A Brief Introduction to Stereo Vision

Ron Parr

CPS 1/296

Stereo

- Stereo attempts to match pixels in one frame with pixels in the other frame
- Matches are based pixel luminance and (optionally), color, other heuristic features

"Teddy" images from Middlebury Stereo Vision page.

Depth from Stereo

- Given a matching, depth is trivially estimated from camera geometry

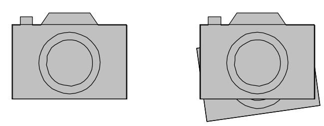
$$Z = f\frac{B}{d}$$

- f = focal length
- B = baseline (distance between camers)
- d = disparity (convert pixels to distance)

- See derivation on board
- Geometry is trivial; establishing correspondence is hard

What is Calibration?

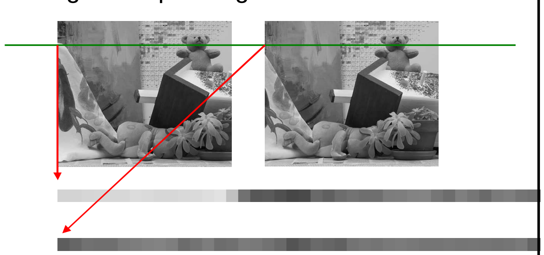
- Put image sensors in the same plane, same rotation, horizontal offset, etc.



- Or rectify with software,
- Or figure out exact relative position of cameras and compensate on the fly

Role of Calibration

- Calibration allows restriction of search along corresponding rows



Very Basic Stereo Assumptions

- Define a cost function over matchings
  - Squared luminance difference
  - Optional smoothness /coarseness measures
- Allow some pixels to be unmatched (occluded), but regularize with "occlusion penalty"
- Introduce simple constraints
  - Uniqueness
  - Ordering
- Assume independence between rows

Very Basic Stereo Algorithms

- Match pixels from left image to right (WLOG)
- $X_i \in \{0\ldots d, \Omega\}$ is the disparity in right frame for pixel i in left frame
  - If $X_i = k$, $X_{i+1} \in \{0\ldots k, \Omega\}$
  - If $X_i = \Omega$, $X_{i+1} \in \{0\ldots X_{i-1}+1, \Omega\}$
- $C(X_i=k)$ = f(luma(i, left), luma(i-k, right))
- $C(X_i=\Omega)$= "occlusion penalty"=S
- Minimize: $\sum_i C(X_i)$
- Subject to constraints

Dynamic Programming for Stereo

- Want to minimize: $\sum_i C(X_i)$
- Over all assignments to all pixels
- Is it necessary to consider all possible sequences of choices?

Dynamic Programming: Main Idea

- Suppose we have the lowest cost matching that ends with disparity level d at pixel i
- Do we every need to reconsider other ways of reaching disparity level d at pixel i as we move forward to pixels j>i?
- No!

Real World Example

- Suppose you want to go to NY via Washington DC
- If you have an optimal plan to go from Durham to Washington, then you don't need to revise this plan as you plan your trip from Washington to DC

Getting back to stereo

- Suppose you have an optimal (lowest cost) matching that ends with disparity level d in pixel i (= solution 1)
- Assume that you later find an optimal total solution (= solution 2) that assigns disparity level d to pixel i, but differs from solution 1 for some pixels <= i.
- Decompose solution 2 into two parts (2a, 2b), where 2a is the half up to pixel i.
- Assume (for contradiction) that (1, 2b) has cost higher than (2a, 2b)
- However, since cost is additive and solution 1 is optimal up to i, (1, 2b) must have cost <= (2a, 2b)

Implementing it

- Not hard, but tricky.
- Three cases
- Continue at current disparity level:
  - $best(x_i,d) = best(x_{i-1},d) + c(x_i=d)$
- Skip k pixels in the left frame:
  - $best(x_i,d) = \min(best(x_i,d),\min_{k<d} best(x_{i-1},d-k)+kS+c(d-k))$
- Skip the current pixel in the right frame:
  - $best(x_i,d) = \min(best(x_i,d), best(x_{i-1},d+1)+S)$

Issues

- Computational complexity
  - Good compared to alternatives
  - Still slow for large images
  - Can be improved slightly with clever formulation (Bobick & Intille)
- Boundary conditions/parameters
  - Max disparity level
  - Starting disparity level (edge penalties?)
  - Occlusion penalty
- Assumption of independence between rows
  - Oversimplified?
  - Tends to cause streaking

More Advanced Approaches

- More advanced approaches typically use a more complicated cost function
- Pros:
  - Permits encoding of more background knowledge into optimization
  - Produces better results in most cases
- Cons:
  - Hard to justify the numbers used
  - Slow- can’t use dynamic programming
  - Problem is still underdetermined