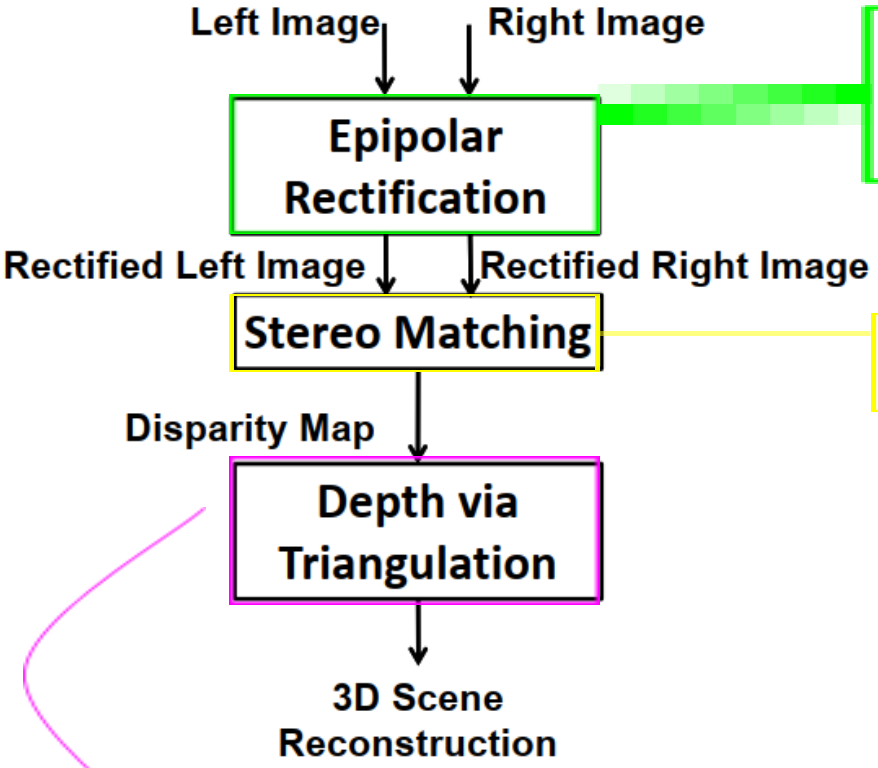
Stereo Vision Exam

Questions Vowi 2019

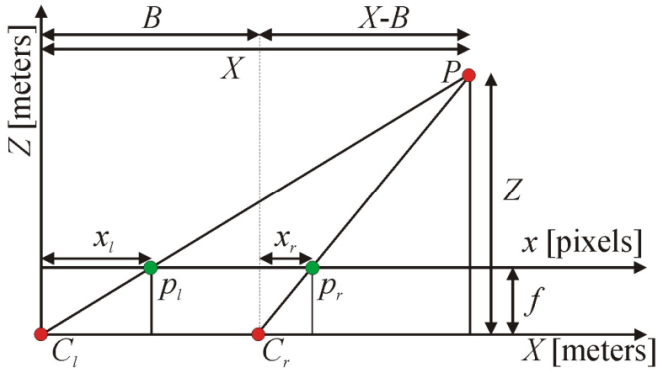
1) Stereo Pipeline



In rectified images epipolar lines coincide with horizontal scan lines.
Achieved by projecting images onto virtual cameras

Generating a disparity map of 2 2D images

Depth via Triangulation



<u>Known from Camera Calibration:</u>
Z[meters] = Z-value in 3D space
x[pixels] = x-coordinate in 2D space
X[meters] = x-coordinate in 3D space
B = Baseline (Distance between focal point and cameras)
C_1 C_2 = Focal Points
f = focal length

<u>Known from stereo matching:</u>
x_l x_r = x-coordinates of corresponding points in each picture

<u>Unknown:</u>
X = x-coordinate in 3D space
Z = z-coordinate in 3D space (depth)

How to find Z?
$x_l - x_r$ = disparity d (displacement of single pixel in two images)

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

2) Which dataset is used for evaluating stereo matching algorithms? Explain

Middlebury Set
Ground truth images for stereo vision
Generated either through hand labelling or structured light (most used in middlebury set)

Middlebury Benchmark

1. Build absolute difference between computed and ground truth (Middlebury) disparity maps
2. if disparity difference > 1 pixel → pixel is counted as error

Error metrics:

1. Percentage of erroneous pixels in **unoccluded regions**
2. Percentage of erroneous pixels in **the whole image**
3. Percentage of erroneous pixels in **regions close to disparity borders**

