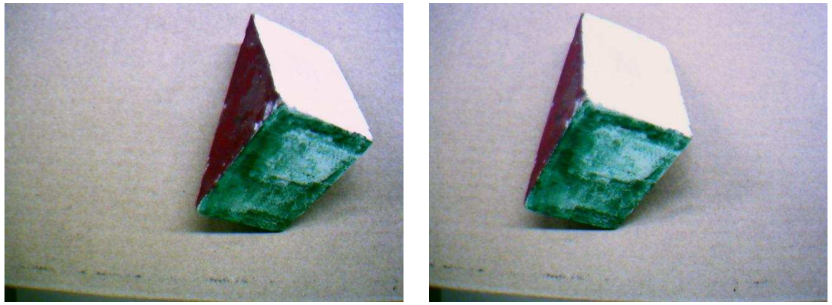
System 5 Introduction

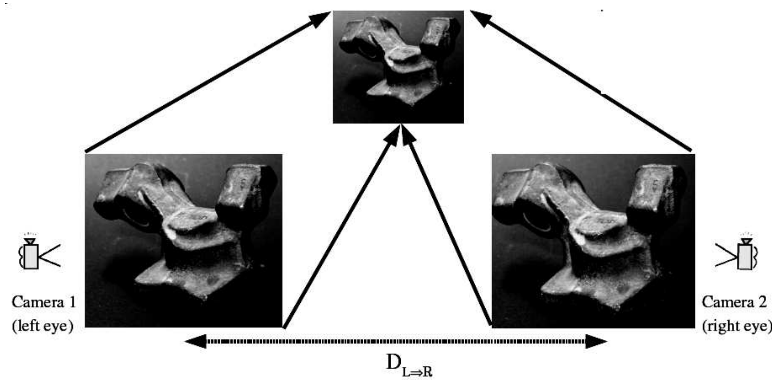
Is there a Wedge in this 3D scene?



Data a stereo pair of images!

3D part recognition using geometric stereo

Binocular Stereo Vision



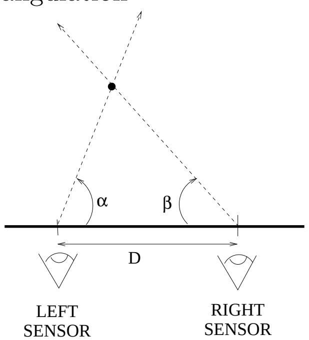
Given two 2D images of an object, how can we reconstruct 3D awareness of it?

Binocular Stereo

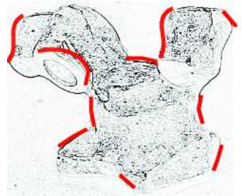
Goal: build 3D scene description (eg. depth) given two 2D image descriptions

Useful for: obstacle avoidance, grasping, object location

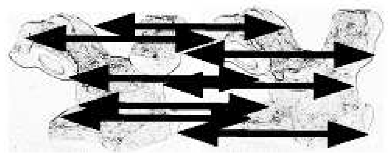
Key principle: triangulation



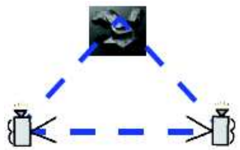
Stereo vision - a solution



1) **Feature extraction**



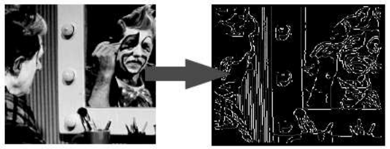
2) **Feature matching**:



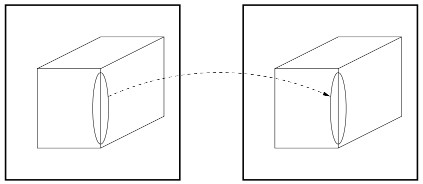
3) **Triangulation**:

Possible image features

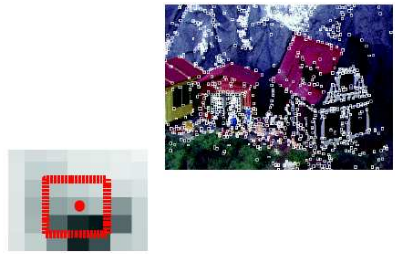
1) **Edge fragments**



2) **Edge structures (eg vertical indoor lines)**



3) **General interest points**



Larger easier to match but harder to get and less dependable
Human visual system thought to work at edge fragment level

System 5 Overview

1. **Feature extraction**:
   Canny edge detector
   RANSAC straight line finding
   SIFT point features
2. **Feature matching**:
   Stereo correspondence matching lines
   SIFT points
3. **Triangulation**:
   3D line feature position estimation
4. **3D Object recognition**:
   3D geometric model
   Model-data matching
   3D pose estimation

Edge Detector Introduction

- Edge detection: find pixels at large changes in intensity
- Much historical work on this topic in computer vision (Roberts, Sobel)
- Canny edge detector first modern edge detector and still commonly used today
- Edge detection never very accurate process: image noise, areas of low contrast, a question of scale. Humans see edges where none exist.

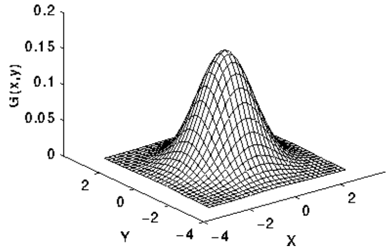
Canny Edge Detector

Four stages:

1. Gaussian smoothing: to reduce noise and smooth away small edges
2. Gradient calculation: to locate potential edge areas
3. Non-maximal suppression: to locate "best" edge positions
4. Hysteresis edge tracking: to locate reliable, but weak edges

Canny: Gaussian Smoothing

Convolve with a 2D Gaussian



$\frac{1}{273}$

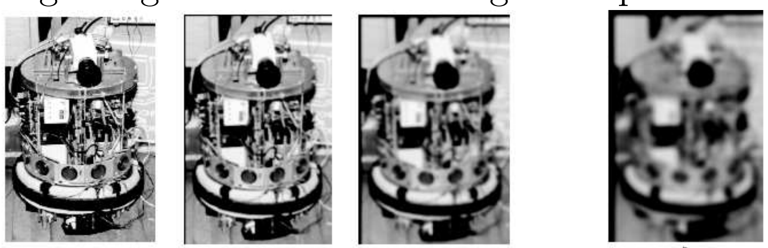
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Averages pixels with preference near center: smooths noise without too much blurring of edges

σ of Gaussian controls smoothing explicitly

$$\text{convolution } mask(r,c) = \frac{1}{2\pi\sigma} e^{-\frac{r^2+c^2}{2\sigma^2}}$$

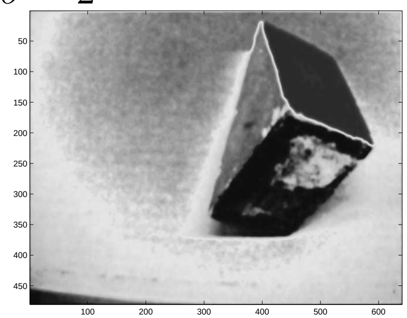
Larger σ gives more smoothing - low pass filter

