AI (B) - Computer Vision Part

Pixel Operations, Histogram, Thresholding

version-watch&understand

试用水印

Learning Outcomes

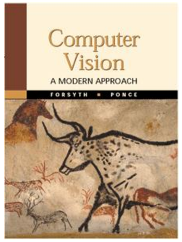
Be able to understand the basics of point operators
Be able to compute histograms, optimal threshold for a given image by programming
Be able to design and apply simple background extraction algorithms.

Contents

- ❑Pixel Processing
- ❑Histogram and Normalization
- ❑Intensity Thresholding
- ❑Background Extraction

试用水印

Reading



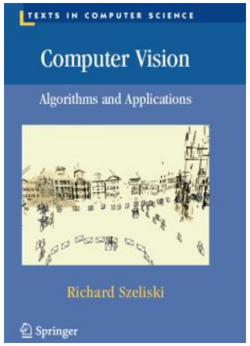
FP: Parts of 7 & 8:

Books from library

Relevant *HIPR2* worksheets

http://homepages.inf.ed.ac.uk/rbf/HIPR2/wksheets.htm

试用水印



RS: Parts of 3

http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

Pixel Processing: Basics

试用水印

Negate (Invert)

Formula: $$\boxed{\begin{array}{l} O(i,j) = Max - I(i,j) \\ Max = 255 \end{array}}$$

Pseudo Code:

```
for (p=0; p<nPixels; p++)
    image[p] = MAXVAL - image[p];
```

试用水印

Examples

Original Image

Inverted Image

试用水印

Contrast Scaling

Scale a selected grey-scale range [low, high] to fill the available range [0,MAXVAL]

$$O(i,j)=\begin{cases} MAXVAL, & I(i,j)>=H \\ \frac{I(i,j)-L}{H-L}*MAXVAL, & L<I(i,j)<H \\ 0, & I(i,j)<=L \end{cases}$$

Pseudo code:

```
for (p=0; p<nPixels; p++)
  if (image[p] >= low && image[p] <= high)
      image[p] =
        (image[p] - low) * MAXVAL / (high-low);
else if (image[p] < low) image[p] = 0;
   else image[p] = MAXVAL;
```

试用水印

Image Arithmetic

addition, subtraction, multiplication, division

Example: averaging two images

```
for (p=0; p<nPixels; p++)
{
  imageout[p] = image1[p] + image2[p];
  imageout[p] = imageout[p] / 2;
}
```

怎么解决overflow的问题？？？？？？？？

when the new value after adding is beyond the 255

Potential for overflow !

试用水印

Hazard of Overflow/Underflow

If pixels are bytes, 200 + 100 = ???

- **Wraparound ?**
- **Stay at maximum ?**

Control it:

- **Step1: make pixel storage larger (double int)**
- **Step2: add images**
- **Step3: scale to original range (single int)**
- **Step4: convert image back to original storage type**

***or* use float or double!**

试用水印

D-I-Y

All these functions (and many more) are explained in more detail in HIPR and examples are given with images.

Almost all these functions can be implemented in Paint Shop Pro & most other image packages

Try some DIY in labs!

