Chapter 6

Stereo Correspondence Algorithms
\[ \frac{x'_l}{f} = \frac{x + b/2}{z} \quad \frac{x'_r}{f} = \frac{x - b/2}{z} \]

\[ \Rightarrow \quad \frac{x'_l - x'_r}{f} = \frac{b}{z} \]

Disparity \[ x'_l - x'_r = \frac{bf}{z} \propto \frac{1}{z} \]
1. **Stereo Matching Using Constraints**

**Depth from Stereo Images**

*Correspondence problem:*

Matching entities in the left image with corresponding entities in the right image.

- **Area-based matching**
  
  Primitives: pixels
  
  - Can use e.g., *SSD (Sum of Square Differences)* between pixel intensities in candidate windows
  
  + immediate dense depth map
  
  - too many entities to match

- **Feature-based matching**
  
  Primitives: features, e.g. lines, curves, etc.
  
  + alleviate the matching problem
  
  - need surface interpolation afterwards
Correspondence Using Window-based matching

SSD – Sum of Square Differences

Problems with Window-based matching

- Disparity within the window must be constant.
- Bias the results towards frontal-parallel surfaces.
- Blur across depth discontinuities.
- Perform poorly in textureless regions.
- Erroneous results in occluded regions
Possible Constraints for Stereo Matching

- *Epipolar* constraints (e.g., same scan line)

- [Marr]
  - *Compatibility* (e.g., black to black)
  - *Uniqueness* (no more than one match)
  - *Continuity* (disparity)

- [Mayhew & Frisby] *Figural Continuity*

- [Burt & Julesz] *Limit on Disparity Gradient* (e.g., ordering constraint)

- Larger entities (e.g., extended lines, curves) can reduce the matching complexity.
Epipolar constraint

$\overline{0_x, \overline{0_r} \quad \text{— Optical centers}}$

$x_L, y_L, Z_L$ and $x_r, y_r, Z_r \quad \text{— Local camera coordinates}$

$\overline{0_x \overline{0_r}} \quad \text{— Baseline}$

$\overline{\varepsilon_L, \varepsilon_r} \quad \text{— Epipolar lines}$

* Corresponding $\overline{p_r}$ of $\overline{p_L}$ must lie on an epipolar line in $I_r$, and vice versa.

$\overline{\varepsilon_L, \varepsilon_r} \quad \text{— Epipoles}$

* All epipolar lines intersect at $\overline{\varepsilon_L}$ and $\overline{\varepsilon_r}$ in $I_L$ and $I_r$, respectively.
More on "Disparity Gradient"

Suppose two candidate matches:

\[ m_1 = (l_1, r_1) \]
\[ m_2 = (l_2, r_2) \]

\( d(m_1) \) and \( d(m_2) \) are the disparities.

then the disparity gradient

\[ DG = \frac{\left| d(m_1) - d(m_2) \right|}{D(m_1, m_2)} \leq 1, \]

where \( D(m_1, m_2) \) is \( \frac{(x_{l_2} - x_{l_1}) + (x_{r_2} - x_{r_1})}{2} \).
the randomly placed dots of the left-hand image is displaced sideways. The dots which are thus covered are lost, and the space left by displacing the patch is filled in with random dots.

Interestingly enough, a very simple algorithm [Marr and Poggio 1976] can be formulated that computes disparity from random dot stereograms. First consider the simpler problem of matching one-dimensional images of four points as depicted in Fig. 3.24. Although only one depth plane allows all four points to be placed in correspondence, lesser numbers of points can be matched in other planes.

The crux of the algorithm is the rules, which help determine, on a local basis, the appropriateness of a match. Two rules arise from the observation that most images are of opaque objects with smooth surfaces and depth discontinuities only at object boundaries:

1. Each point in an image may have only one depth value. \textit{Uniqueness}.
2. A point is almost sure to have a depth value near the values of its neighbors. \textit{Continuity}.

Fig. 3.23 A random-dot stereogram.

