3D Computer Vision - Lecture Notes

Table of Contents

---

1. Projective Geometry and Camera Matrix

1.1 Homogeneous Coordinates

Definition: A point in $\mathbb{R}^n$ is represented in homogeneous coordinates by a vector in $\mathbb{R}^{n+1}$, defined up to scale.

For a 2D point (x, y):

$$\begin{pmatrix} x \\ y \end{pmatrix} \sim \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix}, \quad \lambda \neq 0$$

For a 3D point (X, Y, Z):

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \sim \begin{pmatrix} \lambda X \\ \lambda Y \\ \lambda Z \\ \lambda \end{pmatrix}, \quad \lambda \neq 0$$

Notation: We write $x \sim y$ to mean $x = \lambda y$ for some $\lambda \neq 0$.

Points at infinity: Homogeneous coordinates allow representation of points at infinity:

- 2D: $(x, y, 0)^T$ represents direction (x, y)
- 3D: $(X, Y, Z, 0)^T$ represents direction (X, Y, Z)

1.2 Projective Transformations

2D Projective Transformation (Homography):

$$x' = Hx$$

where $H \in \mathbb{R}^{3\times3}$ is an invertible matrix, $x, x' \in \mathbb{R}^3$ (homogeneous 2D coordinates).

Properties:

- Preserves collinearity (lines remain lines)

- Does NOT preserve distances or angles in general
- Has 8 degrees of freedom (9 parameters, scale arbitrary)

3D Projective Transformation:

$$X' = GX$$

where $G \in \mathbb{R}^{4\times4}$ is invertible, $X, X' \in \mathbb{R}^4$ (homogeneous 3D coordinates).

1.3 The Pinhole Camera Model

Perspective Projection: Maps 3D world points to 2D image points.

Basic model (camera at origin, looking along z-axis):

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} fX/Z \\ fY/Z \end{pmatrix}$$

In homogeneous coordinates:

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Then $(x/w, y/w)$ are the image coordinates.

1.4 Camera Calibration Matrix

Intrinsic Parameters Matrix:

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

where:

- f_x, f_y: focal lengths in pixel units (x and y directions)
- (c_x, c_y): principal point (image center)
- s: skew parameter (usually 0 for modern cameras)

Typical simplification: $K = \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix}$

Degrees of freedom: 5 (or 4 if square pixels assumed)

1.5 General Camera Model

Extrinsic Parameters: Camera position and orientation in world coordinates

- Rotation: $R \in SO(3)$ (3 DOF)
- Translation: $T \in \mathbb{R}^3$ (3 DOF)

Complete Projection:

$$x = K[R \mid T]X = K(RX + T)$$

where $X \in \mathbb{R}^3$ (world coordinates), $x \in \mathbb{R}^3$ (homogeneous image coordinates).

Projection Matrix (for homogeneous world coordinates):

$$P = K[R \mid T] \in \mathbb{R}^{3\times 4}$$

$$x = PX$$

Degrees of freedom: 11 total

- 5 intrinsic (or 4)
- 6 extrinsic

1.6 Camera Center

Definition: The camera center $C \in \mathbb{R}^4$ (homogeneous) is the 3D point that projects to all image points (pre-image of projection).

Property: $PC = 0$

Computing Camera Center: For $P = \begin{pmatrix} P_1^T \\ P_2^T \\ P_3^T \end{pmatrix}$ where $P_i \in \mathbb{R}^4$:

The camera center satisfies:

$$P_i^T C = 0, \quad i = 1, 2, 3$$

This is the null space of P.

Explicit formula: If $P = [M \mid m]$ with $M \in \mathbb{R}^{3\times 3}$, $m \in \mathbb{R}^3$:

$$C = \begin{pmatrix} -M^{-1}m \\ 1 \end{pmatrix}$$

Using determinants:

$$C_i = (-1)^{i+1} \det(P_{\setminus i})$$

where $P_{\setminus i}$ is P with the i-th column removed.

1.7 Canonical Camera Configuration

Normalization: Without loss of generality, we can set:

$$P_0 = (I_3 \mid 0) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

This corresponds to:

- Camera at world origin
- Aligned with world coordinate system
- Identity calibration matrix

Justification: Any stereo/multi-view configuration can be brought to this form by:

1. Applying a 3D projective transformation to the scene
2. Multiplication by calibration matrices

1.8 Panorama Construction

Goal: Stitch multiple images into a single wide-angle panorama.

