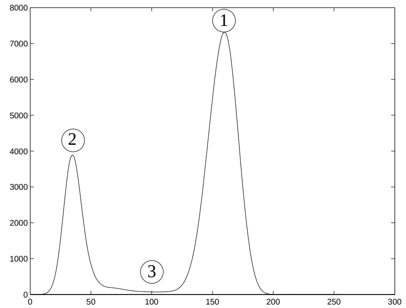
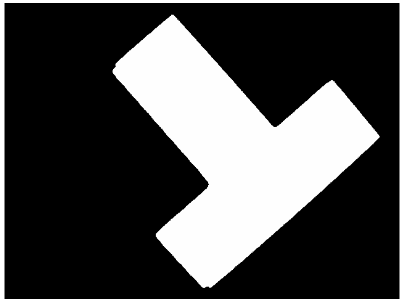
Threshold Selection

Assume 2 big peaks, brighter background is higher:

1. Find biggest peak (background)
2. Find next biggest peak in darker direction
3. Find lowest point in trough between peaks





Peak Pick Code

Omit special cases for ends of array and closing ‘end’s.

```
peak = find(tmp1 == max(tmp1));   % find largest peak

% find highest peak to left
xmaxl = -1;
for i = 2 : peak-1
  if tmp1(i-1) < tmp1(i) & tmp1(i) >= tmp1(i+1) ...
  & tmp1(i)>xmaxl
      xmaxl = tmp1(i);
      pkl = i;
```



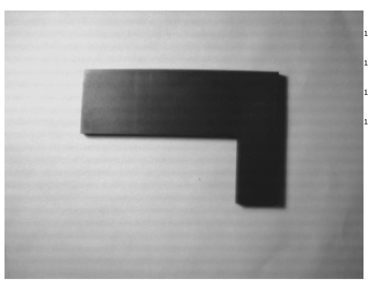
```
% find deepest valley between peaks
xminl = max(tmp1)+1;
for i = pkl+1 : peak-1
  if tmp1(i-1) > tmp1(i) & tmp1(i) <= tmp1(i+1) ...
  & tmp1(i)<xminl
      xminl = tmp1(i);
      thresh = i;
```


Adaptive Thresholding

What if varying and unknown background? Can select threshold locally

At each pixel, use a different threshold calculated from an NxN window (N=100)

Use: threshold = mean(window) - Constant (eg. 12)



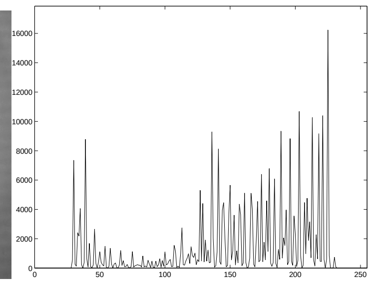


Image with intensity gradient   Histogram

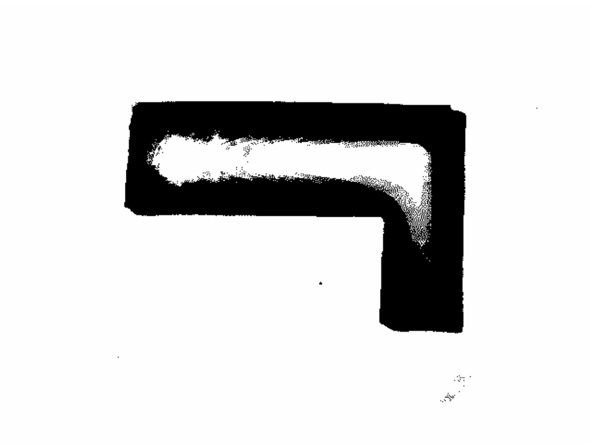
Adaptive Thresholding Code

```
N = 100;
[H,W] = size(inimage);
outimage = zeros(H,W);
N2 = floor(N/2);
for i = 1+N2 : H-N2
  for j = 1+N2 : W-N2
    % extract subimage
    subimage = inimage(i-N2:i+N2,j-N2:j+N2);
    threshold = mean(mean(subimage)) - 12;
    if inimage(i,j) < threshold
     outimage(i,j) = 1;
    else
     outimage(i,j) = 0;
```



```
    end
  end
end
```


Adaptive Thresholding Results



Selection has included shadow at bottom and right

Background Removal

If known but spatially varying illumination

Reflectance: percentage of input illumination reflected. A function of the light source, viewer and surface colors and positions.

