure 22: 3D Reconstruction.

Problem Formulation: Given a set of n images of a scene, estimate the 3D structure of the scene and the camera poses.

Aggregated Photometric Error:

- Photometric Consistency: The intensity of a pixel in one image should be consistent with the intensity of the corresponding pixel in another image.
- Error Aggregation: Sum of the photometric errors over all images.
- Pairwise Photometric Error: The photometric error between two images.

$$e(x, y) = |I_1(x, y) - I_2(x, y)|$$

- Aggregated Photometric Error: The sum of the pairwise photometric errors over all image pairs.

$$E = \sum_{i,j} e_{ij}$$

Disparity Space Image (DSI): The Disparity Space Image is a 3D volume that stores the photometric error between two images at different disparities.

$$C(u, v, d) = \sum_{k=R+1}^{R+n-1} \rho(I_R(u, v) - I_k(u', v', d))$$

where $C(u, v, d)$ is the cost volume, $I_R(u, v)$ is patch in the reference image I_R, and depth hypothesis d.

$$I_k(u', v', d) = I_k(\pi(T_{k,R}(\pi^{-1}(u, v, d))))$$

where $T_{k,R}$ is the transformation between frames k and R.

Depth Map Estimation: The solution to the depth map estimation problem is the disparity value that minimizes the cost volume.

$$d^*(u, v) = \text{argmin}_d C(u, v, d(u, v))$$

Visual Inertial Odometry

IMU: An Inertial Measurement Unit (IMU) is a sensor that measures the linear acceleration and angular velocity of a device.

The IMU measurement model is given by:

$$\hat{\omega}_B(t) = \underbrace{\omega_B(t)}_{\text{true}} + b_\omega(t) + n_\omega(t)$$

$$\hat{a}_B(t) = R(t)(\underbrace{a_W(t)}_{\text{true}} - g) + b_a(t) + n_a(t)$$

- $\omega_B(t)$: Angular velocity in the body frame.
- $a_B(t)$: Linear acceleration in the body frame.
- $b_\omega(t)$, $b_a(t)$: Gyroscope and accelerometer biases.
- $n_\omega(t)$, $n_a(t)$: Gyroscope and accelerometer noise.
- $R(t)$: Rotation matrix from the world to the body frame.
- g: Gravity vector.

Loosely-Coupled VIO:

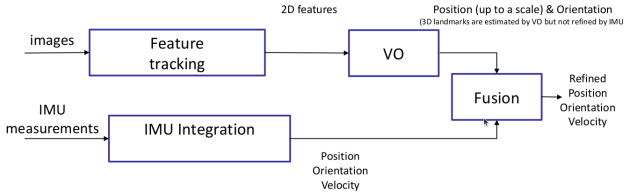

Figure 23: Loosely-Coupled VIO.

Tightly-Coupled VIO:

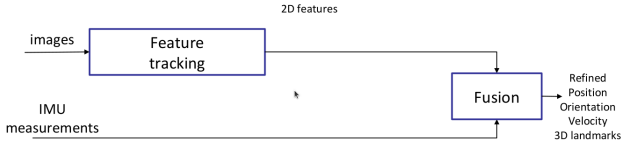

Figure 24: Tightly-Coupled VIO.

Event-based Vision

Event-based cameras are bio-inspired sensors that capture the intensity changes in the scene. They are particularly useful for high-speed applications.

Advantages

- High temporal resolution.
- Low latency.
- High dynamic range.
- Low power consumption.

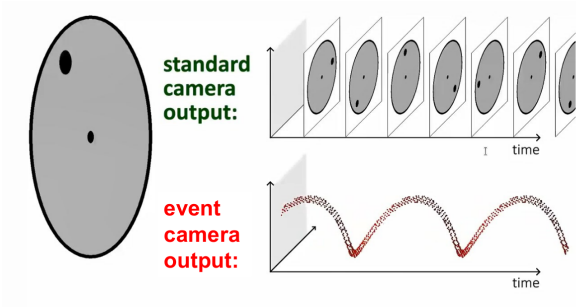

Figure 25: Event-based Vision.

Generative Event Camera Model: Consider the intensity at a single pixel (x, y). An even tis generated when the following condition is met:

$$\log I(x, y, t + \Delta t) - \log I(x, y, t) = \pm C$$

- $I(x, y, t)$: Intensity at pixel (x, y) at time t.
- C: Constant threshold.

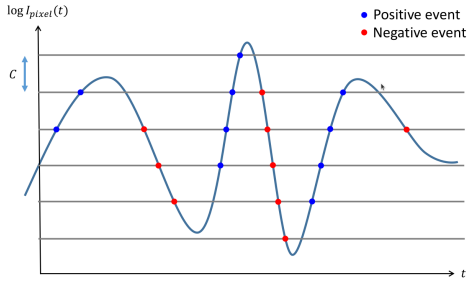

Figure 26: Event Camera Model.

DAVIS Sensor Event + Images + IMU