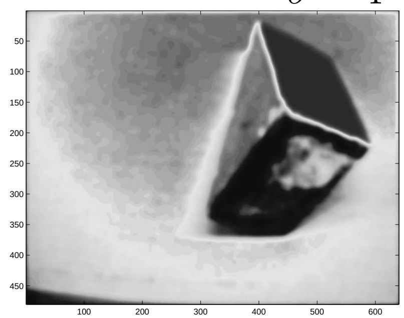


original σ=1.0 σ=2.0 σ=4.0

Gaussian Smoothing Examples

$\sigma = 2$ $\sigma = 4$





Conservative Smoothing

Gaussian smoothing inappropriate for salt&pepper/spot noise



Noisy image   Gauss smooth   Conservative

Canny: Gradient Magnitude Calculation

$G(r,c)$ is smoothed image

Compute local derivatives in the r and c directions as $G_r(r,c)$, $G_c(r,c)$:

Edge gradient: $\nabla G(r,c) = (G_r(r,c), G_c(r,c))$

Gradient magnitude:

$$H(r,c) = \sqrt{G_r(r,c)^2 + G_c(r,c)^2}$$

$$\doteq \mid G_r(r,c) \mid + \mid G_c(r,c) \mid$$

Gradient direction

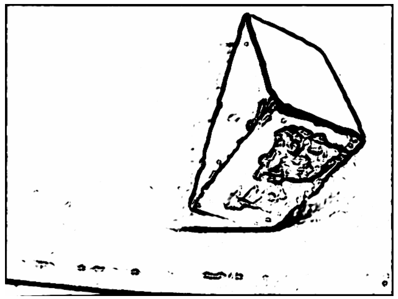
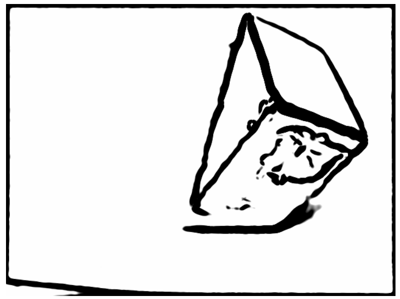
$\theta(r,c) = arctan(G_r(r,c), G_c(r,c))$

$$G_r(r,c) = \frac{\partial G}{\partial r} = lim_{h \to 0} \frac{G(r+h,c) - G(r,c)}{h}$$

$$\doteq G(r+1,c) - G(r,c)$$

Gradient Magnitude Examples

$\sigma = 2$ $\sigma = 4$

σ controls amount of smoothing
Smaller σ gives more detail & noise
Larger σ gives less detail & noise

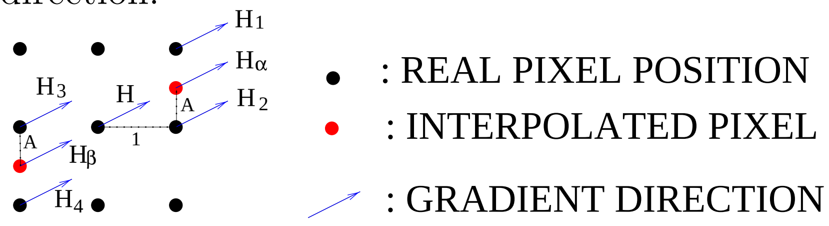
Canny: Non-maximal Suppression

Where exactly is the edge? peak of gradient

Suppress lower gradient magnitude values: need to check **ACROSS** gradient

0	0	3	12	4	0
0	0	6	10	2	0
0	2	8	7	1	0
0	3	11	4	0	0

Estimate gradient magnitudes using gradient direction:



$A = \frac{|G_r|}{|G_c|}$

$H_\alpha = AH_1 + (1-A)H_2$

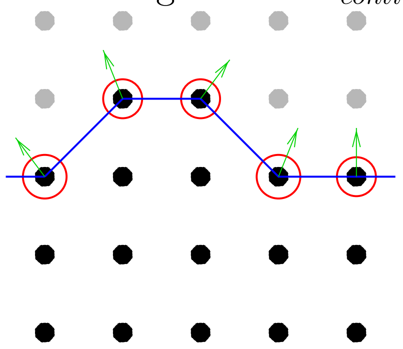
$H_\beta = AH_4 + (1-A)H_3$

Suppress (set to 0) if $H < H_\alpha$ OR $H < H_\beta$

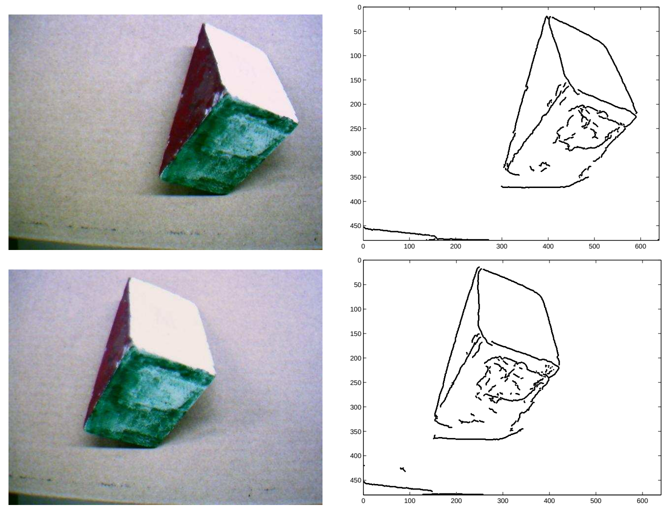
Canny: Hysteresis Tracking

Start edges at certainty: $H > \tau_{start}$

Reduce requirements at connected edges to get weaker edges: $H > \tau_{continue}$



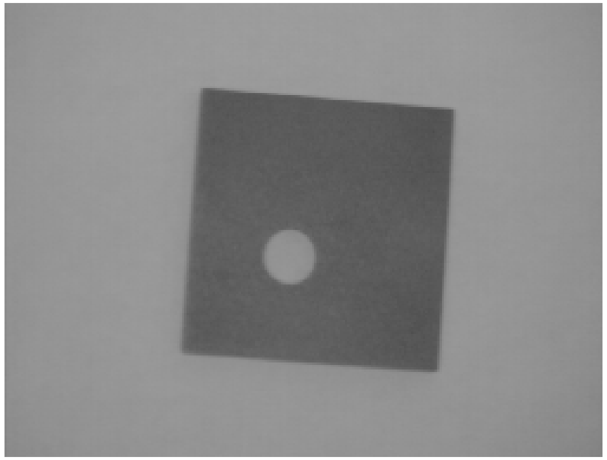
Stereo Canny Edges



Matlab has Canny: `edge(leftr,'canny',[0.08,0.2],3);`

Midlecture Problem

Where might the Canny edge detector find edges in this image?



