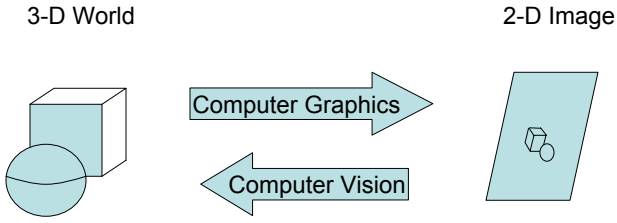
Perception (Vision)

- Sensors
  - images (RGB, infrared, multispectral, hyperspectral)
  - touch sensors
  - sound

Perceptual Tasks

- Scene Understanding
  - Reconstruct the location and orientation ("pose") of all objects in the scene
  - If objects are moving, determine their velocity (rotational and translational)
- Object Recognition
  - Identify object against arbitrary background
  - Face recognition
  - "Target" recognition
- Task-specific Perception (Minimum perception needed to carry out task)
  - Obstacle avoidance
  - Landmark identification

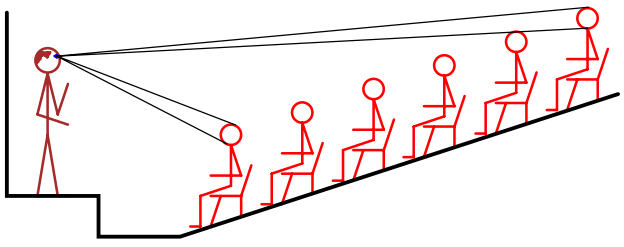
Scene Understanding: Vision as Inverse Graphics



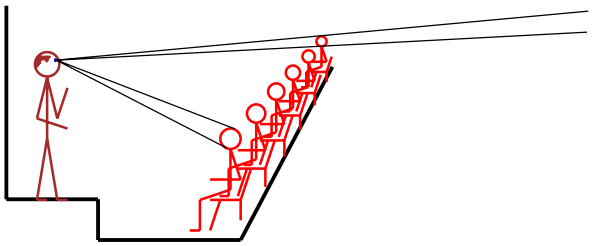
Fundamental problem:

- 3-D → 2-D transformation loses information

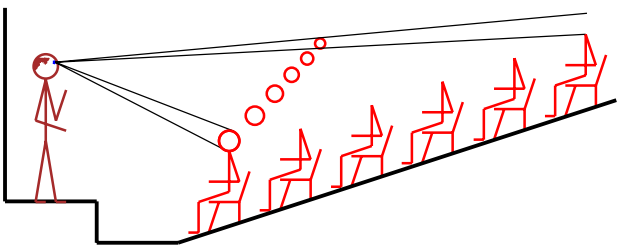
3-D ➔ 2-D Information Loss



3-D ➔ 2-D Information Loss



3-D ➔ 2-D Information Loss



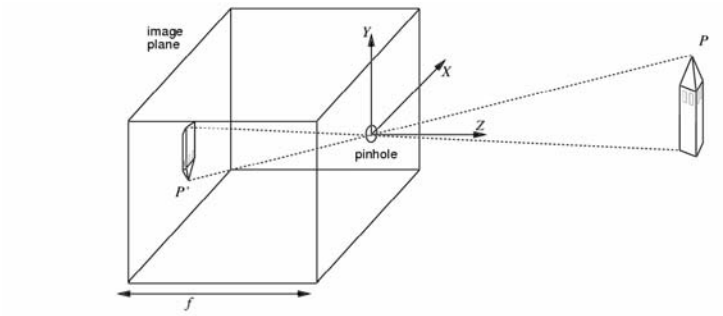
Probabilistic Formulation

- I: image
- W: world
- Goal:
  - $\text{argmax}_W \; P(W|I) = \text{argmax}_W \; P(I|W) \cdot P(W)$
  - Which worlds are more likely?

Image Formation

- Object location (x,y,z) and pose (r, θ, ω)
- Object surface color
- Object surface material (reflectance properties)
- Light source position and color
- Camera position and focal length

Image Formation



P is a point in the scene, with coordinates (X, Y, Z)
P' is its image on the image plane, with coordinates (x, y, z)

$$x = \frac{-fX}{Z}, \; y = \frac{-fY}{Z}$$

by similar triangles. Scale/distance is indeterminate!