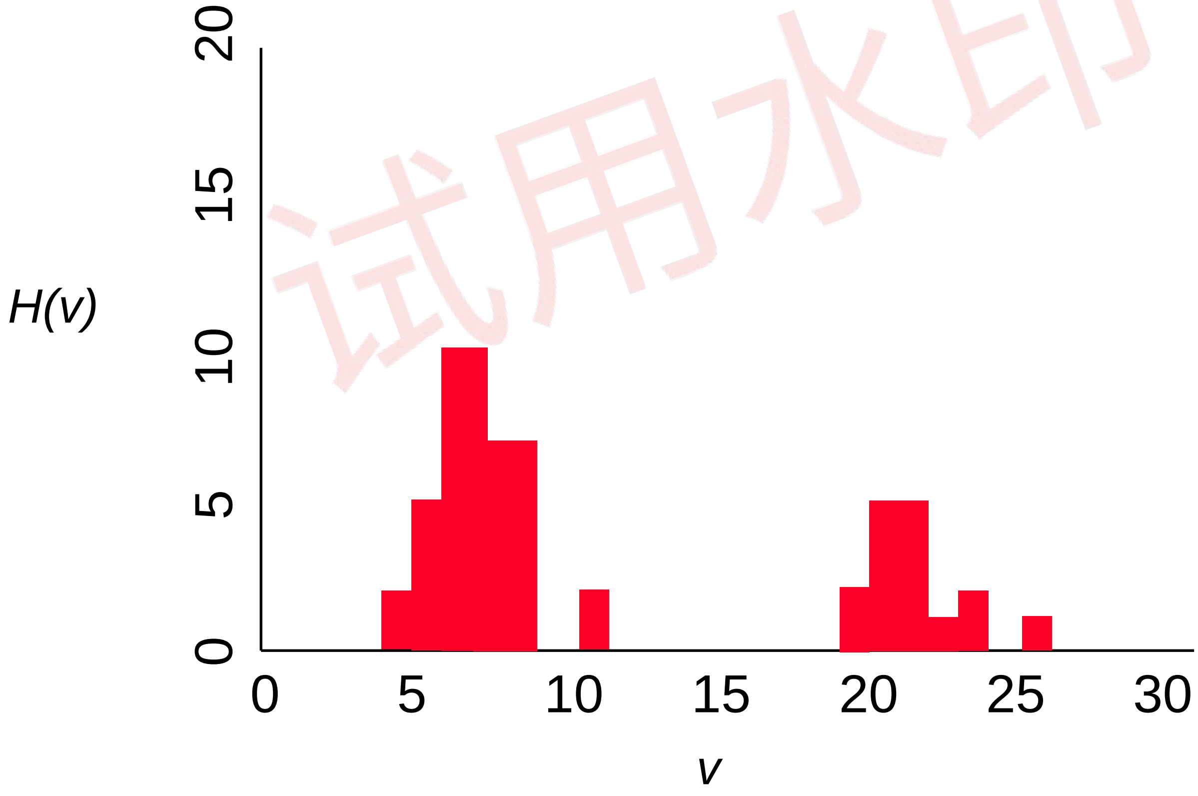
试用水印

Intensity Histograms

试用水印

Grey-level Histogram

Count number of times each grey-level, *v*, occurs in image. Shows *distribution* of values. (like a bar chart!)



试用水印

Statistics from Histogram

❖**Mean:** $$u = \sum_{i=1}^{N} i * h(i)$$

❖**Variance:** $$v = \sum_{i=1}^{N} (i-u)^2 * h(i)$$

❖**Entropy: The information of the histogram. (flatness)** entropy= expected information

$$en(h) = -\sum_{i=1}^{N} h(i)\log(h(i))$$

❖**Modes: the most frequently happened items, i.e., the intensities**

! Histogram needs to be normalised first, .i.e.,

$$\sum_{i=1}^{N} h(i) = 1$$

试用水印

Thresholding

Image intensity thresholding separate pixels into two categories:

❖Larger than or equal a threshold

❖Smaller than a threshold

试用水印

Thresholding creates a binary image, often called binarization, e.g: counting the number of cells in a histological images, number of rice in a visual image.

Binary Threshold

Formula

$$\boxed{B(i,j)=\begin{cases}0, & I(i,j)>=T\\ 1, & otherwise\end{cases}}$$

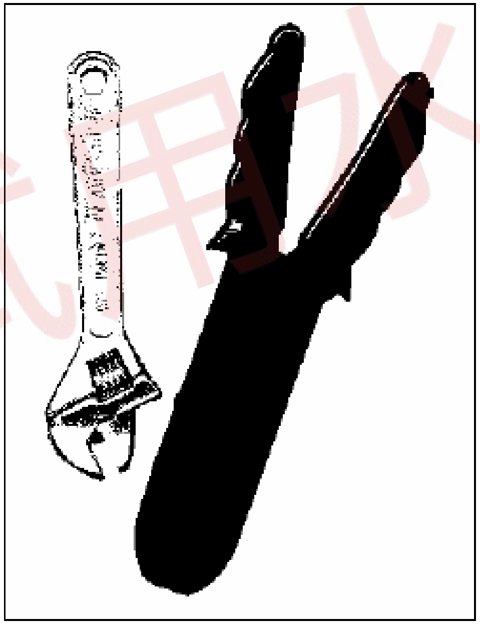
Pseudo Code:

```
for (p=0; p<nPixels; p++)
  if (image[p] < threshold)
    image[p] = 0;
  else image[p]= MAXVAL;
```