Finding Lines from Edges

RANSAC: Random Sample and Consensus

Model-based feature detection: features based on some *a priori* model

Works even in much noise and clutter

Tunable failure rate

Assume

- Shape of feature determined by T true data points
- Hypothesized feature is valid if V data points nearby

RANSAC Pseudocode

```
for i = 1 : Trials
  Select T data points randomly
  Estimate feature parameters
  if number of nearby data points > V
    return success
  end
end
return failure
```


RANSAC Termination Limit

p_{all-f} is probability of algorithm failing to detect a feature
p_1 is probability of a data point belonging to a valid feature
p_d is probability of a data point belonging to same feature
Algorithm fails if $Trials$ consecutive failures

$$p_{all-f} = (p_{one-f})^{Trials}$$

Success if all needed T random data items are valid

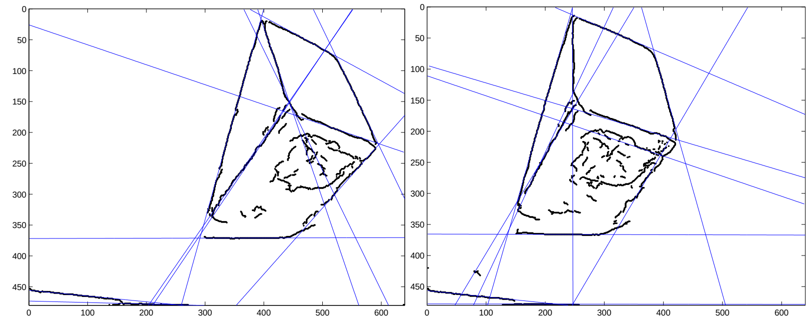
$$p_{one-f} = 1 - p_1(p_d)^{T-1}$$

Solving for expected number of trials:

$$Trials = \frac{log(p_{all-f})}{log(1 - p_1(p_d)^{T-1})}$$

RANSAC Line Detection

Line model: infinite line through 2 points ($T = 2$)
$T = 2$ edge points randomly chosen
Accept if $V = 80$ edge points within 3 pixels
$p_{all-f} = 0.001$, $p_1 = 0.1$, $p_d = 0.01$, $Trials = 688$

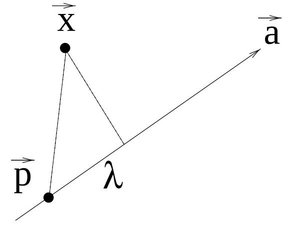


Mostly accurate lines, but don't know endpoints

Finding line segments

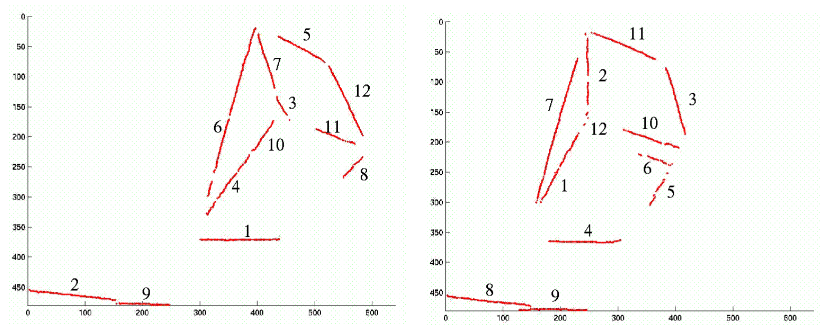
Some random data crossings
Want to find approximate observed start and end of true segment

1. Project points $\{\vec{x}_i\}$ onto ideal line thru point $\vec{p}$ with direction $\vec{a}$: $\lambda_i = (\vec{x}_i - \vec{p}) \cdot \vec{a}$. Projected point is $\vec{p} + \lambda_i \vec{a}$



2. Remove points not having 43 neighbor points within 45 pixels distance
3. Endpoints are given by smallest and largest remaining λ_i.

Found line segments

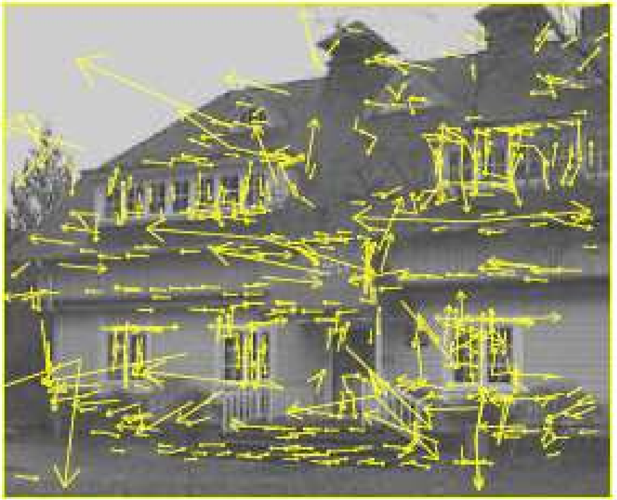


SIFT Features

SIFT: Scale Invariant Feature Transform

Image point feature + local description (128 vector)
Sparse, reasonably distinguishable points
Invariant to translation, rotation, scale, some 3D

Example feature locations



Matching Applications

Matchable features for:

- Object recognition
- Model-data alignment
- Image registration
- Stereo matching

Four Step Algorithm

1. Detect extremal points in scale space
2. Accurate keypoint localization
3. Feature orientation estimation
4. Keypoint descriptor calculation

Scale Space Smoothing

Gaussian smoothing via convolution

$$L(x,y,\sigma) = G(x,y,\sigma) \circ I(x,y)$$

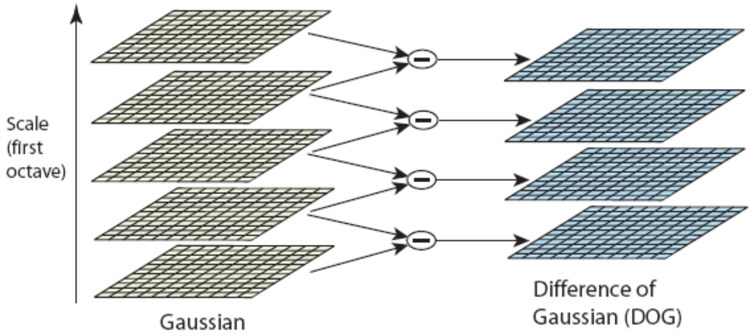
$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Difference of Gaussians:

$$D(x,y,n) = L(x,y,2^{\frac{n}{S}}) - L(x,y,2^{\frac{n-1}{S}})$$

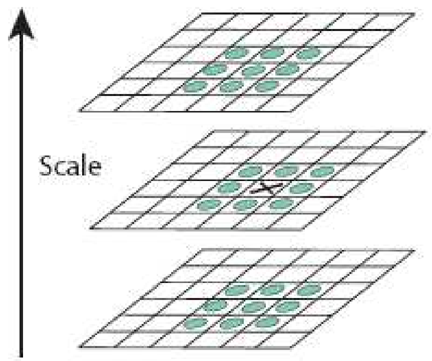
where $n = 1 \ldots N$
$S = 3$ best

Scale Space



Point Extrema

Pick extremal points larger/smaller than their 26 neighbours:



Subpixel Extrema Localization

Hessian:

$$\mathrm{H} = \begin{bmatrix} \partial^2 D/\partial x^2 & \partial^2 D/\partial x \partial y & \partial^2 D/\partial x \partial \sigma \\ \partial^2 D/\partial x \partial y & \partial^2 D/\partial y^2 & \partial^2 D/\partial y \partial \sigma \\ \partial^2 D/\partial x \partial \sigma & \partial^2 D/\partial y \partial \sigma & \partial^2 D/\partial \sigma^2 \end{bmatrix}$$