3) Explain local stereo

Local methods for smoothness assumption application
implicit smoothness assumption
(Global methods use explicit form → smoothness term)

Naive algorithm

1. compare color of each pixel p in left image with color of each pixel on same horizontal scan line in right image
2. select pixel with most similar color as match

Window-Based Matching

1. center small window on a pixel
2. match whole window in right image

$$d_p = \arg\min_{0 \le d \le d\,\max} \sum_{q \in Wp} c(q, q-d)$$

d_{max} = max disparity
Wp = pixels in window
c(p,q) = color difference

dp = minimum disparity between all pixels in the window in the left and right image

$\sum_{q \in Wp} c(q, q-d)$ → Aggregated Matching Scores

Problems:

Results depend very strongly on window size
Large window good with:

- untextured regions
- aperture problems
- repetitive patterns

Small window good with:

- foreground fattening effect

Untextured regions = color variation inside window necessary
Aperture = texture with vertical orientation necessary
Repetitive Patterns = certain amount of non-repetitive texture needed

Run time depends on window size = very slow

Solution for runtime - **Sliding window technique:**
removes dependency on window size
enables real-time implementation

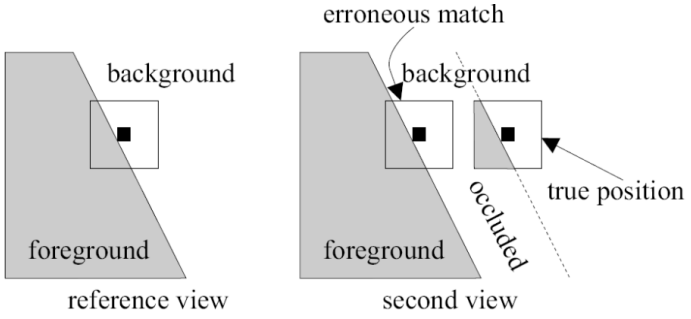
$W_{x,y}$ $W_{x+1,y}$

c1	c2	c3	c4	c5

c2	c3	c4	c5	c6

Using the computed aggregated matching costs form the window before (Ax,y)
Ax+1,y = Ax - c1 + c6

Foreground Fattening Problem
Foreground objects are enlarged because
window matching applies implicit smoothness assumption:
“All pixels within the window are assumed to have the same disparity”
leads to systematic error in regions close to disparity discontinuities → falsely assigned to foreground disparity



<u>Adaptive Windows</u>

Combines advantages of small and large windows
Good window size individually for each pixel

Option 1 (Fusiello):
Center 9 windows at each pixel
Take window with minimum aggregated costs
Assumption: at least one window will not overlap a disparity discontinuity
very efficient
Option 2 (Hirschmueller):
Divide Window into 9 sub-windows
aggregate costs over 5 best sub-windows (computer matching score)
only moderate quality

4) Explain Adaptive Support Weight Approaches

Local Stereo Algorithm

Compute **A**ggregate Matching Costs of each pixel **p** at disparity **d**

$$A_{p,d} = \sum_{q \in W_p} c(q, q-d)$$

W_p = all pixels in window of p

Foreground fattening because all pixels in window have the same influence on costs → formula needs to be modified

$$A_{p,d} = \sum_{q \in W_p} w(p,q) \bullet c(q, q-d)$$

multiply color difference with weight function w(p,q)
determines likelihood of p and q laying on same disparity

Ideally:

$$w(p,q) = \begin{cases} 1 & \text{if } p \text{ and } q \text{ lie on the same disparity} \\ 0 & \text{otherwise} \end{cases}$$

Now only pixels that lie on the same disparity influence aggregated matching costs

<u>Weight functions:</u>
Using monocular cues (color, spatial distance)

- Yoon
  + Good quality results
  - High computational costs (sliding window does not work here)
- Hosni
  includes color edges
  Weights proportional to the amount of connectivity

Pixels are connected if path with constant color between them
Better performance than Yoon

5) Explain some smoothness term variants

Smoothness assumption: neighbouring pixels should be assigned to the same disparities

Smoothness Term

$$E_{smooth}(D) = \sum_{<p,q>\in N} s(d_p, d_q)$$

N = spatial neighbouring pixels in left image
s() = smoothness function → imposes penalty if disparities are different

Potts Model

$$s(d_p, d_q) = \begin{cases} 0 & \text{if } dp = dq \\ P & \text{otherwise} \end{cases}$$