Principle: Images related by a homography when:

- Camera rotates around its optical center (pure rotation)
- Scene is planar
- Camera is very far from the scene

Homography Estimation **Problem**: Given correspondences (x_i, x_i'), find H such that $x_i' = Hx_i$.

Direct Linear Transform (DLT):

For each correspondence:

$$\begin{pmatrix} x_i' \\ y_i' \\ w_i' \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ w_i \end{pmatrix}$$

Cross product gives:

$$x_i' \times Hx_i = 0$$

This yields 2 independent linear equations in the 9 unknowns of H.

Matrix form:

$$\begin{pmatrix} 0 & -w_i' x_i^T & y_i' x_i^T \\ w_i' x_i^T & 0 & -x_i' x_i^T \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = 0$$

Solution:

- Need at least 4 correspondences (8 equations for 8 DOF)
- Stack equations: $Ah = 0$ where A is $2n \times 9$
- Solve via SVD: h is last column of V in $A = UDV^T$
- Reshape h into 3×3 matrix

Normalization Problem: Numerical instability if coordinates have very different scales.

Solution: Normalize coordinates before DLT:

1. Translate so centroid is at origin
2. Scale so average distance to origin is $\sqrt{2}$

$$T = \begin{pmatrix} s & 0 & -s\bar{x} \\ 0 & s & -s\bar{y} \\ 0 & 0 & 1 \end{pmatrix}$$

Algorithm:

1. Normalize: $\tilde{x}_i = Tx_i$, $\tilde{x}'_i = T'x'_i$
2. Compute $\tilde{H}$ from $(\tilde{x}_i, \tilde{x}'_i)$
3. Denormalize: $H = T'^{-1}\tilde{H}T$

Image Warping and Blending Warping: Transform each image by its homography:

$$I'(x') = I(H^{-1}x')$$

Use bilinear or bicubic interpolation for non-integer coordinates.

Blending strategies:

1. **Simple average**: $I_{\text{pano}} = \frac{1}{n}\sum_i I_i$ (visible seams)
2. **Feathering**: Linear blend in overlap regions
3. **Multi-band blending**: Blend low frequencies over wide region, high frequencies narrowly

Seam placement: Choose optimal seam through overlap region minimizing difference.

---

2. Fundamental and Essential Matrices

2.1 Epipolar Geometry

Epipolar Constraint: For corresponding points x (in image 1) and x' (in image 2) viewing the same 3D point X:

The three points C (camera center 1), C' (camera center 2), and X are coplanar.

Epipolar plane: Plane containing C, C', and X.

Epipole:

- e: projection of C' in image 1
- e': projection of C in image 2

Epipolar line:

- In image 1: line from e through x
- In image 2: line from e' through x'
- Corresponding point x' must lie on epipolar line in image 2

2.2 Fundamental Matrix

Definition: The fundamental matrix $F \in \mathbb{R}^{3\times 3}$ relates corresponding points:

$$x'^T F x = 0$$

Properties:

- Rank 2 (det $F = 0$)
- 7 degrees of freedom
- Encodes the epipolar constraint
- Depends only on camera parameters (intrinsic + relative pose)

Epipolar lines:

- Line in image 2 corresponding to x: $l' = Fx$
- Line in image 1 corresponding to x': $l = F^T x'$

Epipoles:

- $Fe = 0$ (right epipole)
- $F^T e' = 0$ (left epipole)

2.3 Essential Matrix

Definition: For **calibrated** cameras (known K, K'), the essential matrix E relates normalized coordinates:

$$\hat{x}'^T E \hat{x} = 0$$

where $\hat{x} = K^{-1}x$ (normalized coordinates).

Relation to Fundamental Matrix:

$$E = K'^T F K$$

Decomposition: For cameras with $P = K[I \mid 0]$ and $P' = K'[R \mid T]$:

$$E = [T]_\times R$$

where $[T]_\times$ is the skew-symmetric matrix of T:

$$[T]_\times = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}$$

Properties:

- Rank 2
- 5 degrees of freedom (3 for rotation, 2 for translation direction)
- Two singular values equal, one zero
- $\det(E) = 0$

2.4 Eight-Point Algorithm

Goal: Estimate F from point correspondences (x_i, x_i').