Optimal position is $(x, y, \sigma) + \hat{x}$, where

$$\hat{x} = -\mathrm{H}^{-1} \begin{bmatrix} \partial D/\partial x \\ \partial D/\partial y \\ \partial D/\partial \sigma \end{bmatrix}$$

Low Contrast Extrema Pruning

Predict DoG value at extrema:

$$p = \mid D(x, y, \sigma) + \frac{1}{2}\left[\frac{\partial D}{\partial x}, \frac{\partial D}{\partial y}, \frac{\partial D}{\partial \sigma}\right] \hat{x} \mid$$

Reject if $p < 0.03$

Unstable Point Extrema Pruning

Let

$$\mathrm{H} = \begin{bmatrix} \partial^2 D/\partial x^2 & \partial^2 D/\partial x \partial y \\ \partial^2 D/\partial x \partial y & \partial^2 D/\partial y^2 \end{bmatrix}$$

Reject if $\det(\mathrm{H}) < 0$ or

$$\frac{trace(\mathrm{H})^2}{det(\mathrm{H})} > \tau \ (e.g. 12)$$

Rejects points that can slide along an edge

Getting Rotation Invariance

Local orientation $\hat{\theta}$ estimation

Use keypoint scale σ
Let $\vec{v} = \nabla L(r, s, \sigma)$ for $(r, s) \in neigh(x, y)$
Compute strength $m = \mid \vec{v} \mid$ and
$\theta = direction(\vec{v})$
Compute histogram of θ values weighted by m

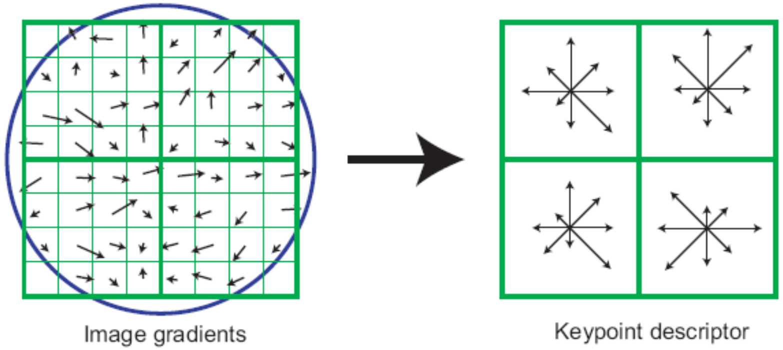
Pick top peak direction $\hat{\theta}$ in histogram for feature orientation

Local Descriptor Computation

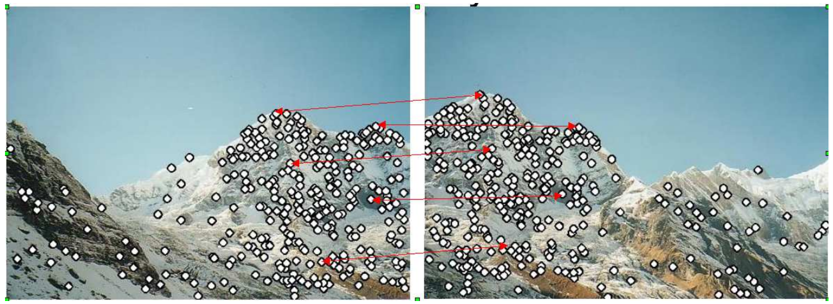
Use 16×16 neighbourhood about feature point subdivided into 16 4×4 pixel blocks
Create an 8 orientation histogram for each block
$\rightarrow$ 128 vector
Compute gradient orientation at each point
Rotate all orientations by $\hat{\theta}$ (for invariance)
Add to histogram weighted (details in paper)
Normalize 128 vector to unit length for illumination invariance
Descriptor similarity using Euclidean distance

Descriptor Example

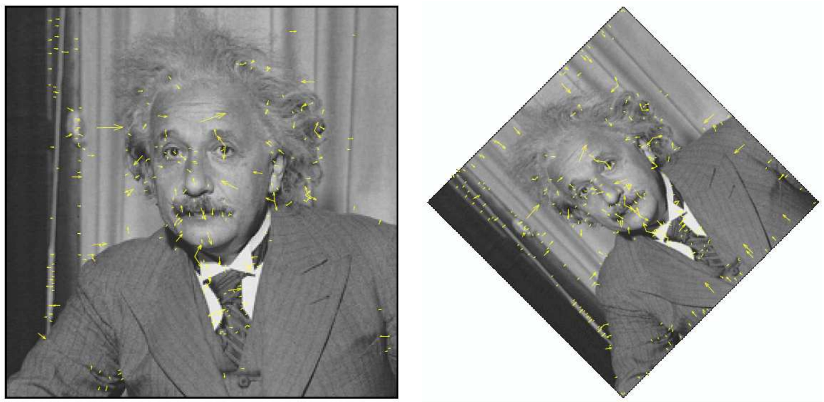
4 histograms from 8×8 neighbourhood about feature point:



Example Points



Rotation & Scale Invariance



SIFT Summary

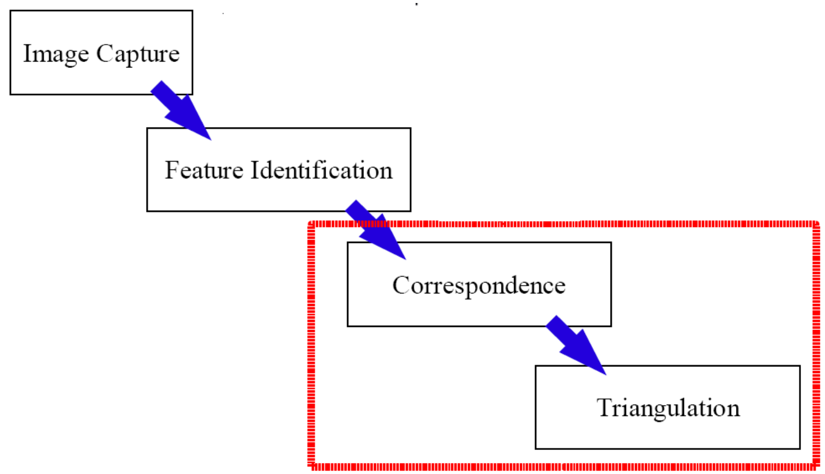
- Sparse, distinctive point features
- Translation independent by using local histogram
- Rotation independent by orientation adjustment
- Scale independent by extremal scale estimation
- Illumination independent by descriptor normalisation
- Widely used
- Real-time implementation possible

SIFT References

www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

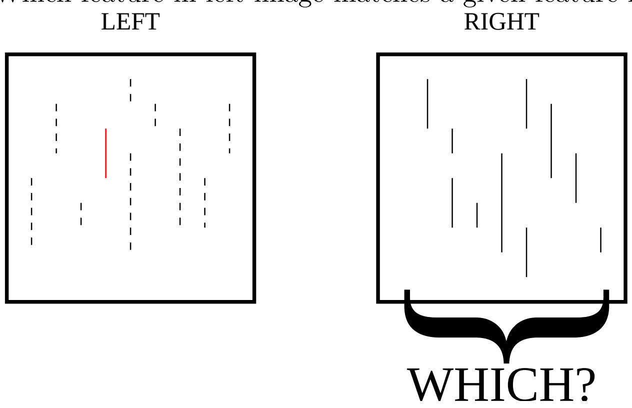
en.wikipedia.org/wiki/Scale-invariant_feature_transform