Inverse Graphics Fallacy

- We don't really need to know the location of every leaf on a tree to avoid hitting the tree while driving
- Only extract the information necessary for intelligent behavior!
  - obstacle avoidance
  - face recognition
  - finding objects in your room
- The probabilistic framework is still useful in each of these tasks

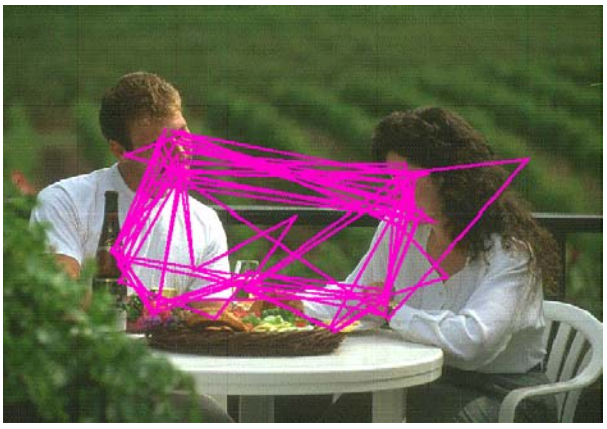
We do not form complete models of the world from images

Another Example

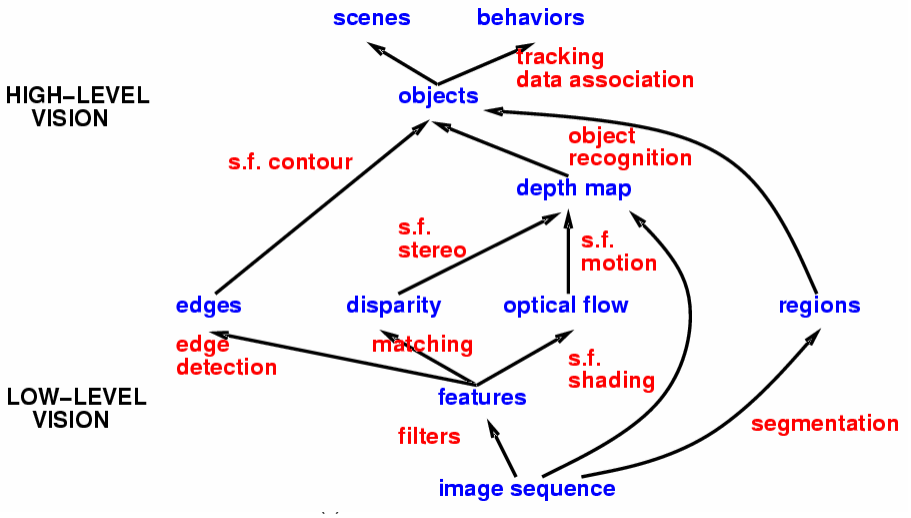
And Another

The Point:

- We only attend to the “relevant” part of the image



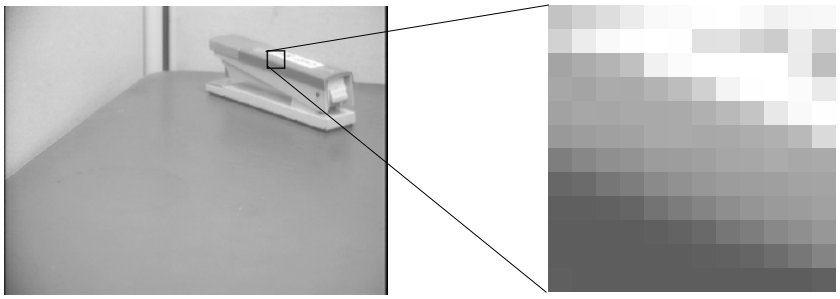
Computer Vision



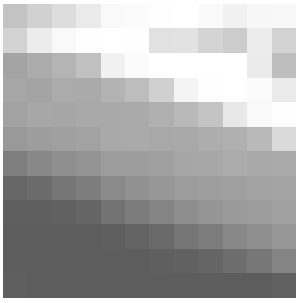
Bottom-Up vs. Top-Down

- Bottom-Up processing
  - starts with image and performs operations in parallel on each pixel
  - find edges, find regions
  - extract other important cues C
- Top-Down processing
  - starts with P(W) expectations
  - computes P(C | W) for groups of cues C