Recall:
background(r,c) = illumination(r,c)*bg_reflectance(r,c)
object(r,c) = illumination(r,c)*obj_reflectance(r,c)

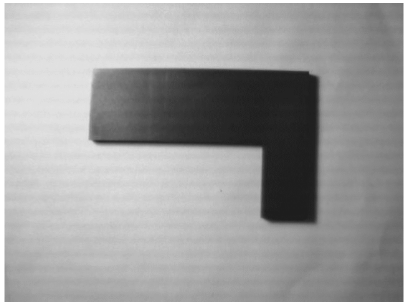
Divide to remove illumination:
unknown(r,c)/background(r,c) =

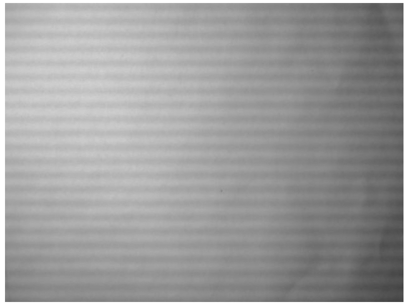
1	if unknown = background
<<1	if unknown = dark object

Pick threshold in [0,1] e.g. 0.6

School of Informatics, University of Edinburgh

Background removal results 1

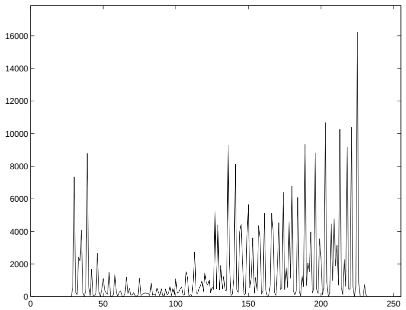


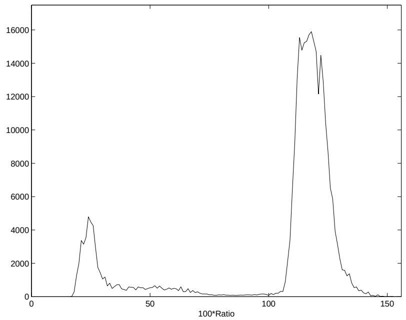


Part Background

School of Informatics, University of Edinburgh

Background removal results 2





Raw histogram ratio histogram

School of Informatics, University of Edinburgh

Background removal results 3

Has also included shadow below and right.

Midlecture Problem

What might happen to the background detection process if the background was highly textured?

Colour Image Detection?

Before

After

change=open(2,coloror(thresh(35,abs(Before-After))))
(Use HSI instead of RGB to cope with illumination changes?)

Colour Image Detection?

Red change

Green change

ORed change

Opened

Isolation in Complex Scenes

Threshold problems with image I:

- Many objects
- Space varying illumination

If have constant background image B (ie. before actions)
Try: $thres(\mid I - B \mid)$ instead of $thres(I)$
Do in each of 3 colour channels:
$thres(\mid I_r - B_r \mid) \| thres(\mid I_g - B_g \mid) \| thres(\mid I_b - B_b \mid)$

Background Differencing Results

BACKGROUND

FOREGROUND

DIFFERENCE

Isolation with varying lighting

Use normalised RGB:

$$(r, g, b) \rightarrow (\frac{r}{r+g+b}, \frac{g}{r+g+b}, \frac{b}{r+g+b})$$

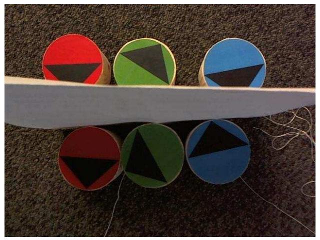
Double illumination still gives same normalised RGB:

$$(\frac{r}{r+g+b}, \frac{g}{r+g+b}, \frac{b}{r+g+b})$$

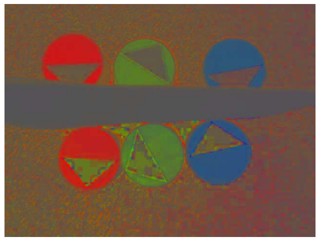
$$= (\frac{2r}{2r+2g+2b}, \frac{2g}{2r+2g+2b}, \frac{2b}{2r+2g+2b})$$

Normalised RGB Example

Original

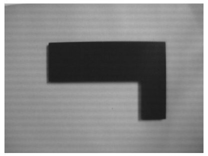


Normalised



Reduces shadow effects, too.

Description for Recognition



'L'-shaped part of length 12 cm, width 8 cm, ...
Hard to get accurate descriptions:

- Need a good language for object description. Here edges or corners would work but not in general - eg. human faces
- Hard to get reliable, consistent data descriptions: noise, shadows, shading, surface texture, highlights, viewpoint changes, ...