Linear constraint: Each correspondence gives:

$$x_i'^T F x_i = 0$$

Expanding:

$$(x_i' y_i' w_i') \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ w_i \end{pmatrix} = 0$$

$$x_i' x_i f_{11} + x_i' y_i f_{12} + \cdots + w_i' w_i f_{33} = 0$$

Matrix form: Stack equations from n correspondences:

$$Af = 0$$

where A is $n \times 9$:

$$A = \begin{pmatrix} x_1' x_1 & x_1' y_1 & x_1' w_1 & y_1' x_1 & y_1' y_1 & y_1' w_1 & w_1' x_1 & w_1' y_1 & w_1' w_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Algorithm:

1. **Normalize** coordinates: $\tilde{x}_i = T x_i$, $\tilde{x}_i' = T' x_i'$
2. **Construct** matrix A from normalized correspondences
3. **Solve** $Af = 0$ via SVD: f is last column of V
4. **Reshape** f into 3×3 matrix $\tilde{F}$
5. **Enforce rank constraint**:
   - SVD: $\tilde{F} = UDV^T$ with $D = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$
   - Set $D' = \text{diag}(\sigma_1, \sigma_2, 0)$
   - $\tilde{F} = UD'V^T$

6. **Denormalize**: $F = T'^T \tilde{F} T$

Minimum points: 8 correspondences (hence "eight-point"), but more are better.

2.5 Recovering Camera Pose from Essential Matrix

Given: Essential matrix E, correspondences $(\hat{x}_i, \hat{x}'_i)$ in normalized coordinates.

Goal: Extract rotation R and translation T (up to scale).

SVD decomposition: $E = UDV^T$ where $D = \text{diag}(1, 1, 0)$ (after normalization).

Four solutions:

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

1. $R = UWV^T$, $T = U(:, 3)$
2. $R = UWV^T$, $T = -U(:, 3)$
3. $R = UW^TV^T$, $T = U(:, 3)$
4. $R = UW^TV^T$, $T = -U(:, 3)$

Disambiguation: Check that reconstructed 3D points have positive depth in both cameras. Only one solution satisfies this.

Verification: Ensure $\det(R) = 1$ (if -1, negate R).

2.6 RANSAC Algorithm

Problem: Robust estimation in presence of outliers.

Principle:

- Randomly sample minimal set
- Fit model
- Count inliers (points consistent with model)
- Iterate and keep best

RANSAC for Fundamental Matrix

```
Input: Correspondences (x_i, x'_i), threshold delta, iterations N
Output: Fundamental matrix F, inlier set

best_F = null
best_inliers = []
best_count = 0

for iter = 1 to N:
```

```
    # 1. Sample minimal set
    sample = randomly select 8 correspondences

    # 2. Compute model
    F = eight_point_algorithm(sample)

    # 3. Compute inliers
    inliers = []
    for i = 1 to n:
        # Compute Sampson distance
        d = (x'_i^T F x_i)^2 / (||F^T x'_i||^2 + ||F x_i||^2)

        if d < delta^2:
            add i to inliers

    # 4. Update best
    if |inliers| > best_count:
        best_F = F
        best_inliers = inliers
        best_count = |inliers|

# 5. Refine with all inliers
F_final = eight_point_algorithm(best_inliers)

return F_final, best_inliers
```

Distance metrics:

1. **Symmetric epipolar distance**:

$$d(x, x', F) = d(x', Fx)^2 + d(x, F^T x')^2$$

   where $d(x, l)$ is point-to-line distance.

2. **Sampson distance** (first-order approximation):

$$d_S = \frac{(x'^T F x)^2}{||F^T x'||^2 + ||Fx||^2}$$

Number of Iterations Probability formulation:

- w: inlier ratio
- s: sample size (e.g., 8 for fundamental matrix)
- p: probability of success (e.g., 0.99)

Probability that one sample is all inliers: w^s

Probability that all N samples contain outlier: $(1 - w^s)^N$

To achieve success probability p:

$$(1 - w^s)^N = 1 - p$$

$$N = \frac{\log(1-p)}{\log(1-w^s)}$$

Example: $w = 0.5$, $s = 8$, $p = 0.99$: $N \approx 1177$ iterations

2.7 Advanced RANSAC Variants

LO-RANSAC (Locally Optimized RANSAC) **Enhancement**: Refine hypothesis using all inliers.

```
for each RANSAC hypothesis F:
    inliers = find_inliers(F, delta)

    repeat:
        F_new = eight_point_algorithm(inliers)
        inliers_new = find_inliers(F_new, delta)

        if |inliers_new| <= |inliers|:
            break

        F = F_new
        inliers = inliers_new
```

Benefit: Better convergence, fewer iterations needed.

PROSAC (Progressive Sample Consensus) **Idea**: Use ordering of matches by quality.

- Sample from highest-quality matches first
- Progressively expand to lower-quality matches

MSAC (M-estimator SAC) **Idea**: Minimize cost instead of counting inliers:

$$\text{cost} = \sum_i \rho(e_i, \sigma)$$

where $\rho(e, \sigma) = \min(e^2, \sigma^2)$ (truncated quadratic).