Edge Detection



Edge Detection (2)



195	209	221	235	249	251	254	255	250	241	247	248
210	236	249	254	255	254	225	226	212	204	236	211
164	172	180	192	241	251	255	255	255	255	235	190
167	164	171	170	179	189	208	244	254	255	251	234
162	167	166	169	169	170	176	185	196	232	249	254
153	157	160	162	169	170	168	169	171	176	185	218
126	135	143	147	156	157	160	166	167	171	168	170
103	107	118	125	133	145	151	156	158	159	163	164
095	095	097	101	115	124	132	142	117	122	124	161
093	093	093	093	095	099	105	118	125	135	143	119
093	093	093	093	093	093	095	097	101	109	119	132
095	093	093	093	093	093	093	093	093	093	093	119

Look for changes in brightness

- Compute Spatial Derivative

$$\left(\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y}\right)$$

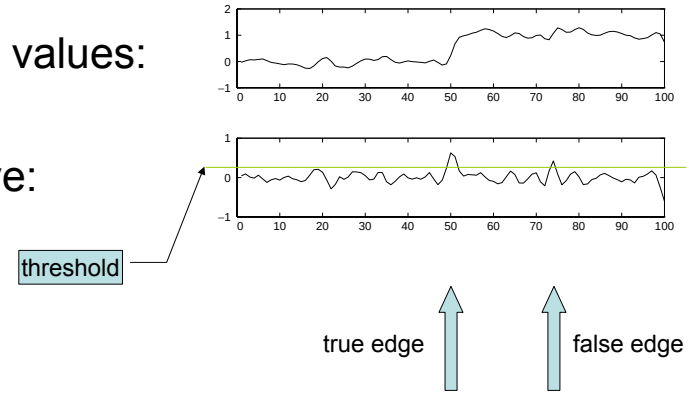
- Compute Magnitude

$$\left(\frac{\partial I(x,y)}{\partial x}\right)^2 + \left(\frac{\partial I(x,y)}{\partial y}\right)^2$$

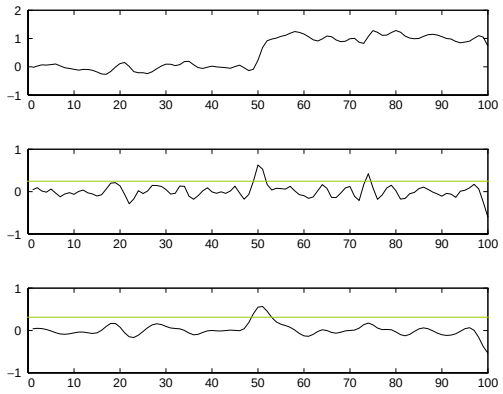
- Threshold

Problem: Images are Noisy

- intensity values:
- derivative:



Solution: Smooth Edges Prior to Edge Detection



Derivative of Smoothed Intensities:

Efficient Implementation: Convolutions

$$h = f * g$$

$$h(x,y) = \sum_{u=-\infty}^{u=+\infty} \sum_{v=-\infty}^{v=+\infty} f(u,v) \cdot g(x-u, y-v)$$

- Smoothing: Convolve image with gaussian
- f(x,y) = I(x,y) the image intensities
- g(u,v) = $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-(u^2+v^2)/2\sigma^2}$

Convolutions can be performed using Fast Fourier Transform

- $FFT[f * g] = FFT[f] \cdot FFT[g]$
  - The FFT of a convolution is the product of the FFTs of the functions
- $f * g = FFT^{-1}(FFT[f] \cdot FFT[g])$

Computing the Derivative

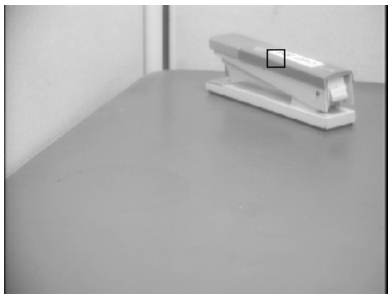
- $(f * g)' = f * (g')$
  - The derivative of a convolution can be computed by first differentiating one of the functions
- To take the derivative of the image after gaussian smoothing, first differentiate the gaussian and then smooth with that!
- Can only be done in one dimension: do it separately for x and y.