Stereo Overview



Stereo Correspondence Problem

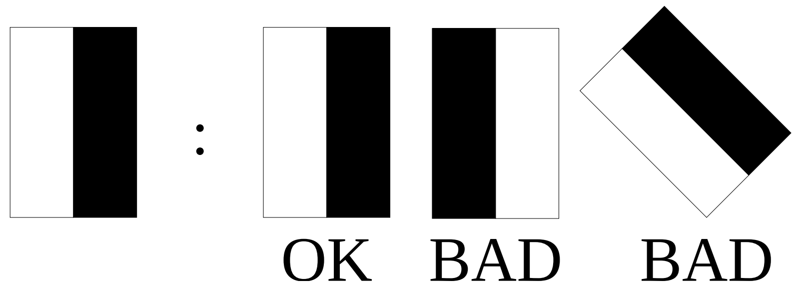
Which feature in left image matches a given feature in the right?



Different pairings give different depth results

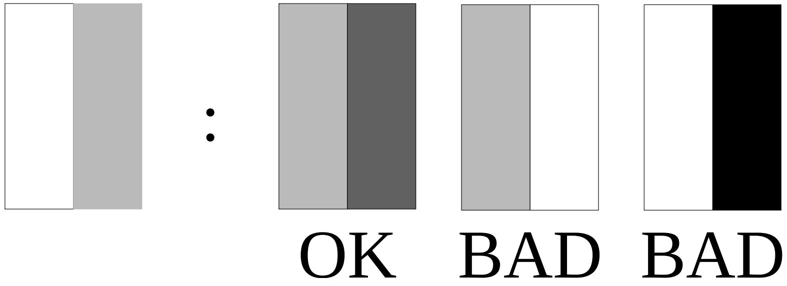
Often considered the key problem of stereo

Constraining Matches: Edge Direction



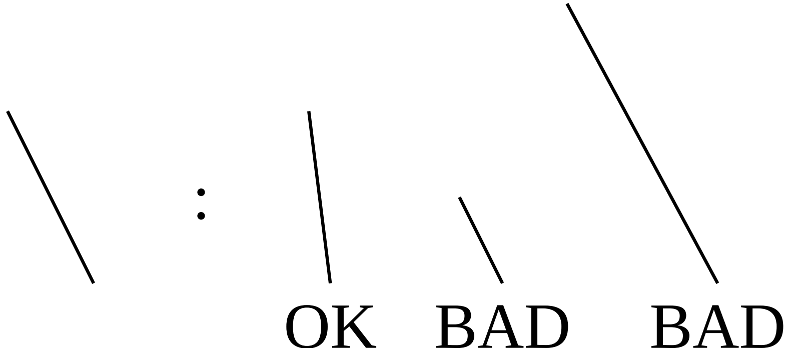
Match features with nearly same orientation

Constraining Matches: Edge Contrast



Match features with nearly same contrast across edge

Constraining Matches: Feature Shape



Match features with nearly same length

Constraining Matches: Uniqueness and Smoothness

Smoothness: match features giving nearly same depth as neighbors

Uniqueness: a feature in one image can match from the other image:

0 - occlusion

1 - normal case

2+ - transparencies, wires, vines, etc from coincidental alignments

Midlecture Problem

Which stereo correspondence constraint would you use to reject these matches?

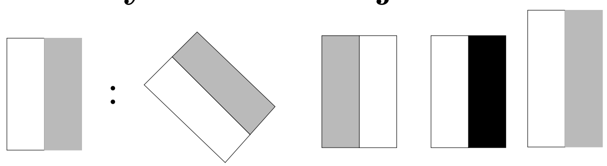
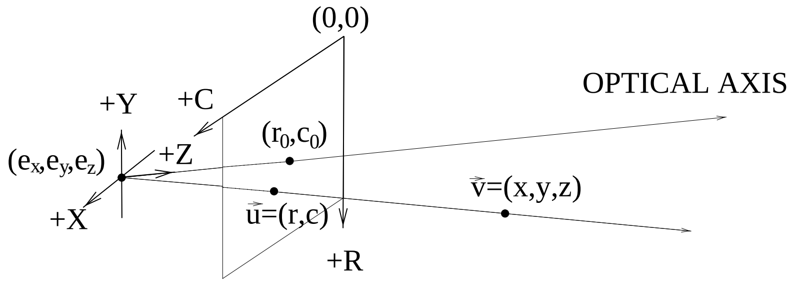


Image Projection Geometry

Pinhole camera model: Matrix P_i projects 3D point $\vec{v} = (x, y, z, 1)'$ onto image point $\vec{u}_i = (r_i, c_i, 1)'$: $\lambda_i \vec{u}_i = \mathrm{P}_i \vec{v}$. $i = L, R$.

Notice use of homogeneous coordinates in 2D and 3D



Projection matrix P_i decomposes as

$$\mathrm{P}_i = \mathrm{K}_i \mathrm{R}_i [I | -\vec{e}_i]$$

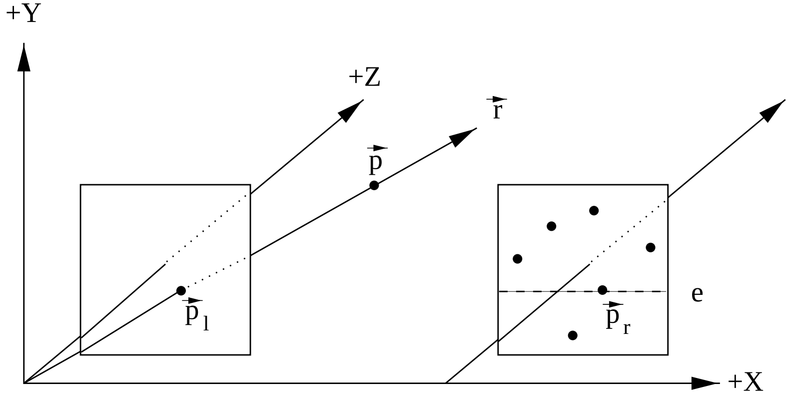
where R_i : orientation of camera (3 degrees of freedom)
$\vec{e}_i = (e_{xi}, e_{yi}, e_{zi})'$: camera center in world (3 DoF)
K_i : camera intrinsic calibration matrix =

$$\begin{bmatrix} f_i m_{ri} & s_i & r_{0i} \\ 0 & f_i m_{ci} & c_{0i} \\ 0 & 0 & 1 \end{bmatrix}$$