P = Penalty (user defined)

Linear Smoothness function
absolute disparity difference
special case → global optimal solution can be computed via graph cuts

for each pixel add d_{max} + 1 nodes, add edges between consecutive nodes
edge weights → according to pixel dissimilarities
add smoothness edges between pixels

A cut that assigns p to d_p and q to d_q includes $|d_p - d_q|$ smoothness edges of costs P
Problem: the costs for max small jumps (wrong) are not higher than for one big jump (correct) in disparity → No favoring of the correct solution

Blurry disparity discontinuities
non-discontinuity preserving
poor choice for stereo

Potts Model

$$s(d_p, d_q) = \begin{cases} 0 & \text{if } dp = dq \\ P & \text{otherwise} \end{cases}$$

np complete problem
costs for many small jumps are significantly higher → Favoring of the correct solution

performs poorly in reconstruction of slanted surfaces

Modified Potts Model

$$s(p,q) = \begin{cases} 0 & \text{if } dp = dq \\ P_1 & \text{if } |dp - dq| = 1 \\ P_2 & \text{if } |dp - dq| > 1 \end{cases}$$

P_1 P_2 = user defined penalties
P_1 = penalty for small jumps at slightly slanted surfaces
P_2 = penalty for large jumps at disparity discontinuities

Correct solution might be taken
gives reasonable results in practice

Truncated Linear Model
Same idea as modified Potts Model

$$s(p,q) = \min(|d_p - d_q|, k)P$$

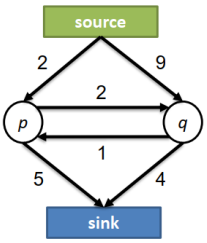
k = truncation value (user defined)

small disparity variations derive small penalty
disparity discontinuities derive large penalty

currently state of the art

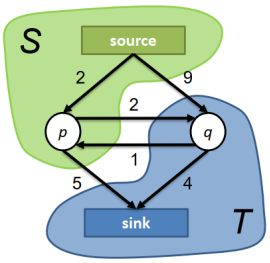
6) Linear Smoothness Term as Graph (Min-Cut)

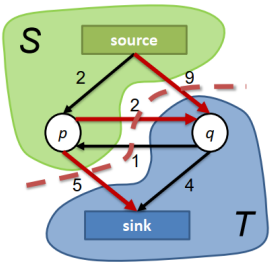
Min Cut Problem



Two dedicated nodes: source and sink

Partition graph into two sets →

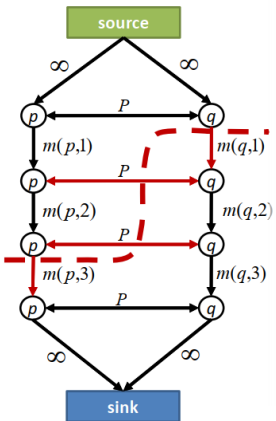




Cut consist of all edges from S to T

Costs of a cut are the sum of the weights of the edges → 5+2+9 = 16
Wanted: Cut with minimum costs

Global Optimum of the Linear Smoothness Function:



This cut assigns *p* to disparity 3 and *q* to disparity 1.

The costs are:

m(p,*3*)+*m*(*q*,1)+2*P*

(which represents our energy with a linear smoothness function)

Add d_{max} + 1 nodes (here 3 + 1) from source to sink

7) Mutual information

Is a radiometric insensitive measure
Radiometric distortion = intensity is different in left and right image

+ is a pixel based measure
+ artifacts at disparity borders are avoided
- Global Model (models only radiometric changes that are valid in whole image)

to compute mutual information matching scores we need the disparity map
solved iteratively
- compute initial disparity map
- mutual information matching scores
- compute new disparity map using mutual information
- repeat

How are mutual information matching scores computed?
for each pixel p we look up its matching point q
look up intensity values
create diagram P and makes entries at I_p,I_q for each possible pair of intensity values
intensity pairs that occur frequently are given less matching costs
for diagram P compute -log(P) → **Mutual Information Scores**

8) Sampling Insensitive Measure

samples are taken at different curve positions in left and right image
therefore corresponding pixels have different intensity values

Solution: look at horizontal neighbours and interpolate pixel between samples

$$m(p,p') = \min(|p-p'|, |p_- - p'|, |p_+ - p'|)$$

should be done in a symmetric

+ includes correctly sampled pixels
+ low intensity dissimilarity
+ high chances for correct match