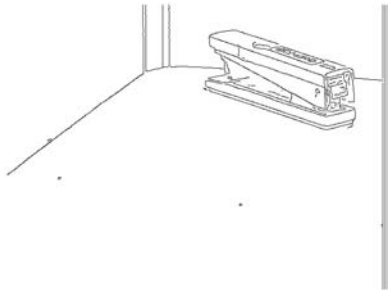
Canny Edge Detector

$$
\begin{aligned}
f_V(u,v) &= G'_\sigma(u)G_\sigma(v) \\
f_H(u,v) &= G_\sigma(u)G'_\sigma(v) \\
R_V &= I * f_V \\
R_H &= I * f_H \\
R(x,y) &= R_V(x,y)^2 + R_H(x,y)^2
\end{aligned}
$$

- Define an edge where R(x,y) > θ (a threshold)

Results





Interpreting Edges

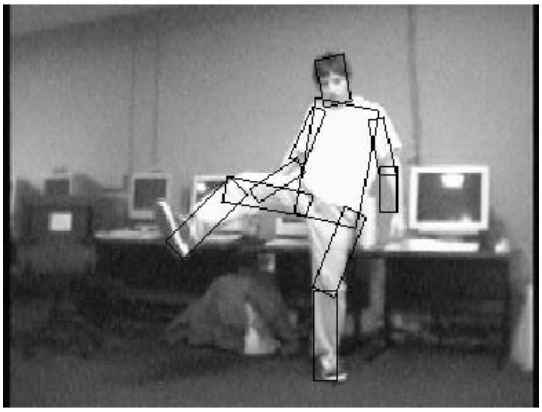
- Edges can be caused by many different phenomena in the world:
  - depth discontinuities
  - changes in surface orientation
  - changes in surface color
  - changes in illumination

Example Optical Illusion

Steps Movie

Bayesian Model-Based Vision
(Dan Huttonlocher & Pedro Felzenszwalb)

- Goal: Locate and track people in images

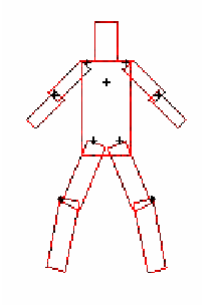


White Lie Warning

- The actual method is significantly different than the version I'm describing here
- For the real story, see the following paper:
  - Efficient Matching of Pictorial Structures, Proceedings of the IEEE Computer Vision and Pattern Recognition Conference, pp. 66-73, 2000
  - http://www.cs.cornell.edu/~dph/

Probabilistic Model of a Person

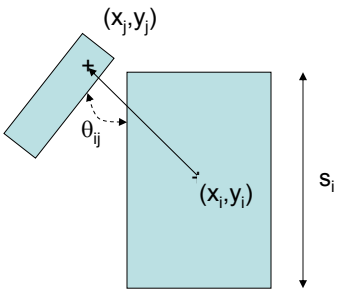
- 10 body parts
- connected at points
- probability distribution over the locations of the points
- probability distribution over relative orientations of the parts
- appearance distribution tells what each part looks like



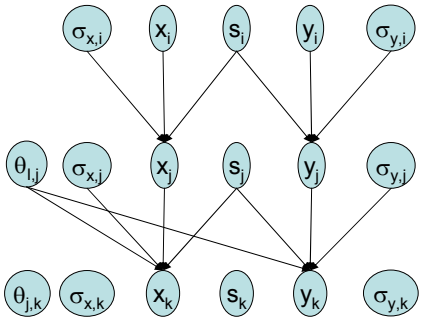
- $P(L|I) \propto P(I|L) \cdot P(L)$

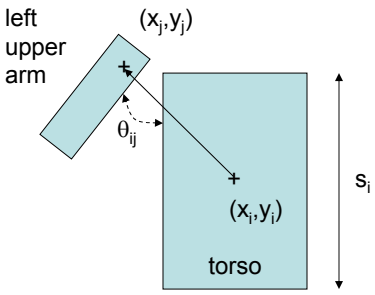
Relationship between body part locations

- Each body part is represented as a rectangle
- s_i = degree of foreshortening
- (x_j,y_j) = relative offset
- $\theta_{i,j}$ = relative orientation