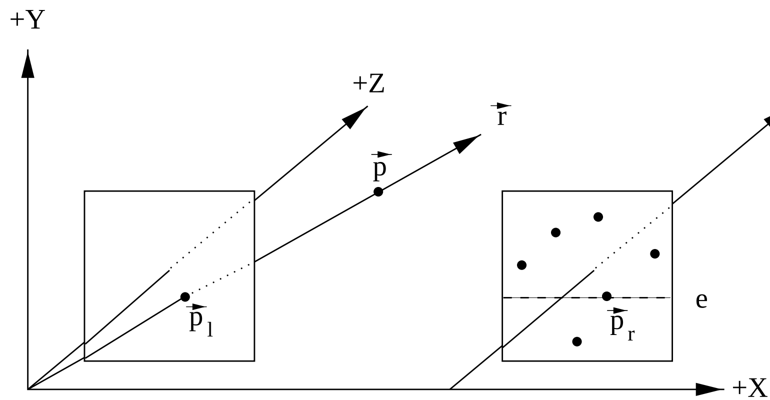
f_i : camera focal length in mm
m_{ri}, m_{ci} : row, col pixels/mm conversion on image plane
r_{0i}, c_{0i} : where optical axis intersects image plane
s_i : skew factor
==========
12 parameters (11 Degrees of Freedom) per camera

Constraining Matches: Epipolar Geometry

Feature $\vec{p}_l$ in left image lies on a ray $\vec{r}$ thru space.
$\vec{r}$ projects to an epipolar line e in the right image, along which the matching image feature must lie.

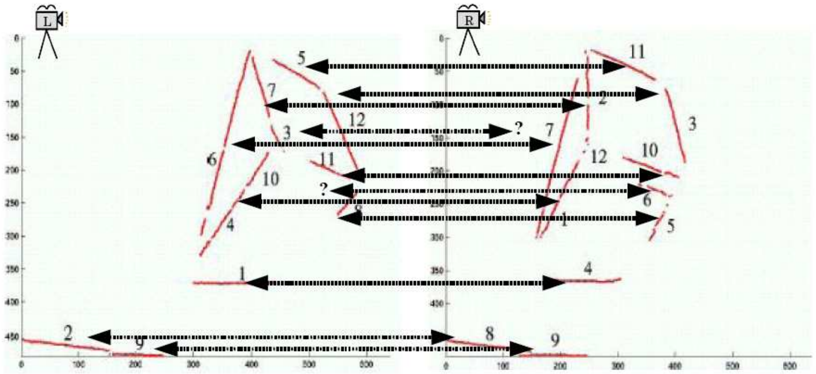


If images are 'rectified', then the epipolar line is an image row.
Reduces 2D search to 1D search



Images are linked by the **Fundamental matrix F**
Matched points satisfy $\vec{p}_l{}' \mathrm{F} \vec{p}_r = 0$
(Points are in homogeneous coordinates, **F** is 3×3)

Stereo Matching Results



Maximally consistent set of matches
Based on local and epipolar constraints

Estimating the Fundamental matrix

Assume $N \geq 7$ matched points $\vec{u}_i : \vec{v}_i, i = 1 \ldots N$ in 2 images
Each should satisfy $\vec{u}_i' \mathrm{F} \vec{v}_i = 0$
Noisy, so use a least squares algorithm. Expanding $\vec{u}_i' \mathrm{F} \vec{v}_i$ gives an equation in N variables:

$$[u_{ix}v_{ix}, u_{ix}v_{iy}, u_{ix}, u_{iy}v_{ix}, u_{iy}v_{iy}, u_{iy}, v_{ix}, v_{iy}, 1]\vec{f} = A_i \vec{f} = 0$$

when we unfold

$$\mathrm{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

into $\vec{f} = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})'$.

Then we stack the A_i up as:

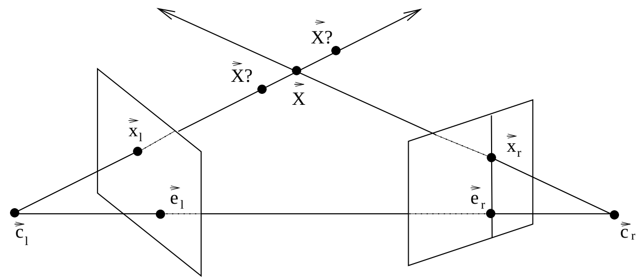
$$\mathrm{A}\vec{f} = \begin{bmatrix} A_1 \\ \ldots \\ A_N \end{bmatrix} \vec{f} = 0$$

Solve for $\vec{f}$: $\mathrm{svd}(A) = UDV'$, $\vec{f} = V(:, 9)$ (Plus some numerical fixes)

Not numerically best algorithm, but simple to understand
See Hartley and Zisserman Chapter 10

Epipoles

Line connecting the 2 camera centres intersects the image planes



Estimate epipoles $\vec{e}_l$, $\vec{e}_r$, by exploiting $\vec{e}_l{}'F = F\vec{e}_r = \vec{0}$

Solve 3 equations
in 2 variables for unknown epipoles $(e_{lx}, e_{ly}, 1)F = F(e_{rx}, e_{ry}, 1)' = \vec{0}$

```
eR=null(F,'r'); eR = eR/eR(3)  eL=null(F'); eL = eL/eL(3)
```

Estimating Projection Matrices

Use the epipoles

$$P_L = K_L * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P_R = K_R * \left[\begin{bmatrix} 0 & -1 & e_{ry} \\ 1 & 0 & -e_{rx} \\ -e_{ry} & e_{rx} & 0 \end{bmatrix} * F \,\middle|\, \vec{e}_r \right]$$

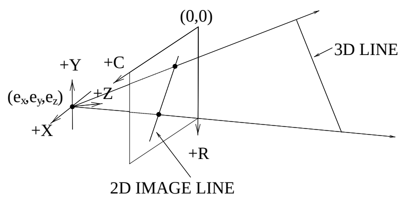
3D Line Calculation

Aim: recovery of 3D line positions

Assume: line successfully matched in L & R images

1. Compute 3D plane that goes through image line and camera origin
2. Compute intersection of 3D planes from 2 cameras (which gives a line)

3D plane passing thru 2D image line



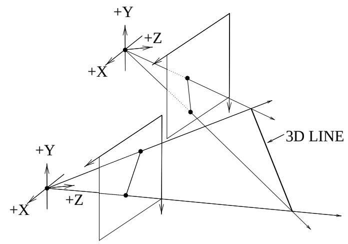
2D image line $l = [a, b, c]'$ is:

$$a * col + b * row + c = 0$$

Then plane is $l'P$

Do for left and right images

3D Plane Intersection → 3D Line

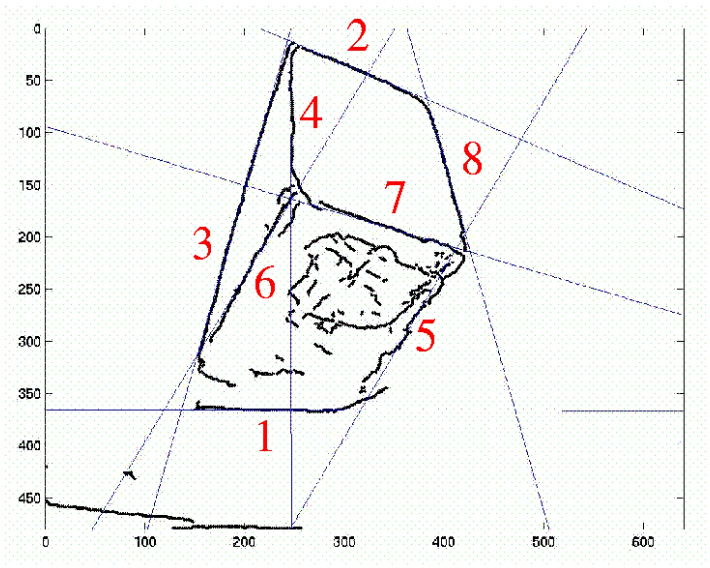


Let $\vec{l}_L$ and $\vec{l}_R$ be the left and right image lines (2D, but in homogeneous coordinates)