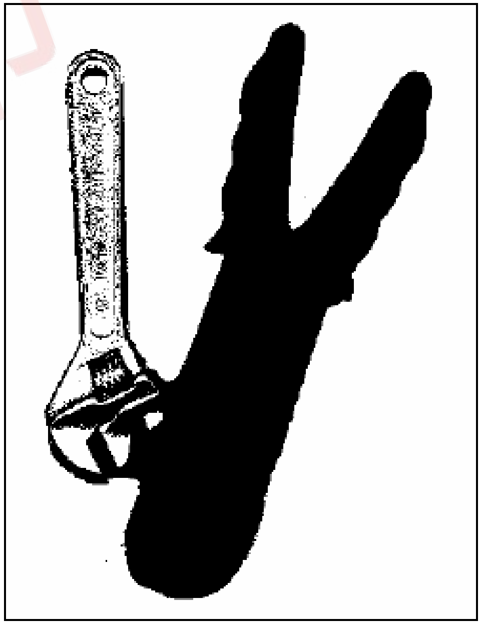
试用水印

Thresholding Examples

Original



Threshold = 50



Threshold = 75

试用水印

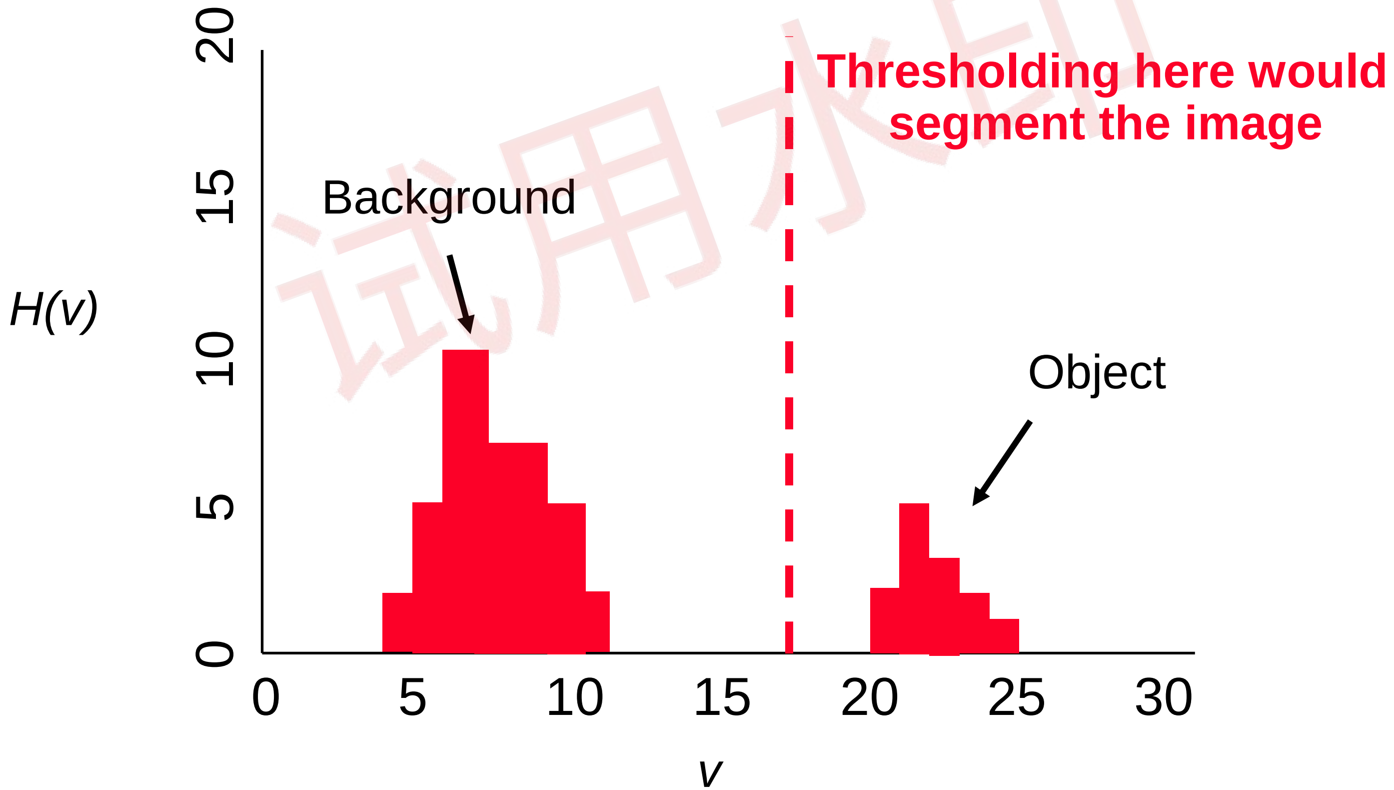
Thresholding

Intensity thresholding usually needs to *analyzing the distribution of the intensity – the histogram*

