L22. A trip through the sensor zoo
Stereo

- Classic approach to stereo vision: matching pixel patches between left and right.

- Shortcoming: in low-detail areas, results are erratic. (How would we enforce local consistency?)

Figure 3. Half-pel depth map on Renault Automobile Part using 16x16 block matching.

Block size has a significant impact on the quality of the depth map. Larger blocks are more immune to noise and other differences (such as specular reflections and occlusions) that can occur between the left and right images. However, larger blocks cannot capture the fine depth detail that smaller blocks can. Figure 4 demonstrates a view of the Pentagon with large and small blocks. The larger blocks provide cleaner data (fewer discontinuities), but many of the details are missing.

Figure 4. Comparison of depth maps created with different block sizes.

Conclusion

Perceiving depth from multiple source images is a computationally expensive process. The block-matching algorithm used here has generally good performance, and the confidence estimate can be used to effectively mask out regions of high noise. Selection of block size is a trade-off between depth-map noise and detail.

While only simple block-matching approaches were outlined here, depth perception remains a much-researched topic.

MatLab source code and raw images can be obtained from: http://www.ravenousbirds.com/eolson/6.344
Block Matching

- Exploit epipolar geometry
  - A pixel in the left camera corresponds to a ray.
  - The image of a ray (in the right camera) is a line.
  - Thus, if we know the geometry of the cameras, we only need to search for matches along a line.

- Matching procedure
  - Block size (5x5, 7x7, ...)
  - Comparison (SAD, SSE)

- Sub-pixel matching
  - Fractional translation of reference image
  - Polynomial interpolation of full-pixel data

$$\sum_{(i,j) \in W} |I_1(i,j) - I_2(x + i, y + j)|$$
$$\sum_{(i,j) \in W} (I_1(i,j) - I_2(x + i, y + j))^2$$

Fit parabola through >3 points approximately

Bernd Girod: EE398A Image and Video Compression
Stereo Vision: Graphical Model

- Label each pixel with a disparity
  - Maximize agreement between adjacent pixels ("discontinuity cost")
  - Maximize agreement between left and right pixel ("data cost")
Stereo MRFs

- Could approach as a least-squares problem
  - State: disparity at each node (relax to continuous values)
  - Optimize product of function potentials (or equiv. sum of log of potentials... “log likelihood”)

- Very difficult local minimum
  - Least-squares solves a local quadratic problem. If you’re not in the right basin, you won’t converge.
  - Least squares doesn’t work well.
Iterated Conditional Modes

- Simple idea:
  - Consider a single node at a time. (i.e., fix the values of all other nodes)
  - Compute a new disparity for that node that minimizes the log likelihood
    - Only a function of the neighboring factor potentials... cheap!
    - Always reduces global error
- Not much better than least squares--- still get stuck in local minima.
- Need a method that can “look ahead”, leaping out of local minima
  - Consider two nodes a=0, b=0. Cost f(a,b) has local minimum at 0,0, but global minimum at 1,1.
Loopy Belief Propagation

- Each node passes messages to its neighbors:
  - “If you take on value \( v \), the cost could be as low as \( m(v) \).”
  - All possible values of \( v \) are evaluated in a best-case sense, allowing the recipient to “teleport” to a new minimum.

\[
m_{t_{p ightarrow q}}(f_q) = \min_{f_p} \left( V(f_p, f_q) + D(f_p) + \sum_{s \in N(p) \setminus q} m_{t_{s ightarrow p}}^{-1}(f_p) \right)
\]
Isn’t this fun?

• With an almost trivial model, we can destroy block matching problems.
  ▶ You can be competitive with Middlebury top 100 in a couple days’ effort!

SSD+min-filter [scharstein szeliski], rank = 90
LBP [olson], rank* = 60
The disappointment

• MRF approaches are too slow for robots
  ▶ #1. [Wang/Zheng]: 20s
  ▶ #2. [Yang/Nister]: 62s

• Block matching is fast!
  ▶ (unranked) [Konolige], 10ms
Why is LBP slow?

- Short answer: because computing messages is slow

\[ m_{p \rightarrow q}^t(f_q) = \min_{f_p} \left( V(f_p, f_q) + D(f_p) + \sum_{s \in N(p) \setminus q} m_{s \rightarrow p}^{t-1}(f_p) \right) \]

P’s message to q specifies a “cost” for each value that q might take.