---

3. Disparity Maps and Correlation

3.1 Stereo Rectification

Goal: Transform images so epipolar lines are horizontal and parallel.

Rectified configuration:

- Epipolar lines parallel to x-axis
- Corresponding points have same y-coordinate
- Search for correspondence is 1D (along scanline)

Rectification transformation: For cameras $P = K[I \mid 0]$ and $P' = K'[R \mid T]$:

1. Rotate both cameras so translation is along x-axis
2. Make image planes coplanar and parallel
3. Apply homographies H and H' to images

Result: $P_{\text{rect}} = \begin{pmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

$$P'_{\text{rect}} = \begin{pmatrix} f & 0 & c_x & -fB \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

where B is the baseline (distance between cameras).

3.2 Disparity

Definition: For rectified images, disparity is the horizontal displacement:

$$d(x, y) = x_L - x_R$$

where (x_L, y) and (x_R, y) are corresponding points in left and right images.

Depth from disparity:

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

where:

- f: focal length
- B: baseline
- d: disparity

Properties:

- Disparity inversely proportional to depth
- Larger disparity = closer objects
- Disparity ≥ 0 for objects in front of cameras

3.3 Disparity Range

Minimum disparity $d_{\min}$: from farthest objects of interest **Maximum disparity** $d_{\max}$: from nearest objects

Computational cost: $O(w \cdot h \cdot (d_{\max} - d_{\min}))$

Discrete disparity: Typically integer pixel values, but sub-pixel refinement possible.

3.4 Correlation-Based Matching

Principle: Match pixels/patches by comparing intensities.

Matching Cost Functions For pixels (x_L, y) in left image and (x_R, y) in right image:

1. **Sum of Absolute Differences (SAD)**:

$$C_{\text{SAD}}(x_L, x_R, y) = \sum_{(i,j)\in W} |I_L(x_L + i, y + j) - I_R(x_R + i, y + j)|$$

2. **Sum of Squared Differences (SSD)**:

$$C_{\text{SSD}}(x_L, x_R, y) = \sum_{(i,j)\in W} (I_L(x_L + i, y + j) - I_R(x_R + i, y + j))^2$$

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

$$C_{\text{NCC}}(x_L, x_R, y) = \frac{\sum_{(i,j)\in W} I_L(x_L + i, y + j) \cdot I_R(x_R + i, y + j)}{\sqrt{\sum (I_L)^2 \cdot \sum (I_R)^2}}$$

4. **Zero-mean NCC (ZNCC)**:

$$C_{\text{ZNCC}} = \frac{\sum (I_L - \bar{I}_L)(I_R - \bar{I}_R)}{\sqrt{\sum (I_L - \bar{I}_L)^2 \cdot \sum (I_R - \bar{I}_R)^2}}$$

where W is the window (patch) around the pixel.

Window size tradeoff:

- Small window: More detail, more noise, less distinctive
- Large window: Smoother, less detail, violates constant disparity assumption

Winner-Takes-All (WTA) **Simple matching**: For each pixel in left image:

$$d(x, y) = \arg \min_{d \in [d_{\min}, d_{\max}]} C(x, x - d, y)$$

Problems:

- Noisy results
- No smoothness constraint
- Errors in textureless regions
- Occlusions not handled

3.5 Occlusions

Problem: Some pixels visible in one image but not the other.

Left-right consistency check:

1. Compute disparity map left→right: $d_L(x, y)$
2. Compute disparity map right→left: $d_R(x, y)$
3. Check consistency: $|d_L(x, y) - d_R(x - d_L(x, y), y)| < \tau$

Pixels failing this test are likely occluded.

Handling occlusions:

- Mark as invalid
- Fill in using neighboring disparities
- Assign high matching cost

3.6 Sub-pixel Refinement

Goal: Improve disparity precision beyond integer pixels.

Parabolic fitting: Fit parabola to 3 cost values around minimum:

$$C(d-1), \quad C(d), \quad C(d+1)$$

Sub-pixel disparity:

$$d_{\text{sub}} = d + \frac{C(d-1) - C(d+1)}{2(C(d-1) - 2C(d) + C(d+1))}$$

3.7 Cost Aggregation

Problem: Matching costs at individual pixels are noisy.

Solution: Aggregate costs over regions.

Box Filter Simple averaging:

$$C_{\text{agg}}(x,d) = \frac{1}{|W|} \sum_{(i,j)\in W} C(x+i, y+j, d)$$

Fast implementation using integral images.