试用水印

Thresholding: A Bimodal Histogram

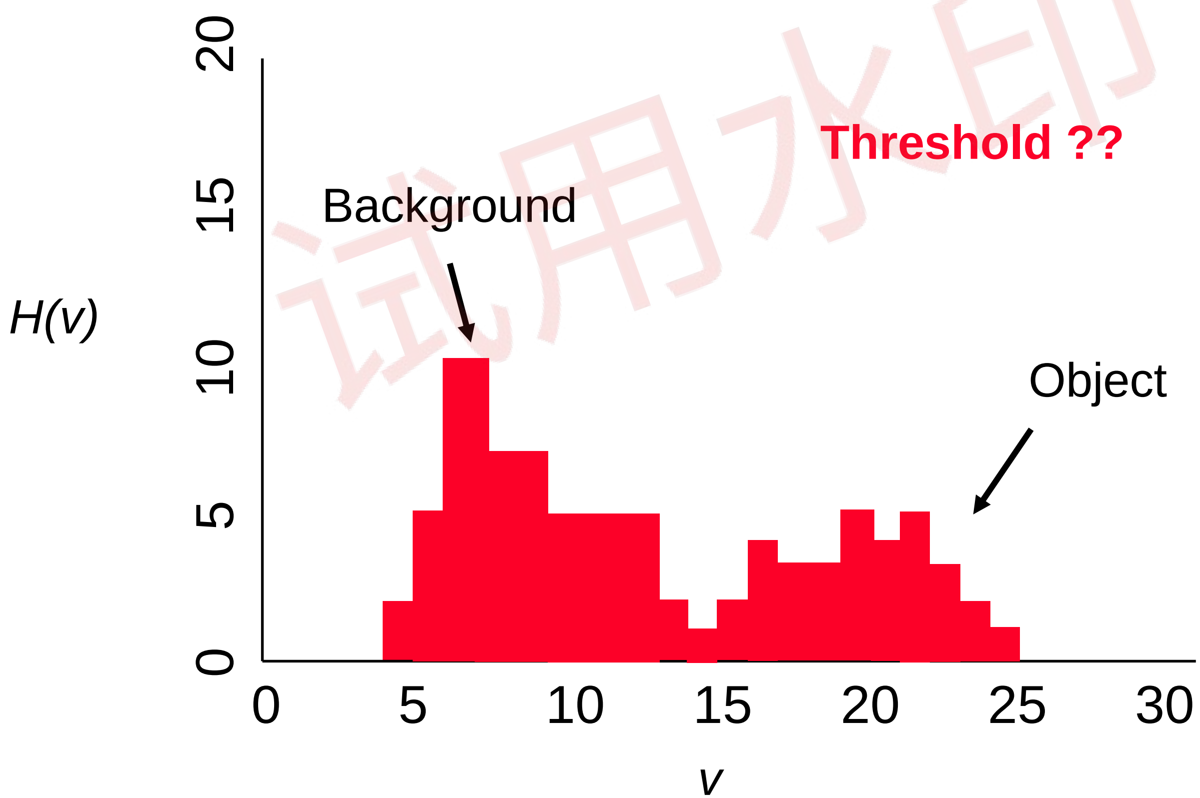
e.g. A bright object on a dark background



试用水印

… but more commonly…

Object and background distributions overlap:



试用水印

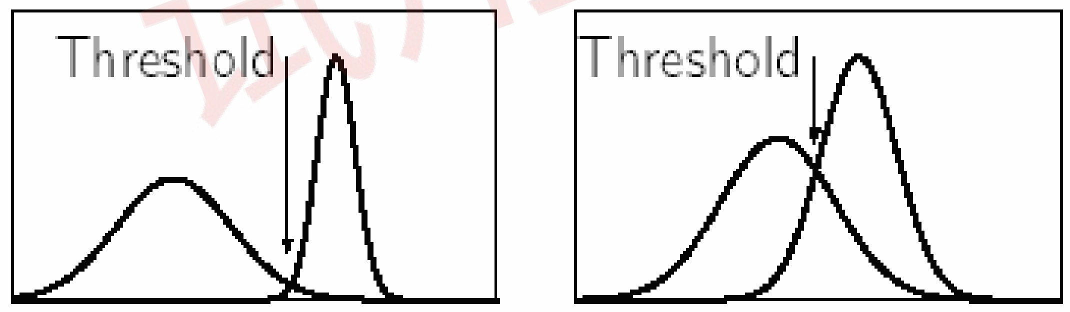
Optimal Thresholding

Histogram shape can be useful in locating the threshold

– However it is not reliable for threshold selection when peaks are not clearly resolved.

– A “flat” object with small intensity variations will give rise to a relatively narrow histogram peak.

试用水印



Normally works well for Bimodal Histogram

Optimal Thresholding

-Usually need to specify a criterion function to measure separation between groups
-- 1) Otsu's method: choose the threshold to minimize the intra-class variance while maximize the inter-class distance (such a criteria is often called Fisher criterion in statistics).
-2) Approximate Ostu's method: Auto Thresholding

Paper: Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms". *IEEE Trans. Sys., Man., Cyber.* **9** (1): 62–66.