Cost of assigning fp to pixel p

Cost of assigning fp to pixel p

For every value fq, we’ll report the cost for the best-case fp.

How much do our neighbors say it would cost for q to have value p

for x=1:width
for y=1:height
for n=1:neighbors
for fq=1:labels
for fp=1:labels
...
Cool trick #1: Min Convolution

\[ m_{p\rightarrow q}^t(f_q) = \min_{f_p} \left( V(f_p, f_q) + D(f_p) + \sum_{s \in N(p) \setminus q} m_{s \rightarrow p}^{t-1}(f_p) \right) \]

\[ m_{p \rightarrow q}(f_q) \]
Cool trick #1: Min Convolution

\[ m_{p \rightarrow q}^t(f_q) = \min_{f_p} \left( V(f_p, f_q) + D(f_p) + \sum_{s \in N(p) \setminus q} m_{s \rightarrow p}^{t-1}(f_p) \right) \]
Cool trick #1: Min Convolution

\[ m_{p \rightarrow q}^t(f_q) = \min_{f_p} \left( V(f_p, f_q) + D(f_p) + \sum_{s \in N(p) \backslash q} m_{s \rightarrow p}^{t-1}(f_p) \right) \]
Cool trick #1: Min Convolution

\[ m_{p \rightarrow q}^t(f_q) = \min_{f_p} \left( V(f_p, f_q) + D(f_p) + \sum_{s \in N(p) \setminus q} m_{s \rightarrow p}^{t-1}(f_p) \right) \]
Cool trick #1: Min Convolution

- This is a min-convolution operation
  - Naive implementation is $O(k^2)$
- Efficient algorithms exist for special cases!
  - In linear case, forward-backwards algorithm $O(k)$
  - Quadratic case also has a method... a bit messier, but still $O(k)$
- Exact!

```
m_{p\rightarrow q}(f_q)
```

(fp=1) (fp=2) (fp=3)
Performance

tsukuba (384 x 288), 16x subpixel, nlabels=256
Cool Trick #2: Multi-Grid

[Felzenszwalb/Huttenlocher 2004]

- Advantages:
  - Information spreads rapidly around graph

- Disadvantages:
  - Have to come up with function potentials for other levels of the image pyramid
  - Can lead to artifacts due to the arbitrary alignment of the grid cells
Multi-resolution LBP
Cool Trick #3: Quantized labels

- Idea: Start iterating with fewer labels, slowly increase number of labels

- Advantages:
  - Information spreads rapidly around graph
  - No spatial blocking artifacts

- Disadvantage:
  - Not as fast as multi-grid

[Strom, Olson 2010*]
Quantized LBP
PrimeSense/Kinect

- Similar to a stereo camera in concept
  - But replace one camera with a *projector*
  - Second camera detects projected camera.

- Why is this a good idea?
  - It works even when the environment is devoid of distinguishing features (e.g. white walls)
  - Under favorable conditions, very good results

- What are the shortcomings?
  - Brightness of projector limits effectiveness at long ranges and outdoors
  - Power consumption / stealth
Diffraction grating

IR Laser

RGB Camera

IR bandpass filter

Depth Camera

Disparity Matching

Registration

Speckle pattern projected onto world. Different for every sensor!

In focus at all ranges due to coherent light source.

Reliable data on how Kinect works hard to find. Sources:
-- libfreenect
-- www.ros.org/wiki/kinect_calibratin/technical (good!)
-- iFixit teardown and PrimeSense Bill of Materials
-- PrimeSense patent filings (ugh)
Kinect Particulars

- Produces 640x480 RGBD Image
  - IR Camera is 1280x1024 @ 15Hz
    - Uses 2x2 binning to increase sensitivity and frame rate to 30Hz
    - Monochrome... 16 bit?

- Matching
  - Calibration image stored in device at factory
  - Repeatedly “streamed” in sync with acquired IR image, fed into matching engine
  - Block based matching
    - 9x9 blocks
    - 1/8 pixel interpolation
    - 64 (?) pixel search range  (Kinect returns 11 bit range values)

- Registration
  - Corrects for parallax of RGB and depth sensor. (Could be eliminated by using a single sensor with both RGB and IR pixels in an RGBI “Bayer” pattern).