Adaptive Support Weights Weight based on color similarity and spatial distance:

$$C_{\text{agg}}(p,d) = \frac{\sum_{q\in W} w(p,q)C(q,d)}{\sum_{q\in W} w(p,q)}$$

$$w(p,q) = \exp\left(-\frac{\Delta c(p,q)}{\gamma_c} - \frac{\Delta g(p,q)}{\gamma_p}\right)$$

where:

- $\Delta c(p,q)$: color difference
- $\Delta g(p,q)$: spatial distance

Guided Filter Similar to bilateral filter but faster and edge-preserving.

3.8 Dynamic Programming

Idea: Enforce smoothness along scanlines.

Energy per scanline:

$$E(d_1,\ldots,d_w) = \sum_{x=1}^{w} C(x,d_x) + \sum_{x=1}^{w-1} V(d_x, d_{x+1})$$

where $V(d_x, d_{x+1})$ is smoothness penalty.

Solution: Dynamic programming in $O(w \cdot D^2)$ where D is disparity range.

Problem: Streaking artifacts (no inter-scanline consistency).

3.9 Semi-Global Matching (SGM)

Idea: Aggregate costs along multiple directions.

Energy:

$$E(D) = \sum_{p} C(p, D_p) + \sum_{(p,q)\in\mathcal{N}} V(D_p, D_q)$$

Smoothness penalty:

$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ P_1 & \text{if } |d_p - d_q| = 1 \\ P_2 & \text{if } |d_p - d_q| > 1 \end{cases}$$

Aggregation: Sum costs along 8 or 16 directions:

$$S(p, d) = C(p, d) + \sum_{r \in \mathcal{R}} L_r(p, d)$$

where L_r is cost along path from direction r.

Dynamic programming along paths:

$$L_r(p, d) = C(p, d) + \min \begin{cases} L_r(p - r, d) \\ L_r(p - r, d \pm 1) + P_1 \\ \min_i L_r(p - r, i) + P_2 \end{cases}$$

Disparity selection: $D(p) = \arg\min_d S(p, d)$

---

4. Graph Cuts for Disparity Computation

4.1 Energy Minimization Framework

Goal: Find disparity map D minimizing global energy.

Energy function:

$$E(D) = E_{\text{data}}(D) + \lambda E_{\text{smooth}}(D)$$

Data term (fidelity to observations):

$$E_{\text{data}}(D) = \sum_{p \in \mathcal{P}} C(p, D_p)$$

where $C(p, D_p)$ is matching cost for pixel p with disparity D_p.

Smoothness term (spatial coherence):

$$E_{\text{smooth}}(D) = \sum_{(p,q) \in \mathcal{N}} V(D_p, D_q)$$

where $\mathcal{N}$ is set of neighboring pixel pairs.

4.2 Graph Construction

For binary labeling (e.g., foreground/background):

Graph $G = (V, E)$:

- Vertices V: pixels + source s + sink t
- Edges E: n-links (between neighbors) + t-links (to source/sink)

Terminal links (t-links):

- Edge (s, p): capacity = cost of assigning label 0 to pixel p
- Edge (p, t): capacity = cost of assigning label 1 to pixel p

Neighbor links (n-links):

- Edge (p, q): capacity = penalty for p and q having different labels

Labeling from cut:

- Pixel p in source set: label 0
- Pixel p in sink set: label 1

Cut cost = Energy:

$$\text{cost}(S, T) = \sum_{p \in S} \text{cap}(p, t) + \sum_{p \in T} \text{cap}(s, p) + \sum_{\substack{p \in S, q \in T \\ (p,q) \in \mathcal{N}}} V(0, 1)$$

4.3 Max-Flow/Min-Cut Theorem

Theorem: The minimum cut in a graph equals the maximum flow from source to sink.

Algorithms:

1. **Ford-Fulkerson**: Augmenting path method
2. **Push-relabel**: Local operations on vertices
3. **Boykov-Kolmogorov**: Specialized for vision problems

Complexity: Polynomial in graph size (pixels) and flow values.

4.4 Alpha-Expansion for Multi-Label

Problem: Multiple disparity labels (not just binary).