Normally works well for Bimodal Histogram

试用水印

Otsu's Method

Otsu's thresholding method is based on selecting the lowest point *between* two *classes* (peaks).

Weight and Mean value

:

Weight: $\omega = \sum_{i=0}^{T} P(i)$ $\quad P(i) = n_i / N$ N: total pixel number

Mean: $\mu = \sum_{i=0}^{T} iP(i) / \omega$ **n_i:** number of pixels in level I

试用水印

Analysis of variance (variance=standard deviation²)

Total variance:

$$\partial_{t}^{2} = \sum_{i=0}^{T} (i - \mu)^2 P(i)$$

Otsu's Method

Inter-classes/between-classes variance (δ_b^2):

The variation of the mean values for each class from the overall intensity mean of all pixels:

$$\delta_b^2 = \omega_0 (\mu_0 - \mu_t)^2 + \omega_1(\mu_1 - \mu_t)^2,$$

Substituting $\mu_t = \omega_0 \mu_0 + \omega_1\mu_1$, we get:

$$\delta_b^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2$$

$\omega_0, \omega_1, \mu_0, \mu_1$ stands for the weights and mean values of two classes, respectively.

试用水印

Otsu's Method

Pseudo Code:

1. Initialize a threshold, normally the starting value (e.g.,0)

Loop -- Repeat:

2. Separate the pixels into two clusters based the threshold
3. Computer the mean value of each cluster
4. Square the difference between the means
5. Multiply the number of pixels on one cluster times the number in the other (the between-class variance) for that threshold
6. Until the every possible threshold being tested (e.g., till 254).
7. Select the threshold that having the largest between class variance

试用水印

Otsu's Method- Matlab

```
I = imread('coins.png');
level = graythresh(I);
BW = im2bw(I,level);
figure, imshow(BW)
```

试用水印

Reading: Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms". *IEEE Trans. Sys., Man., Cyber.* **9** (1): 62–66.

The pdf is available from the VLE!

Auto Thresholding – Approximate to Otsu's Method

Objective: Minimizing the intra-class/within-class variance (= max inter-class variance).

Ostu's method: exhaustive search

Alternatives – approximate search:

1. Select an initial estimate of the threshold *T*. A good initial value is the average intensity of the image.

3. Calculate the mean grey values μ_1 nd μ_2 of the partitions, *R1*, *R2* .

2. Partition the image into two groups, *R1*, *R2* , using the threshold *T*.

4. Select a new threshold:

$$T = \frac{1}{2}(\mu_1 + \mu_2)$$

5. Repeat steps 2-4 until the mean values μ_1 nd μ_2 in successive iterations do not change.

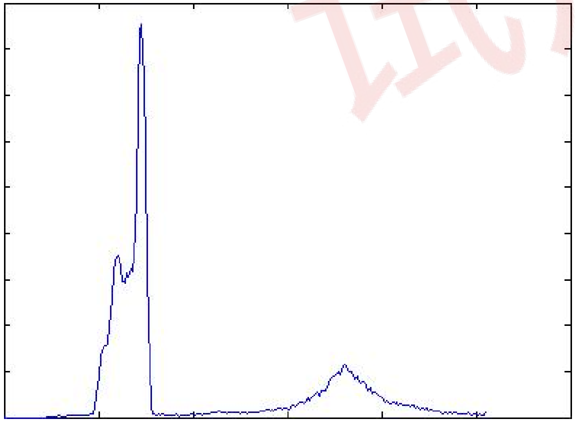
Note: this is essentially a *1-D k-means* algorithm. You will learn the general *k-means* in later lectures

试用水印

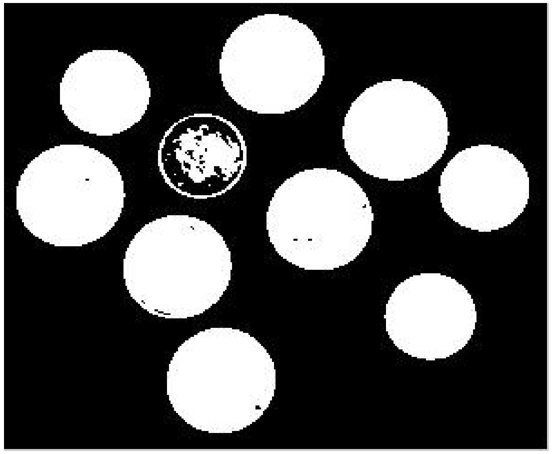
Examples

Coins Image

Threshold at 50



Histogram



Optimal Threshold

试用水印

Change Detection and Background Extraction

试用水印

Motivation: Situational Awareness

- **Having a complete understanding of an environment**
- **Acquiring knowledge of what events happened when:**
  - Is the event a common event?
  - Did a daily event not happened today?
  - Is something abnormal happening?
  - Is the event dangerous?
- **Events that likely can include people and vehicles, happening anytime day or night**