3D line represented by 2×4 matrix:

$$L = \begin{bmatrix} \vec{l}_L' P_L \\ \vec{l}_R' P_R \end{bmatrix}$$

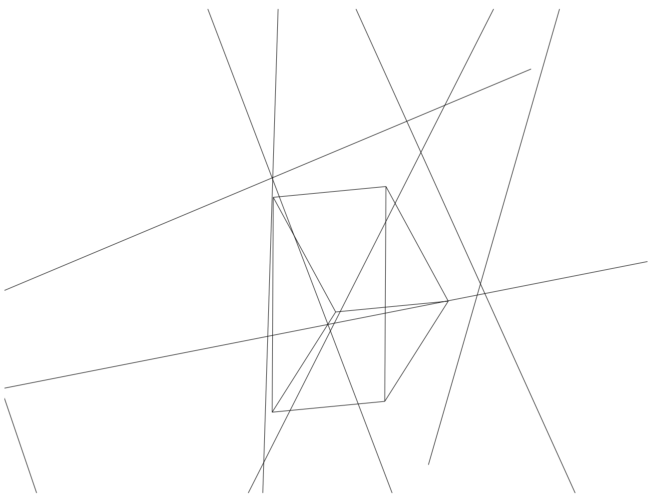
Matched Lines



3D Line Equations

Number	Pairs	direction	point
1	L1:R4	(-0.82, 0.08,-0.56)	(9.1,2.0,-13.0)
2	L5:R11	(0.61,-0.06, 0.78)	(-125.3,98.6,107.1)
3	L6:R7	(-0.28,-0.95,-0.03)	(0.9,-10.6,294.4)
4	L7:R2	(0.07,-0.62,-0.77)	(48.3,-97.0,82.9)
5	L8:R5	(-0.18,-0.45, 0.87)	(114.8,91.8,72.1)
6	L10:R12	(-0.50,-0.73, 0.44)	(71.5,77.0,208.8)
7	L11:R10	(0.79,-0.20, 0.57)	(-98.4,57.2,154.6)
8	L12:R3	(0.11,-0.69,-0.70)	(110.4,-123.6,140.1)

Projected Stereo Line Positions

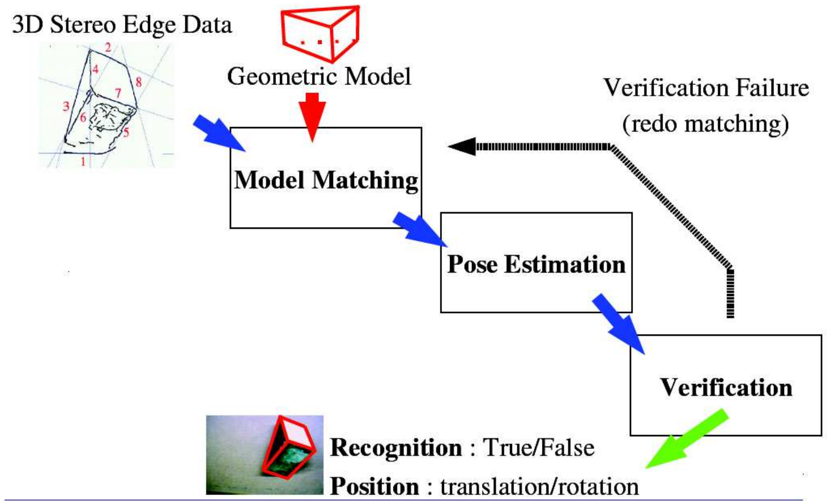


A little off, but: 1) calibration errors, 2) didn't account for lens distortion

Angles Between Lines

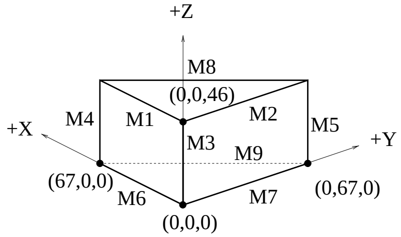
3D line 1	3D line 2	Angle	True
2	3	1.4347	1.57
2	4	1.0221	1.57
2	5	0.9281	1.57
2	6	1.4863	1.57
2	7	0.3023	0.00
2	8	1.1186	1.57
3	4	0.9180	0.71
3	5	1.0964	0.71
3	6	0.5848	0.71
3	7	1.5221	1.57
3	8	0.8457	0.71
4	5	1.1527	1.57
4	6	1.4920	1.57
4	7	1.3026	1.57
4	8	0.1085	0.00
5	6	0.6152	0.00
5	7	1.1060	1.57
5	8	1.2453	1.57
6	7	1.5679	1.57
6	8	1.4276	1.57
7	8	1.3918	1.57

3D Edge Based Recognition Pipeline



3D Edge Based Recognition

Match 3D data edges to 3D wireframe model edges



Model =
(0,0,0)-(67,0,0)
(0,0,0)-(0,67,0)
(0,0,0)-(0,0,46)
(67,0,0)-(67,0,46)
(0,67,0)-(0,67,46)
(0,0,46)-(67,0,46)
(0,0,46)-(0,67,46)
(67,0,0)-(0,67,0)
(67,0,46)-(0,67,46)

3D Model Matching

Use Interpretation Tree algorithm: match edges, Limit = 5

Unary test: similar length $| l_m - l_d | < \tau_l (l_m + l_d)$

Binary test: similar angle between pairs: $| \theta_m - \theta_d | < \tau_a$

Matching Performance

144 interpretation tree matches thru to pose estimation and verification

Two valid solutions found (symmetric model rotated 180 degrees)

Data Segment	Model 1 Segment	Model 2 Segment
3	M8	M9
4	M2	M6
6	M1	M7
7	M3	M3
8	M7	M1

3D Pose Estimation

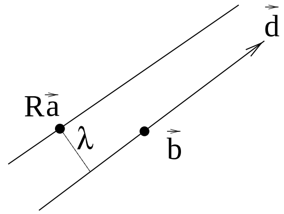
Given: matched line directions $\{(\vec{m}_i, \vec{d}_i)\}$ and points on corresponding lines (but not necessarily same point positions) $\{(\vec{a}_i, \vec{b}_i)\}$

Rotation (matrix R): estimate rotation from matched vectors (same as previous task) except:
1) use line directions instead of surface normals
2) don't know which $\pm$ direction for edge correspondence: try both for each matched segment
3) if $det(\mathrm{R}) = -1$ then need to flip symmetry
4) verify rotation by comparing rotated model and data line orientations