Fig. 3.24 The stereo matching problem.
Figure 3.24 can be viewed as a binary network where each possible match is represented by a binary state. Matches have value 1 and nonmatches value 0. Figure 3.25 shows an expanded version of Fig. 3.24. The connections of alternative matches for a point inhibit each other and connections between matches of equal depth reinforce each other. To extend this idea to two dimensions, use parallel arrays for different values of $y$ where equal depth matches have reinforcing connections. Thus the extended array is modeled as the matrix $C(x, y, d)$ where the point $x, y, d$ corresponds to a particular match between a point $(x_1, y_1)$ in the right image and a point $(x_2, y_2)$ in the left image. The stereopsis algorithm produces a series of matrices $C_n$ which converges to the correct solution for most cases. The initial matrix $C_0(x, y, d)$ has values of one where $x, y, d$ correspond to a match in the original data and has values of zero or otherwise.

**Algorithm 3.2** [Marr and Poggio 1976]

Until $C$ satisfies some convergence criterion, do

$$C_{n+1}(x, y, d) = \left\{ \sum_{x',y',d' \in S} C_n(x', y', d') - \varepsilon \sum_{x',y',d' \in \theta} C_n(x', y', d') + C_0(x, y, d) \right\}$$

where the term in braces is handled as follows:

$$\{ t \} = \begin{cases} 1 & \text{if } t > T \\ 0 & \text{otherwise} \end{cases}$$

$S =$ set of points $x', y', d'$ such that $|x - x'| \leq 1$ and $d = d'$

$\theta =$ set of points $x', y', d'$ such that $|x - x'| \leq 1$ and $|d - d'| = 1$

---

**Fig. 3.25** Extension of stereo matching.
With respect to the "reference image" $xy$, one can have a "disparity space" for all possible matches $(x, y, d)$.

Nodes with the same $(x, y)$ but different $d$'s suppress each other — uniqueness.

In general, use "disparity gradient" to calculate support.
Figure 3: (a) Random-dot stereogram ($64 \times 64$) for a floating rectangle. Disparity value for the rectangle is $d_1 = 6$, for the background is $d_2 = 2$. (b) Random-dot stereogram ($128 \times 128$) for a hemisphere on the background. Disparity value for the top of the sphere is 9, for the background is 1. (c) Iterative relaxation result for (a). Grey level is used to represent disparity values. From iteration 0 to 45, every third iteration result is shown in order (row-major). (d) Iterative relaxation result for (b).
Autostereograms

Divergence (wall-eyed) viewing
Core Code for Generating Autostereograms

By ZNL

• Illustrates how the $i^{th}$ row of an autostereogram can be generated according to a “disparity map” $\text{shift}[i][j]$.

```c
/* Initialization */
for (j = 0; j < COLS + OFFSET; j++)
    done[j] = 0;

/* Sweep each row to make sure the corresponding dots at distance "OFFSET-shift" apart are laid. */
for (j = 0; j < COLS + OFFSET; j++)
    { if (done[j] != 1)
        { num = random () % 2;
          A[i][j] = num * 255;  // "white" is 255.
          done[j] = 1;    // The dot is laid.
        }
        A[i][j + OFFSET - shift[i][j]] = A[i][j];
        done[j + OFFSET - shift[i][j]] = 1;
    }
```
2. Stereo Matching Algorithms

In addition to the Cooperative Stereo studied earlier, we will discuss *Dynamic Programming, Graph Cuts, and Multiple Baseline Stereo*.

2.1 Dynamic Programming

- A technique for solving optimization problem, when not all variables in the evaluation function are interrelated simultaneously.

- The decision stages can be *ordered* so that data needed at a given stage have been processed (prepared) before then.

- The decision process should be *Markovian*, i.e., at any stage the behavior of the process depends solely on the current state (it does not depend on the previous history).

- Usually, dynamic programming has a polynomial time complexity (better than Recursive Methods).
Fig. 8. (a) Trellis labeled with branch lengths; $M = 4$, $K = 5$. (b) Recursive determination of the shortest path via the VA.
Fig. 3. 2D search plane for intra-scanline search. Intensity profiles are shown along each axis. The horizontal axis corresponds to the left scanline and the vertical one corresponds to the right scanline. Vertical and horizontal lines are the edge positions, and path selection is done at their intersections.
Cost Function:

Let $C(\vec{m})$ be the cost of the optimal path from the origin $(0,0)$ to node $\vec{m}$,

$c(\vec{m}, \vec{m}-\vec{i})$ be cost of the primitive path from $\vec{m}-\vec{i}$ to $\vec{m}$.