试用水印

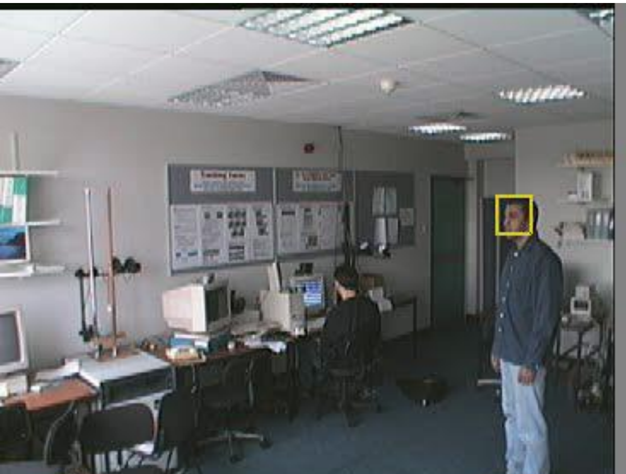
Image Sequences

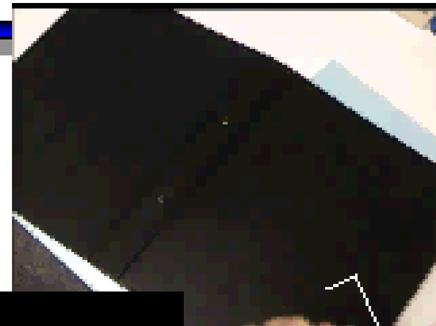
- A typical frame-rate: 25Hz

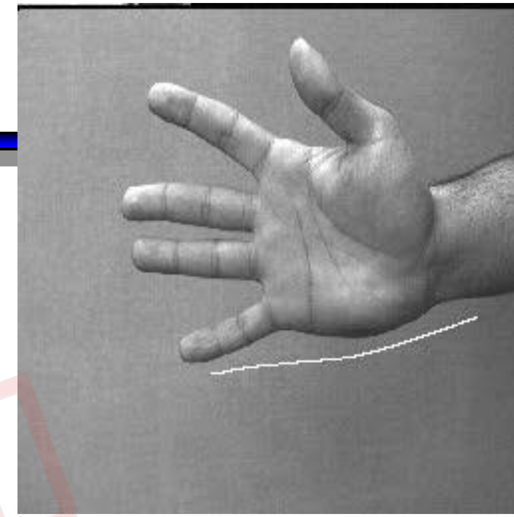
- Notation:
  - An intensity image: $I(x,y)$
  - An image acquired at time t: $I(x,y,t)$
  - An image sequence:

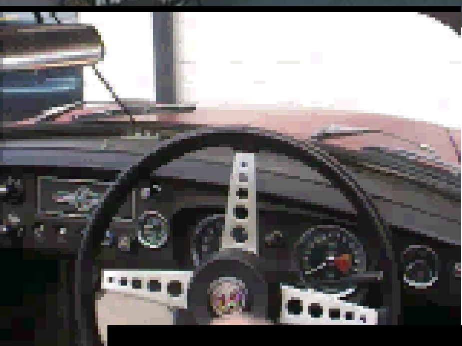
$$I(x,y,0),\ I(x,y,1)\ \ldots\ I(x,y,t-1),\ I(x,y,t)$$