Further Summary

1) Stereo Matching

Challenges:

- Color Inconsistencies
- Untextured Regions
- Occlusion Problem (consequence of discontinuities in depth)

Assumptions:

- Photo consistency assumption: corresponding pixels have the same intensity/color
- Epipolar assumption: matching point lies on the same horizontal scanline
- Smoothness assumption: spatially close pixels have the same disparity
- Ordering assumption: order of pixels is preserved in both images
- Disparity Gradient Limit

Uniqueness Constraint: A pixel in one picture has <= 1 matching point in the other picture
Not true for transparent objects and slanted surfaces

2) Slanted Surfaces/ Plane Sweeping

For surfaces that are not fronto-parallel to the camera

Plane sweeping: Precompute all 3D planes present in the scene → project window onto the planes

3) Occlusion Handling

Left-Right Consistency Check:
compute 2 disparity maps, once with left and once with right image as reference
check matching points in left and right disparity map for every pixel
if it is the same, the disparity is correct, if not it is invalid
fails for: occluded pixels, mismatched pixels

Occlusion Filling:
assign occluded pixels to a disparity → occluded pixels have depth of background
find the first valid pixel left and right of occluded pixel
assign disparity as minimum of disparity of those pixels
Problem: generates horizontal streaks
Solution: apply smoothing median filter

4) Energy Minimization Problem

Defining an energy/cost function to measure the quality of a disparity map
High Energy = BAD
Low Energy = GOOD

Energy function:

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

D = Disparity of left image
E_{data} = measures photo consistency
E_{smooth} = measures smoothness

Data Term: Measures color dissimilarity for each pixel in the left image

Other assumptions can be modelled into the energy function as well

- infinite costs if uniqueness assumption is violated
- Energy is lower if disparity borders coincide with intensity edges

Limitations:
Reasons for poor performance

- Energy modelling (poor representation of the problem)
- Energy minimization (far off from exact minimum)

5) Dynamic Programming:

Special case of energy function
smoothness interactions form a tree

$$L(r) = \min_{d \in D} \left(m(r,d) + \sum_{c \in C_r} E(c,d) \right)$$

$$E(p,d') = \min_{d \in D} \left(m(p,d) + s(d,d') + \sum_{c \in C_p} E(c,d) \right)$$

L(r) = computes exact energy minimum
r = root of the tree
D = all allowed disparities
m(p,d) = costs of matching pixel p at disparity d
s(d,d') = smoothness function
Cp = all pixels that have p as direct predecessor on the path to the root node

Find disparity sequence that led to the optimum by backtracking

+ Dynamic Programming algorithms are very fast
+ Good tradeoff between speed and accuracy
- Can only be applied to tree structures
- Erasing smoothness edges leads to performance degradations

<u>Obtaining Tree Structure:</u>
Disparity discontinuities are typically aligned with intensity edges
Therefore smoothness edges between neighbouring pixels of different intensities are the least important ones → remove those

Compute weight for each smoothness edge between two pixels

$$w(p,q) = | I(p) - I(q) |$$

Build minimum spanning tree using those weights

6) Graph Cuts

Powerful optimization method
Finds strong local minima
energy function is np complete

Input: labeled image (labeled with disparity values)
Change labels with move operations

Move operations:

- ***αβ***-swap
- ***α***-expansion
- fusion move

αβ-swap:
labels ***α***,***β***
a pixel assigned to ***α*** can either switch to ***β*** or keep old label
a pixel assigned to ***β*** can either switch to ***α*** or keep old label

α-expanison
labels ***α***,***β***
α selected
any pixel can switch label to ***α*** or keep old label
typically outperform ***αβ***-swap
Problem: large number of possible ***α***-expansions
Find “best” ***α*** expansion (= largest decrease of energy)

Algorithm:

- Start with an arbitrary labeling *f*.
- Loop (e.g. 3 times)
  - For each allowed label α:
    - Find $f^* = \text{argmin } E(f')$ among f' within one α-expansion of f
    - $f := f^*$

Computing with Graph Cuts:

1. Energy as pseudo-boolean function
2. construct graph that represents that function
3. compute minimum cut in this graph

7) Belief Propagation

optimization algorithm that minimizes energy
can only find approximate solution
Pixel in 4 connected grid communicate with neighbours
each pixel has belief about its disparity assignment
sends belief to its neighbors (which assignment they should have)
gathers incoming messages and updates its belief accordingly
Process is iterated
will not converge

optimal solution if graph is tree