Alpha-expansion move:

- Current labeling: D
- Choose label α
- Each pixel can: keep current label or switch to α

Algorithm:

```
Initialize D arbitrarily

repeat:
    improvement = false

    for each label alpha in disparity range:
        # Binary problem: keep D_p or switch to alpha
        Construct graph:
            t-link (s, p): cost of D_p
            t-link (p, t): cost of alpha
            n!=-link (p, q): V(D_p, alpha) if D_p != alpha

        D_new = min-cut solution

        if E(D_new) < E(D):
            D = D_new
            improvement = true

until not improvement

return D
```

Properties:

- Each iteration decreases energy (or stays same)
- Converges to local minimum
- Quality depends on smoothness penalty V

4.5 Metric and Semi-Metric Smoothness

Metric: $V(d_p, d_q)$ satisfies:

1. $V(a, b) = 0 \Leftrightarrow a = b$
2. $V(a, b) = V(b, a)$ (symmetry)
3. $V(a, c) \leq V(a, b) + V(b, c)$ (triangle inequality)

Common choices:

1. **Potts model** (metric):

$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ K & \text{otherwise} \end{cases}$$

   Encourages piecewise constant (sharp boundaries).

2. **Linear penalty** (metric):

$$V(d_p, d_q) = \lambda |d_p - d_q|$$

Balances smoothness and detail.

3. **Truncated linear** (semi-metric):

$$V(d_p, d_q) = \min(\lambda|d_p - d_q|, K)$$

Allows discontinuities (non-metric).

4. **Quadratic penalty**:

$$V(d_p, d_q) = \lambda(d_p - d_q)^2$$

Over-smooths (blurs boundaries).

Guarantee: Alpha-expansion guarantees solution within factor 2 of global optimum for metric V.

4.6 Regularization Effects

Parameter α in $f_\alpha(x) = |x|^\alpha + |1-x|^\alpha$:

- $\alpha > 1$ **(convex)**: Unique smooth minimum at $x = 1/2$
  - Effect: Strong smoothing, blurred boundaries
- $\alpha = 1$ **(convex)**: All $x \in [0, 1]$ are minima
  - Effect: Total variation, preserves edges
- $0 < \alpha < 1$ **(concave)**: Minima at $x = 0$ and $x = 1$
  - Effect: Promotes sparsity, sharp boundaries

Graph cuts context:

- Lower α: Sharper disparity boundaries
- Higher α: Smoother transitions
- Potts model ($\alpha \to 0$): Sharpest boundaries

Convexity result: For $n > 0$ and $f(0) = 0$:

- Convex f: $f(1) \geq nf(1/n) \to$ smoothing
- Concave f: $f(1) \leq nf(1/n) \to$ sparsifying

4.7 Visibility Constraints and Occlusions

Ordering constraint: For most scenes, disparity order consistent along scanlines.

If $x_1 < x_2$ in left image, then $x_1 - d_1 \leq x_2 - d_2$ in right image (for opaque objects).

Uniqueness constraint: Each pixel in one image matches at most one pixel in the other.

Occlusion cost: Assign penalty for marking pixel as occluded.

Extended graph:

- Add "occluded" label
- Higher cost than matching
- Lower than bad matches

4.8 Practical Implementation

Steps:

1. **Compute matching costs** $C(p, d)$ for all pixels and disparities
   - Use SAD, SSD, or NCC
   - Aggregate if needed
2. **Build graph** for each alpha-expansion:
   - Vertices: one per pixel + source + sink
   - T-links: data costs
   - N-links: smoothness penalties
3. **Solve min-cut** using max-flow algorithm
   - Boykov-Kolmogorov typically fastest
   - Reuse search trees between iterations
4. **Iterate** alpha-expansion until convergence
   - Typically 3-10 iterations
   - Cycle through all disparity labels
5. **Post-process**:
   - Left-right consistency check
   - Fill occlusions
   - Median filter
   - Sub-pixel refinement

Computational cost: $O(n \cdot D \cdot T)$

- n: number of pixels
- D: disparity range
- T: number of alpha-expansion iterations

---

5. Multi-View Stereo

5.1 Problem Formulation

Input:

- Multiple images $I_1, \ldots, I_n$
- Camera parameters $P_1, \ldots, P_n$ (or K_i, R_i, T_i)

Output: Dense 3D reconstruction (depth map or 3D mesh)

Advantages over two-view stereo:

- More constraints → better disambiguation
- Reduced matching ambiguities

- Can handle occlusions better
- More complete reconstruction

5.2 Multi-View Geometry

Trifocal constraint: For a 3D point X viewed as x_1, x_2, x_3 in three cameras:

$$\text{rank}(M) < 2$$

where the **matching matrix** is:

$$M = \begin{pmatrix} [x_2]_\times K_2 R_2 K_1^{-1} x_1 & [x_2]_\times K_2 T_2 \\ [x_3]_\times K_3 R_3 K_1^{-1} x_1 & [x_3]_\times K_3 T_3 \end{pmatrix} \in \mathbb{R}^{6\times 2}$$

Rank interpretation:

- $\text{rank}(M) = 2$: No solution (inconsistent)
- $\text{rank}(M) = 1$: Unique solution (generic case)
- $\text{rank}(M) = 0$: Infinitely many solutions (degenerate: all centers and X collinear)

Coplanarity condition: Rank drops further if camera centers C_1, C_2, C_3 and X are coplanar.