Bayesian Network Model





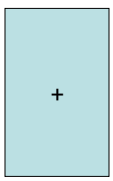
$\underline{P}(s_i) = \text{Gauss}(s_i;1,\sigma_{s,i})$

$\underline{P}(x_j|x_i,\sigma_{xi},s_i) = \text{Gauss}(x_j;\ x_i + \delta_{x,l,j}\cdot s_i,\ \sigma_{x,i})$

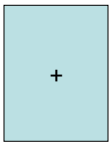
$\underline{P}(y_j|y_i,\sigma_{yi},s_i) = \text{Gauss}(y_j;\ y_i + \delta_{y,l,j}\cdot s_i,\ \sigma_{y,i})$

$\underline{P}(\theta_{i,j}) = \text{vonMises}(\theta_{i,j},\mu_{l,j},k_{l,j})$

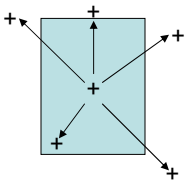
Generating a Person: Step 1: Position of Torso



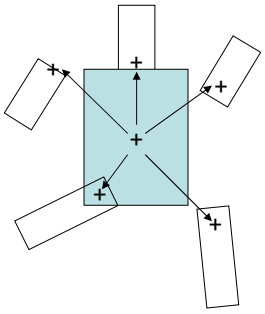
Step 2: Foreshortening of Torso



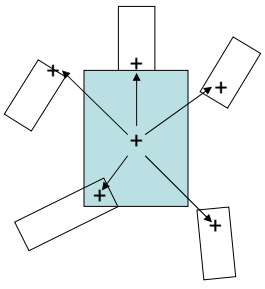
Step 3: Arm, Leg, and Head Joints



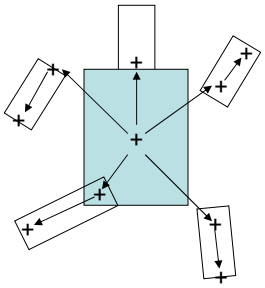
Choose Angle for Each Body Part



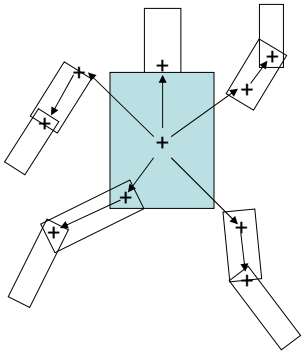
Choose Foreshortening for each part



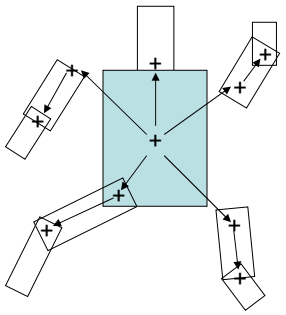
Choose joints of next parts



Choose Angles of Forearms and Lower Legs

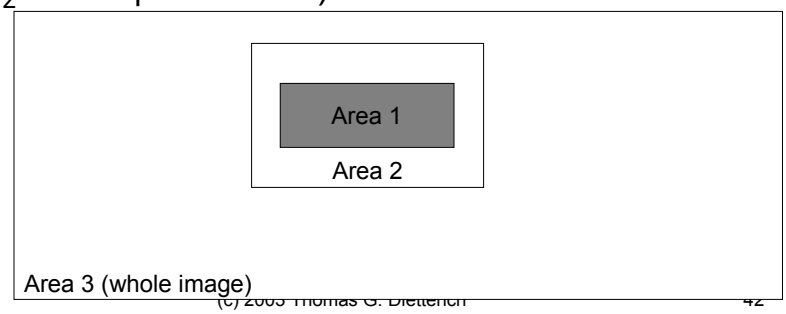


Choose foreshortening of forearms and lower legs



Appearance Model

- Each pixel z is either a foreground pixel (a body part) or a background pixel.
- $P(f_z = \text{true} \mid z \in \text{Area1}) = q_1$
- $P(f_z = \text{true} \mid z \in \text{Area2}) = q_2$
- $P(f_z = \text{true} \mid z \in \text{Area3}) = 0.5$



Appearance Model (2)