School of Informatics, University of Edinburgh

3D Translation Estimation

Given N paired model and data segments, with point $\vec{a}_i$ on model segment i and $\vec{b}_i$ on data segment i
Direction $\vec{d}_i$ of data segment i
Previously estimated rotation R

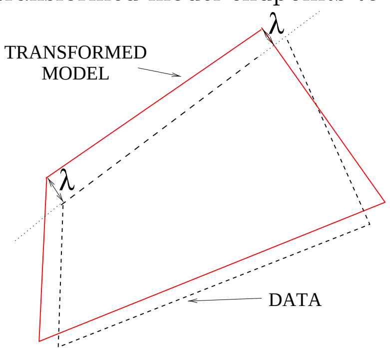


$\vec{\lambda}_i = \mathrm{R}\vec{a}_i + \vec{t} - \vec{b} - \vec{d}_i(\vec{d}_i'(\mathrm{R}\vec{a}_i + \vec{t} - \vec{b}))$ is translation error to minimize
Goal: find $\vec{t}$ that minimizes $\sum_i \vec{\lambda}_i' \vec{\lambda}_i$

School of Informatics, University of Edinburgh

How: $\mathbf{L} = \sum_i (I - \vec{d}_i\vec{d}_i')'(I - \vec{d}_i\vec{d}_i')$
$\vec{n} = \sum_i (I - \vec{d}_i\vec{d}_i')'(I - \vec{d}_i\vec{d}_i')(\mathrm{R}\vec{a}_i - \vec{b}_i)$
$\vec{t} = \mathrm{L}^{-1}\vec{n}$

Verify translation by comparing perpendicular distance of transformed model endpoints to data line

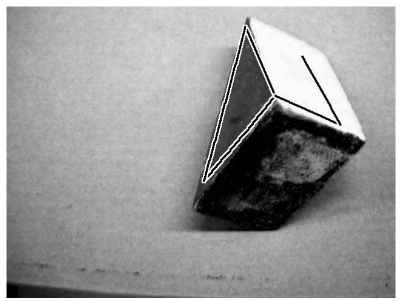
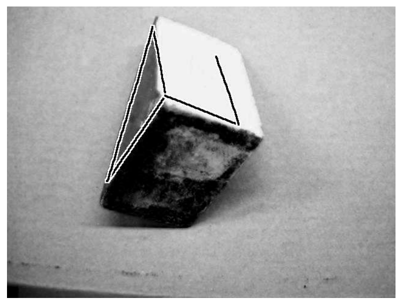


287 verify attempts (2 successes)

School of Informatics, University of Edinburgh

Matching Overlay

Two solutions for symmetric model
Left and right image with matched portion of model overlaid:

Calibration a bit off

Discussion

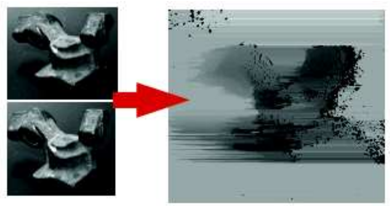
- Hard to find reliable edges/lines, but Canny finds most reasonable edges and RANSAC can put them together for lines
- Given enough stereo correspondence constraints, can get reasonably correct correspondences
- Large features help stereo matching but require more preprocessing
- Stereo geometry easy but needs accurate calibration: not always easy, but now possible to autocalibrate using 7 matched points
- Binocular feature matching stereo gives good 3D at corresponding features, but nothing in between: use scan line stereo?

Dense Depth Data

Problem: have depth only at triangulated feature locations

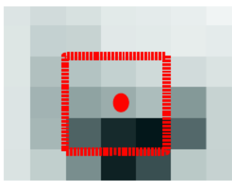
Solution 1: Linear interpolate known values at all other pixels

Solution 2: Correlation-based stereo
Use pixel neighborhoods as features
Triangulate depth at every pixel
But needs to find matching pixel - not easy



Correlation based stereo

- Use stereo image pair



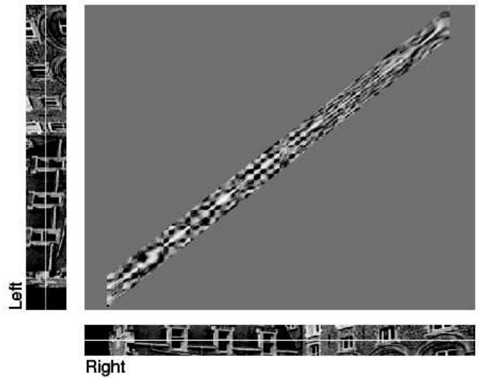
- Features are neighborhoods at each pixel
- Match using similarity metric: SSD - Sum of Squared Differences (of pixel values) of left image at (u, v) to right image at (r, s):

$$SSD(u,v,r,s) = \sum_{i=-\frac{N}{2}}^{\frac{N}{2}} \sum_{j=-\frac{N}{2}}^{\frac{N}{2}} (L(u+i, v+j) - R(r+i, s+j))^2$$

Finding best match

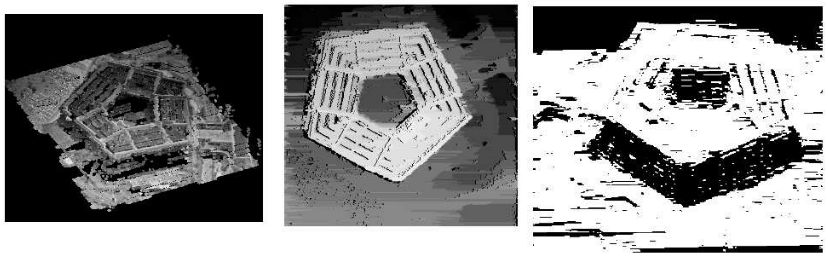
For each scanline on rectified image pair:

1. Build array of all possible matching scores



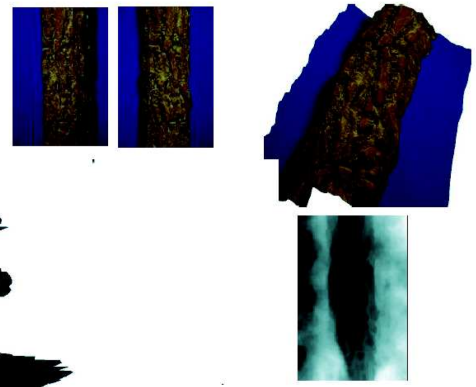
2. Dynamic programming finds lowest cost path (bright line thru middle of array above - optimisation problem)

Dense Stereo Results



Technique = [Cox et al. 1996]

Commercial Dense Stereo Results



http://www.di3d.com/

What We Have Learned

Feature based stereo:

- Feature detection: Canny & RANSAC
- Stereo matching: local & epipolar constraints
- Triangulation & 3D: epipolar geometry
- Recognition: IT algorithm & verification

Correlation based stereo: similar but using pixel neighbourhoods

The Course

- intensity, stereo and range sensors
- Feature detection: curve segmentation, Canny edges, plane region growing, RANSAC line detector, least square estimation
- Geometric coordinate systems, modeling, recognition and pose estimation: 2D and 3D
- Matching algorithms: Interpretation Tree
- Match verification by feature alignment
- Point distribution modelling for variable shapes