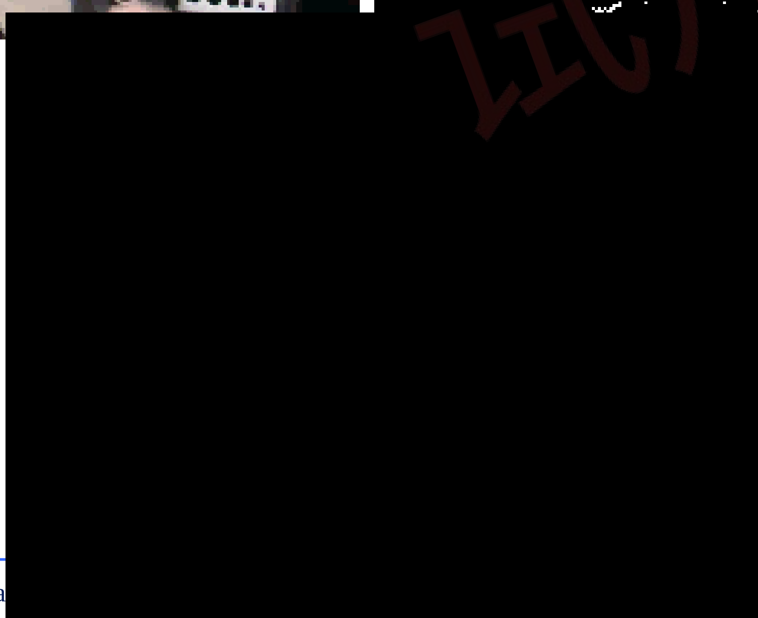
试用水印





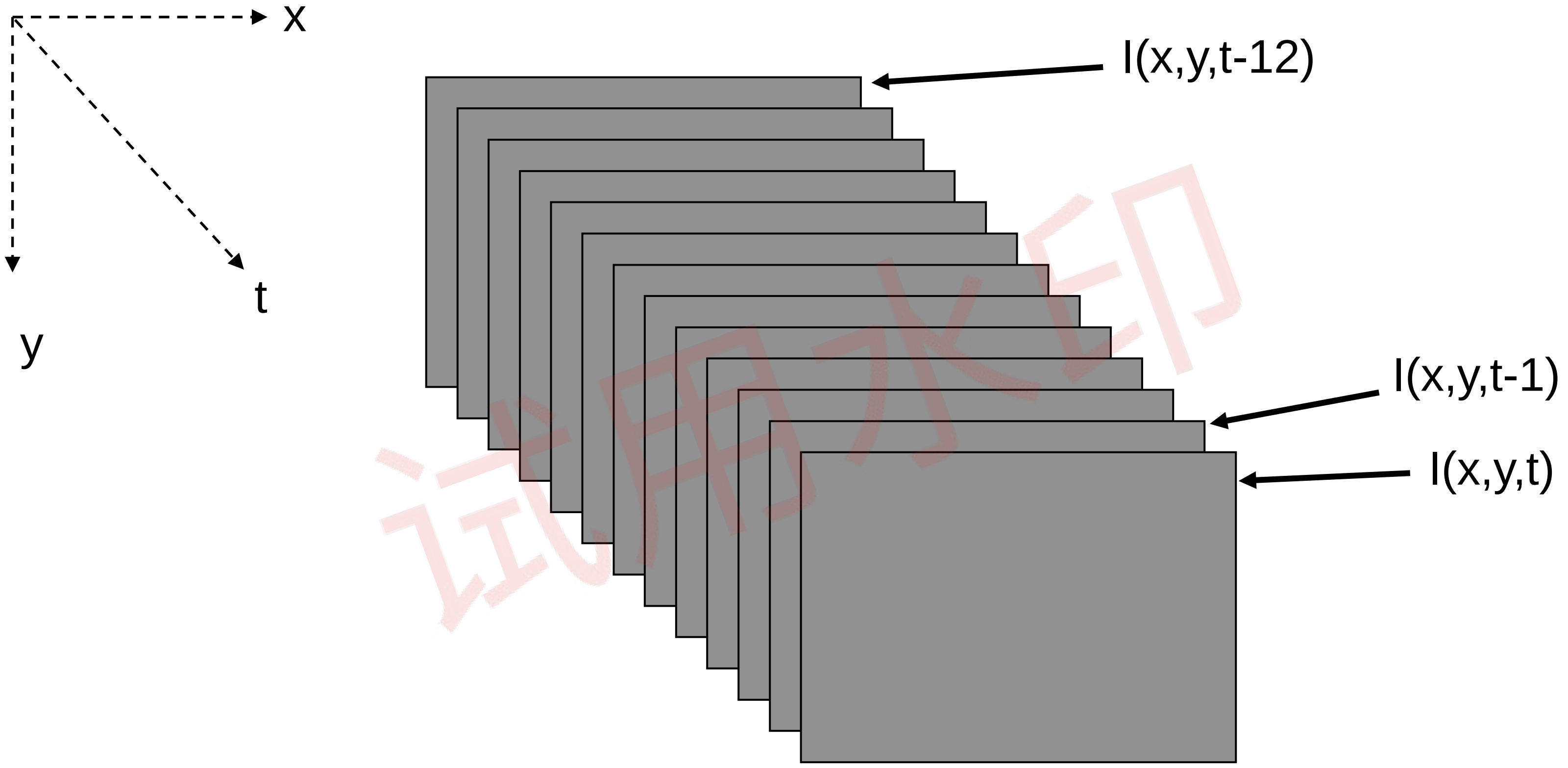






试用水印

Space-time



- Think of sequence as occupying a block of ‘space-time’
- Time is divided into frames like space is divided into pixels