5.3 Depth Map Fusion

Per-view depth maps: Compute depth map for each reference view using other views.

Depth map: $Z(x, y)$ giving depth at each pixel of reference view.

Fusion strategies:

1. **Median**: Robust to outliers

$$Z_{\text{fused}}(X) = \text{median}\{Z_i(X) : X \text{ visible in view } i\}$$

2. **Weighted average**: Based on confidence/angle

$$Z_{\text{fused}}(X) = \frac{\sum_i w_i(X) Z_i(X)}{\sum_i w_i(X)}$$

3. **TSDF (Truncated Signed Distance Function)**:
   - Accumulate signed distances in voxel grid
   - Extract isosurface (Marching Cubes)

5.4 Photo-Consistency Measures

Goal: Measure how well 3D point hypothesis explains observations.

For point X projected to pixels x_i in views i:

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

$$\rho_{\text{NCC}}(X) = \frac{1}{n(n-1)} \sum_{i \neq j} \text{NCC}(P_i(X), P_j(X))$$

2. **Variance**:

$$\rho_{\text{var}}(X) = \text{Var}(I_1(x_1), \ldots, I_n(x_n))$$

   Lower variance = more consistent.

3. **Robust function**:

$$\rho(X) = \sum_{i,j} \rho_{\text{robust}}(I_i(x_i) - I_j(x_j))$$

Visibility: Only consider views where X is visible (not occluded).

5.5 Plane Sweep Algorithm

Idea: Test 3D hypotheses on planes parallel to reference view.

Algorithm:

```
For reference view R:
    For each depth d in [d_min, d_max]:
        # Define plane at depth d
        Plane: pi_d parallel to image plane at depth d

        For each other view i:
            # Warp to reference view
            H_i = homography mapping plane pi_d from view i to R
            I'_i = warp(I_i, H_i)

        # Compute photo-consistency
        For each pixel (x, y) in R:
            C(x, y, d) = photo_consistency(I_R(x,y), {I'_i(x,y)})

    # Select best depth per pixel
    For each pixel (x, y):
        D(x, y) = argmin_d C(x, y, d)
```

Advantages:

- Efficient (homographies computed per plane, not per pixel)

- GPU-friendly (parallel warping)
- Multiple views naturally incorporated

Homography for plane: Given plane $\pi : n^T X = d$:

$$H_{i \to R} = K_R R_{iR} (I - \frac{1}{d} T_{iR} n^T) K_i^{-1}$$

5.6 Volumetric Methods

Space carving:

1. Start with bounding volume
2. Test voxels for photo-consistency
3. Remove inconsistent voxels
4. Extract surface

Level sets: Evolve surface to minimize energy.

Graph cuts in 3D:

- Voxel grid
- Binary labeling (inside/outside)
- Minimize energy with data + smoothness terms

5.7 Patch-Based MVS (PMVS)

Key ideas:

- Represent surface as collection of oriented patches
- Match patches across views
- Expand and filter

Patch representation: (X, n) where X is center, n is normal.

Matching score:

$$g(p) = \sum_{i \in V(p)} \text{NCC}(p, I_i) \cdot w_i$$

where $V(p)$ is set of views seeing patch p.

Algorithm:

1. **Initial matching**: Detect features, match across views
2. **Expansion**: Grow patches into textureless regions
3. **Filtering**: Remove patches with low score or inconsistency

5.8 Depth Map Estimation with Multi-View

PatchMatch Stereo: Randomized search + propagation.