$C(\vec{0}) = 0$

$C(\vec{m}) = \min_{\{\vec{i}\}} \left[ c(\vec{m}, \vec{m}-\vec{i}) + C(\vec{m}-\vec{i}) \right],$

where $\vec{0}=(0,0)$; $\vec{i}=(i, j)$, $0 \leq i \leq 1$, $0 \leq j \leq 1$, $i+j \neq 0$

The path that renders $C(\vec{G})$ at node $\vec{G}=(m,N)$ is the optimal solution path to $\vec{G}$.
Approach 2: Hough Method

- Matching long lines in the Hough space

  Epipolar Constraints: matching edge points
  \( y_l = y_r. \)

  Therefore, \( \Delta \rho = \Delta x \cos \theta. \)

→ A Simpler Problem: for each \( \theta \) find correspondences between peaks in the Hough Space.

+ insensitive to noise, broken edges, and occlusion.

+ natural and powerful way to apply the Constraint of Figural Continuity

- Complexity of the algorithm is \( O(MN) \),
  where \( M \) and \( N \) are the numbers of lines in the left and right images.
Fig. 2 Dynamic Programming for Line Matching in Hough Space. (a) Lines in the image ($x$-$y$) space and their corresponding peak points in the Hough ($\rho$-$\theta$) space; (b) The search plane for matching lines in (a).
(1) **Maximize the number of points matched.** The belief is that, usually, less points can be put in pairs if erroneous correspondence is made among the contending lines. Since the horizontal and vertical segments on a path in the search plane imply a no-match, their cost is set to be equal to the length of the line being skipped. This could be costly, in other words, the skipping of long lines is discouraged. The diagonal segment represents a match between a pair of lines. As shown in Fig. 1 there might not be a perfect point-to-point match between the designated matching lines. Let $l_1$ and $l_2$ be the lengths of the lines $L_l$ and $L_r$, $m$ be the number of the points matched on each line, the cost assigned to the diagonal line segment in the search space is $l_1 + l_2 - 2m$.

\[
\begin{array}{cccccccc}
0 & 3 & 14 & 27 & 45 & 58 & 66 \\
12 & 15 & 4 & 17 & 35 & 48 & 56 \\
26 & 29 & 18 & 5 & 23 & 36 & 44 \\
47 & 50 & 39 & 26 & 10 & 23 & 31 \\
59 & 62 & 51 & 38 & 22 & 15 & 23 \\
67 & 70 & 59 & 46 & 30 & 23 & 19 \\
74 & 77 & 66 & 53 & 37 & 30 & 26 \\
\end{array}
\]

**Fig. 3** An Optimal Path That Maximizes the Number of Points Matched

Fig. 3 shows an example of an optimal path generated by our program with a cost function that maximizes the number of points matched between 6 left lines and 6 right lines. These line all have approximately the same $\theta$. The lengths for the 6 left lines are 3, 11, 13, 18, 13, and 8. The lengths for the right lines are 12, 14, 21, 12, 8, and 7. The number in $(0, j)$ indicates the total cost for skipping the first $j$ left lines. The number in $(i, 0)$ indicates the total cost for skipping the first $i$ right lines. In general, all the numbers indicate the optimal cost $C$ for getting to the node.
3. Stereo Matching using Graph Cuts

- As many other computer vision tasks, Stereo Matching can be turned into an Energy Minimization problem where the matching cost is minimized and some constraints can be enforced.

- Graph Cuts [Boykov et al, PAMI 2001] is used to find approximation to the minimum via min-cut (max-flow) in polynomial time.

- Initially, a graph is constructed by connecting nodes (pixels) to all of their possible disparities. The neighboring nodes and their support are also taken into account.

- Two possible cuts are \( \alpha-\beta \) swap and \( \alpha \)-expansion.

Stereo as Energy Minimization

- Matching cost formulated as energy
  - “data” term penalizing bad matches
    \[
    D(x, y, d) = |I(x, y) - J(x + d, y)|
    \]
  - “neighborhood term” encouraging spatial smoothness (continuity; disparity gradient)
    \[
    V(d_1, d_2) = \text{cost of adjacent pixels with labels } d_1 \text{ and } d_2
    \]
    \[
    = |d_1 - d_2| \quad \text{(or something similar)}
    \]

\[
E = \sum_{(x, y)} D(x, y, d_{x,y}) + \sum_{\text{neighbors}(x_1, y_1, x_2, y_2)} V(d_{x_1, y_1, x_2, y_2})
\]
Stereo as a Graph Problem [Boykov, 2001]

Graph Definition

- Initial state
  - Each pixel connected to it’s immediate neighbors
  - Each disparity label connected to all of the pixels
Max-flow and Min-cut

A flow network is a directed graph.
Below we use $f/c$ to indicate flow/capacity of each edge. $S$ - source $T$ - sink

Initially, no flow:

Some flow, but non-max

Max-flow = 4

A cut will separate S from T.