Change Detection:Temporal Difference

- Absolute difference between subsequent images:

$$D(x,y,t) = \left| I(x,y,t) - I(x,y,t-1) \right|$$

Could threshold to give binary change detection image:

$$\begin{aligned} B(x,y,t) &= 1 \quad \text{if} \quad D(x,y,t) \geq \tau \\ B(x,y,t) &= 0 \quad \text{if} \quad D(x,y,t) < \tau \end{aligned}$$

试用水印

I(t-1)

I(t)

D(t)

B(t)

Poor choice of threshold τ

试用水印

Frame Differencing

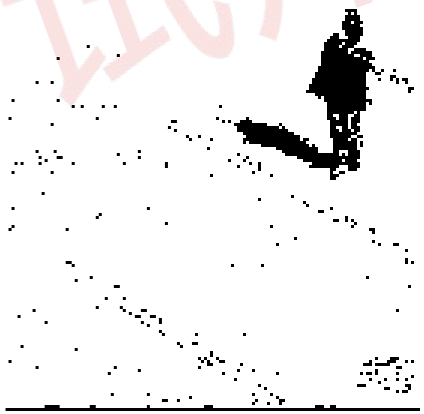
I(0)

I(t)

| I(0) - I(t) |

试用水印

Thresholded:



A better threshold:

From: P.L. Rosin, "Thresholding for Change Detection"
http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/ROSIN2/thresh.html

Background Subtraction

- Detects change in homogenous regions provided different from background

- But background image needs updating

试用水印

Image differencing: pros & cons

- Pros:
  - Fast and simple
  - Useful for focusing visual attention
- Cons:
  - How to set thresholds ? (see Rosin’s article)
  - Detects any change
    - ⇨moving objects
    - ⇨illumination change
    - ⇨moving shadows
    - ⇨camera motion
    - ⇨camera noise (filtering can overcome this)

试用水印

Background Modelling

- Simple Frame Difference against a background – non adaptive
- Moving Average – lower-level adaptation
- Online Gaussian – (not covered, requires further reading)

试用水印

Background Subtraction-Not Adaptive

- Simple Frame Difference against a known background
  - Simple, and easy to implement
  - Not robust
  - Applicable when a camera is fixed, background fixed
  - Can not handle illumination changes well, especially in outdoor scenarios, e.g, sunlight, cloudy, day/night, slight movement of tree leaves.
  - the need for manual initialization. Without re-initialization, errors in the background accumulate over time, making this method useful only in highly-supervised, short-term tracking applications without significant changes in the scene.
- Math Model for foreground extraction
  - B(x,y) = mean_t(v(x,y,t))
  - f(x,y,t) = abs(v(x,y,t)-B(x,y)), and then setting threshold for f

试用水印

Moving Average- Adaptive

- Moving average filtering
  - Current background equals to the mean of previous n frames.
  - Algorithms (pseudo code)

- *Initialise the value of a background estimate B, for each frame F*
  - *Update the background estimate by computing*

$$B(n+1) = (w_a F + \sum_i w(i) B(n-i)) / w_c$$

for a choice of w_a, $w(i)$ *and* $w_c = w_a + \sum_i w(i)$

***Or**:* $$B(n+1) = \sum_i w(i) F(n-i)$$

  - *Substract the background estimate from the frame, and report the value of each pixel where the magnitude of the difference is greater than some threshold*
- *end*

  - Could handle a certain level of background changes and Illumination changes (*ref: Chapter 14, Computer Vision: a Modern Approach, Forsyth and Ponce*)

试用水印

Background Extraction Demo

- Using GMM method
  - Gaussian Mixture
  - ***Landmark Paper***: *Adaptive background* mixture *models* for real-time tracking. Chris Stauffer. W.E.L Grimson, CVPR 2008
  - Watch Video Now

试用水印

Further Reading

- **Further Reading**
  - Chris Stauffer and W.E.L Grimson, *Adaptive background mixture models for real-time tracking*,CVPR08 (landmark paper)
  - Zoran Zivkovic and van der Heijden, *Efficient adaptive density estimation per image pixel for the task of background subtraction*, Pattern Recognition Letter 2005 (Good technique paper)

试用水印

Computer Vision

Pixel Operations, Histogram, Thresholding

试用水印

Learning Outcomes

Be able to understand the basics of point operators
Be able to compute histograms, optimal threshold for a given image by programming
Be able to design and apply simple background extraction algorithms.