- Each part has an average grey level (and a variance). Each pixel z generates its grey level from a Gaussian distribution:
  - $P(g_z \mid f_z=\text{true}, z \in \text{part}_i) = \text{Gauss}(g_z; \mu_i, \sigma_i)$
- Background pixels have average grey level and variance
  - $P(g_z \mid f_z=\text{false}, z \in \text{background}) = \text{Gauss}(g_z; \mu_b, \sigma_b)$
- Does not handle overlapping body parts

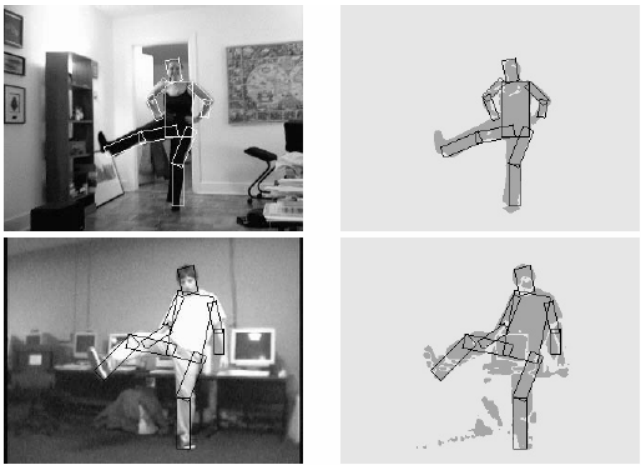
Generating the Image

- Generate body location and pose
- Generate pixel foreground/background for each pixel independently
- Generate pixel grey levels

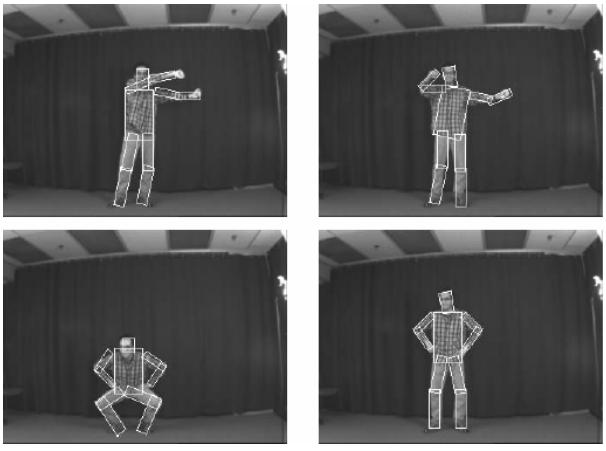
Training

- All model parameters can be fit by supervised training
  - Manually identify location and orientation of body parts
  - Fit joint location and angle distributions, foreshortening distributions
  - Fit q_1 and q_2 foreground probabilities
  - Fit grey level distributions

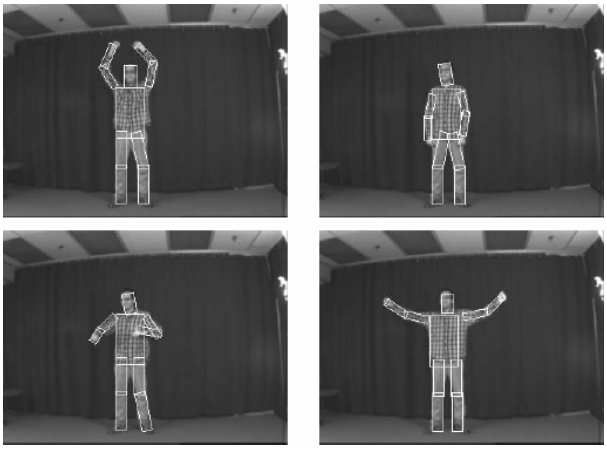
Examples



More Examples



More examples



Implementation Tricks

- argmax_L P(L|I)
  - In theory this would require iterating over all locations L in the image I
  - In practice, the authors developed clever algorithms for using gaussian filter banks to find promising locations and dynamic programming methods for computing the probabilities

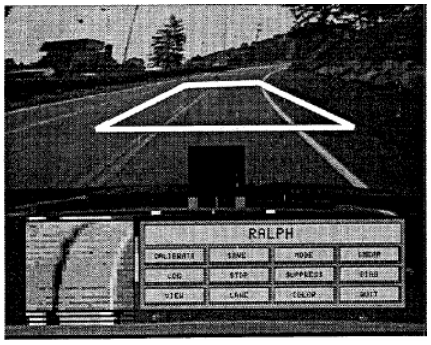
Task-Specific Computer Vision: CMU NavLab Autonomous Driving