For each pixel in reference view:

1. **Initialization**: Random depth and normal

$$Z(x, y) \sim U[Z_{\min}, Z_{\max}]$$

$$n(x, y) \sim \text{random unit vector}$$

2. **Propagation**: Try neighbors' estimates

```
for neighbor (x', y') of (x, y):
    Z_cand = Z(x', y')
    n_cand = n(x', y')

    if cost(Z_cand, n_cand) < cost(Z(x,y), n(x,y)):
        Z(x, y) = Z_cand
        n(x, y) = n_cand
```

3. **Random search**: Try random perturbations

```
deltaZ = delta_Z
while deltaZ > min_deltaZ:
    Z_cand = Z(x, y) + U[-deltaZ, deltaZ]
    n_cand = n(x, y) + random perturbation

    if cost(Z_cand, n_cand) < cost(Z(x,y), n(x,y)):
        Z(x, y) = Z_cand
        n(x, y) = n_cand

    deltaZ = deltaZ / 2
```

4. **Iterate** steps 2-3 several times

Cost function: Combine multiple views

$$C(x, y, Z, n) = \sum_{i \in \mathcal{V}} w_i \cdot \rho(x, y, Z, n, I_i)$$

5.9 Learning-Based MVS

Deep learning approaches:

1. **Cost volume construction**:
   - Warp features to reference view
   - Build 4D cost volume (x, y, depth, features)
   - Aggregate with 3D convolutions
2. **Cost aggregation**:
   - 3D CNN to regularize cost volume
   - Encode contextual information
3. **Depth regression**:
   - Soft argmin over cost volume
   - Or classification + refinement

Examples: MVSNet, R-MVSNet, CasMVSNet

Advantages:

- End-to-end learning
- Better handling of textureless regions
- State-of-the-art accuracy

Challenges:

- Memory for high-resolution
- Generalization to new scenes
- Requires training data

5.10 Viewing Graph and Solvability

Viewing graph:

- Vertices: cameras
- Edges: sufficient correspondences to compute F

Solvable viewing graph: Fixing $P_0 = (I_3 \mid 0)$ determines all other P_i.

Properties:

- Must be biconnected (2-connected)
- Not solvable if 3 camera centers collinear
- For general position cameras, need sufficient connectivity

Matrix formulation: For edge (i, j), matrix $G_{ij} \in \mathbb{R}^{4\times4}$ relates cameras:

$$P_i G_{ij} \sim P_j$$

Solvability condition:

$$v_{hij} = 0$$

for all vertex i with edges to h and j.

This means: $G_{hi}G_{ij}^{-1} = a_{hij}I_4 + c(P_i)v_{hij}^T$ with $v_{hij} = 0$.

5.11 Multi-View Reconstruction Pipeline

Complete workflow:

1. **Image acquisition**: Capture overlapping images
2. **Structure from Motion (SfM)**:
    - Feature detection and matching
    - Incremental or global reconstruction
    - Bundle adjustment
    - Output: camera poses, sparse point cloud
3. **Dense matching**:

- For each view, compute depth map
- Use other views for matching
- Handle occlusions and outliers

4. **Depth map fusion**:
   - Merge depth maps from all views
   - Resolve inconsistencies
   - Output: fused 3D point cloud or mesh
5. **Mesh reconstruction**:
   - Poisson surface reconstruction
   - Or: Marching Cubes on TSDF
   - Texturing
6. **Refinement**:
   - Bundle adjustment with depth maps
   - Mesh optimization

Software: COLMAP, OpenMVS, AliceVision (Meshroom)

5.12 Challenges and Future Directions

Current challenges:

- Textureless regions
- Reflective/transparent surfaces
- Dynamic scenes
- Large-scale scenes
- Real-time performance

Active research:

- Neural rendering (NeRF, Gaussian Splatting)
- Self-supervised learning
- Semantic integration
- SLAM integration
- Uncertainty quantification

Summary Tables

Camera Models Comparison

Model	DOF	Matrix Size	Parameters
Uncalibrated projection	11	3×4	All extrinsic + intrinsic
Calibrated projection	6	3×4	Only extrinsic (R, T)
Fundamental matrix	7	3×3	Epipolar geometry

Model	DOF	Matrix Size	Parameters
Essential matrix	5	3×3	Calibrated epipolar
Homography	8	3×3	Planar transformation

Matching Costs

Method	Formula	Properties
SAD	$\sum \|I_L - I_R\|$	Fast, sensitive to brightness
SSD	$\sum (I_L - I_R)^2$	Smooth, quadratic penalty
NCC	$\frac{\sum I_L \cdot I_R}{\sqrt{\sum I_L^2 \sum I_R^2}}$	Illumination invariant
ZNCC	NCC with zero mean	Best invariance, slower

Regularization Terms

Type	Formula	Effect
Potts	$K \cdot [d_p \neq d_q]$	Sharp boundaries
Linear	$\lambda \|d_p - d_q\|$	Balanced smoothness
Truncated	$\min(\lambda \|d_p - d_q\|, K)$	Preserves discontinuities
Quadratic	$\lambda (d_p - d_q)^2$	Over-smoothing

End of Lecture Notes