So, here use property-based descriptions. A common current approach, but ambiguous (how many flat objects with area A?).

Simple Properties

Let `Image` be a binary image with the desired object as 1

Area - `bwarea(Image)`

Perimeter- `bwarea(bwperim(Image,4))`

Reasonably robust to noise
Independent of translation and orientation
Not independent of scale/zoom

Position and Scale Invariant Properties

compactness:
$\frac{1}{4\pi}\frac{perimeter^2}{area}$ minimum 1.0 for circle

topological properties:
number of corners, concavities

relative properties:
average angle between consecutive line segments

Moments

Family of stable binary (and grey level) shape descriptions

Can be made invariant to translation, rotation, scaling

Let $\{p_{rc}\}$ be the binary (0,1) image pixels for row r and col c where 1 pixels are the object

Moments II

Area $A = \sum_r \sum_c p_{rc}$

Center of mass

$(\hat{r}, \hat{c}) = (\frac{1}{A} \sum_r \sum_c r p_{rc}, \frac{1}{A} \sum_r \sum_c c p_{rc})$

A family of 'central' (translation invariant) moments (for any u and v):

$$m_{uv} = \sum_r \sum_c (r - \hat{r})^u (c - \hat{c})^v p_{rc}$$

Subtracting center of mass makes it translation invariant

Scale invariant moments

If double in dimensions, then moment m_{uv} increases by $2^u 2^v$ for weightings and 4 for the number of pixels.

Similarly, area A increases by 4, and thus $A^{(u+v)/2+1}$ increases by $4 \times 2^u 2^v$

So, the ratio:

$$\mu_{uv} = \frac{m_{uv}}{A^{(u+v)/2+1}}$$

is invariant to scale.

Rotation invariant moments

Moment invariant theory has identified methods to generate various orders of moments invariant to rotation. 6 functions ci_i with rescaling applied to get into similar numerical ranges

Area $A = \sum_r \sum_c p_{rc}$
Center of mass $(\hat{r}, \hat{c})$

Define complex uv central moment:
$c_{uv} = \sum_r \sum_c ((r - \hat{r}) + i(c - \hat{c}))^u ((r - \hat{r}) - i(c - \hat{c}))^v p_{rc}$

Scale invariance

Get specific scale invariant moments:
$s_{11} = c_{11}/(A^2)$
$s_{20} = c_{20}/(A^2)$
$s_{21} = c_{21}/(A^{2.5})$
$s_{12} = c_{12}/(A^{2.5})$
$s_{30} = c_{30}/(A^{2.5})$

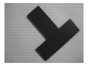
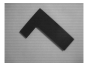
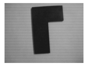
Rotation invariant moments II

Rescaled (so values in similar range) rotation invariants:
$ci_1 = real(s_{11})$
$ci_2 = real(1000 * s_{21} * s_{12})$
$ci_3 = 10000 * real(s_{20} * s_{12} * s_{12})$
$ci_4 = 10000 * imag(s_{20} * s_{12} * s_{12})$
$ci_5 = 1000000 * real(s_{30} * s_{12} * s_{12} * s_{12})$
$ci_6 = 1000000 * imag(s_{30} * s_{12} * s_{12} * s_{12})$

Scaled Moment matlab code

```
function vec = getproperties(Image)
     area = bwarea(Image);
     perim = bwarea(bwperim(Image,4));
     compactness = perim*perim/(4*pi*area);
     c11 = complexmoment(Image,1,1) / (area^2);
     c20 = complexmoment(Image,2,0) / (area^2);
     ...
     ci1 = real(c11);
     ci2 = real(1000*c21*c12);
     tmp = c20*c12*c12;
     ci3 = 10000*real(tmp);
     ...
```

Example invariant property values

			
compactness	1.93	1.81	1.90
ci_1	0.23	0.27	0.25
ci_2	0.18	0.37	0.45
ci_3	0.08	-0.50	0.11
ci_4	-0.00	0.37	-0.64
ci_5	0.23	-0.47	0.09
ci_6	-0.00	0.07	-0.63

Feature Vector

Standard description for many visual processes:
form a vector from set of descriptions:

$$\vec{x} = (compactness, ci_1, ci_2, ci_3, ci_4, ci_5, ci_6)'$$

Multiple vectors if several structures or locations to describe

These vectors are then used in next processes, eg. recognition

What We Have Learned

1. Isolating Objects
2. Moments
3. Moment Invariants
4. Feature Vectors