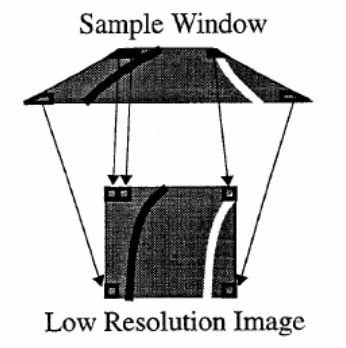
- Camera mounted on rear-view mirror takes image of the road ahead of the vehicle
- Goal: Determine curvature of the road and location of the vehicle in the lane

NavLab (2)

- Trapezoidal region is extracted
  - based on camera geometry
  - vehicle speed
  - so that each scan line in trapezoid covers same size region in physical world (assuming flat road surface)
- Trapezoidal Image is then re-sampled to produce a rectangular image
- For each of several road curvature hypotheses, the rectangular image is recomputed to produce an image that would be straight if the curvature hypothesis is correct
- These images are scored to see which one gives the straightest image and the corresponding curvature hypothesis is accepted

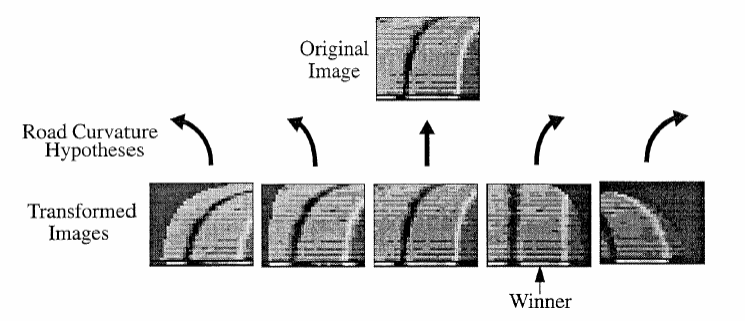
RALPH images



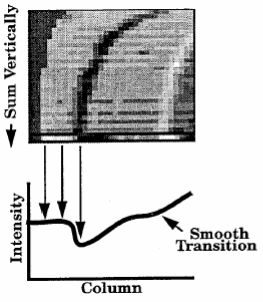


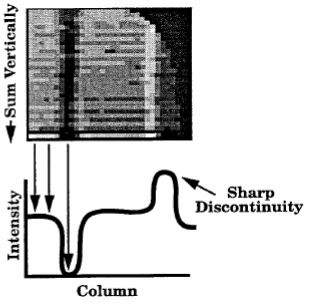
Choosing the Best Road Curvature Hypothesis

- argmax_h straightness(transformedImage(I,h))



Measuring Straightness





$S(x) = \sum_y I(x,y)$

straightness $= \sum_x |S(x) - S(x+1)|$

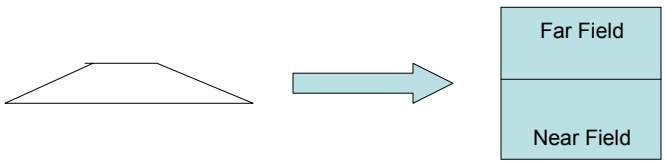
Discussion

- Method works for *any* kind of systematic coloring of the road surface
  - lane marking
  - ruts
  - tire tracks in snow or rain
  - oil droppings in center of lane

Determining Lateral Position

- At time when vehicle is centered in lane, store a template S(x) for all columns x.
  - driver pushes a button
- Compare current template to stored template S(x) under various lateral offsets to find best match ➔ Gives lateral position

Rapidly Learning New Templates



- Subdivide current rectangle into 2 parts
  - Near field is used to determine current lateral position
  - Far Field is used to capture new template

No Hands Across America

- 2797/2849 miles (98.2%) driven autonomously

Computer Vision Summary

- Many different visual tasks; require different amounts of analysis
- Inverse computer graphics is overkill in most cases
- Low level vision: smoothing, edge detection, region finding
  - example: Canny edge detector
- Probabilistic vision methods: H&F people tracker
- Task-specific vision: NavLab lane keeper