Theorem: The minimum cost of the cut (e.g. show in ---)  
Min-cut = Max-flow.
Stereo Matching by Graph Cuts

• Graph Cut
  – Delete enough edges so that
    • each pixel is (transitively) connected to exactly one label node
  – Cost of a cut: sum of deleted edge weights
  – Finding min cost cut equivalent to finding global minimum of the energy function

Graph Cuts

• Graph \( G=(V,E) \)
• Edge weight \( w: E \rightarrow \mathbb{R}^+ \)
• Cost(\( C \)) = \( \sum_{\text{edges in } C} w(\text{edge}) \)
• Problem: find min Cost cut

• Solved in polynomial time w/ min-cut/max-flow
• Boykov and Kolmogorov algorithm
  – runs in seconds
**Difficulties**

- Parameter selection

\[ E(d) = \sum_{p \in P} M(d_p) + \lambda \sum_{\{p,q\} \in N} \delta(d_p \neq d_q) \]

- Running time: from 34 to 86 seconds

**Computing a Multi-way Cut**

- With two labels: classical min-cut problem
  - Solvable by standard network flow algorithms
    - polynomial time in theory, nearly linear in practice
- More than 2 labels: NP-hard [Dahlhaus et al., STOC ‘92]
  - But efficient approximation algorithms exist
    - Within a factor of 2 of optimal
    - Computes local minimum in a strong sense
      - even very large moves will not improve the energy
- Basic idea
  - reduce to a series of 2-way-cut sub-problems, using one of:
    - swap move: pixels with label L1 can change to L2, and vice-versa
    - expansion move: any pixel can change it’s label to L1
Moves

α-β swap and α-expansion algorithms
\(\alpha\)-expansion:

After this cut, \(p\) is assigned label \(\alpha\), \(q\) is assigned label \(\bar{\alpha}\). The cost of this cut is

\[C(p, \alpha) + C(q, \xi) + d(\xi, \alpha)\]

\(\alpha\)-\(\beta\) swap:

After this cut, \(p\) is assigned label \(\alpha\), \(q\) is assigned label \(\beta\). The cost of this cut is

\[C(p, \alpha) + C(q, \beta) + d(\beta, \alpha)\]
Applying to Stereo Matching

- $d_1$, $d_2$, $d_3$ are possible disparity values. Initially, nodes $p$, $q$, … can be assigned more than one $d$ value.

Stereo Matching Results

- Late 90’s state of the art: 5.23% errors
- Recent state of the art: 1.29% errors
4. Multiple baseline stereo


- In stereo matching, *sum square difference* (SSD) can be used to measure the quality of match, where minimal SSDs are expected for good matches.

- *Multiple baseline stereo* is proposed to reduce the difficulty in finding good matches. Cameras are placed at multiple positions with baseline separations at \( b, 2b, \ldots i \cdot b, \ldots \)

---

**Multiple baseline stereo algorithm**

- In Fig. 2(a), \( f(x) \) is a possible feature profile with the 1st possible match (peak) at \( x = 0 \).

- Figs. 2(b) and (c) show SSD versus \( d \) (disparity). In each, all possible minima are shifted by the same amount (proportional to \( i \cdot b \)) along the \( d \)-axis.

- The next two rows show that if we use the *inverse distance* \( \zeta = d / i \), then the 1st valleys (minima) will align.

- A simple algorithm for deriving a good overall minimum is to sum up the SSDs from the multiple baseline stereo pairs (Fig. 2(f)).
Combining multiple baseline stereo pairs

Evaluation of Stereo Algorithms


and more recently, CVPR 2006.
Active Stereo with Structured Light

- Project “structured” light patterns onto the object
  - simplifies the correspondence problem

Laser Scanning

- Optical triangulation
  - Project a single stripe of laser light
  - Scan it across the surface of the object
  - This is a very precise version of structured light scanning