

## Images

- Camera models
- Digital images
- Colour images
- Noise
- Smoothing

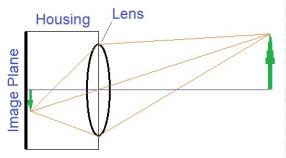
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 1

1

## Camera models

- Components:
  - A photosensitive image plane
  - A housing
  - A lens

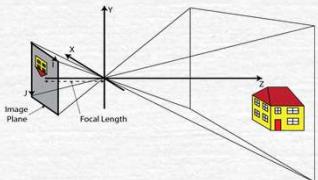


Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 2

2

## Camera models – Simple Pinhole Model

$$\begin{bmatrix} i \cdot w \\ j \cdot w \\ w \end{bmatrix} = \begin{bmatrix} f_i & 0 & c_i \\ 0 & f_j & c_j \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$


- 3-D point  $(x, y, z)$
- 2-D image point  $(i, j)$
- Scaling factor  $w$
- Combination of focal length and image coordinate system  $(f_i \& f_j)$
- Location of the optical centre  $(c_i \& c_j)$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 3

3

## Digital Images

- Theoretically images are continuous 2D functions of reflected scene brightness.
- $(i, j)$  or  $(\text{column}, \text{row})$  or  $(x, y)$

- To process on a computer we need a discrete representation
  - Sample
  - Quantise

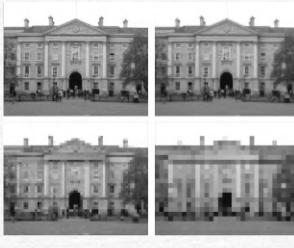
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 4

4

## Digital Images – Sampling

- Sample the continuous 2D function into discrete elements.
- Sensor
  - 2D array
  - Photosensitive elements
  - Non photosensitive gaps
- Issues
  - Elements have a fixed area
  - Gaps



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 5

5

## Digital Images – Sampling

- How many samples do we need ?
  - Wasted space and computation time
  - Enough for the objects of interest



```
Mat image, smaller_image;
resize( image, smaller_image,
Size( image1.cols/2, image1.rows/2 ));
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 6

6

## Digital Images – Quantisation

- Represent the individual image points as digital values.
  - Typically 8 bits



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images

Slide 7

7

## Digital Images – Quantisation

- How many bits do we need?
  - Wasted space ?
  - Losing the ability to distinguish objects



```
void ChangeQuantisationGrey( Mat &image, int num_bits )
{
    CV_Assert( (image.type() == CV_8UC1) && (num_bits >= 1) &&
               (num_bits <= 8) );
    uchar mask = 0xFF << (8-num_bits);
    for (int row=0; row < image.rows; row++)
        for (int col=0; col < image.cols; col++)
            image.at<uchar>(row,col) = image.at<uchar>(row,col) & mask;
}
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images

Slide 8

8

## Colour Images

- Luminance only
  - Simple representation
  - Humans can understand



- Colour images ( luminance + chrominance )
  - Multiple channels (typically 3)
  - Around 16.8 million colours
  - More complex to process
  - Facilitate certain operations



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images

Slide 9

9

## Colour Images – Processing

```
void InvertColour( Mat& input_image, Mat& output_image )
{
    CV_Assert( input_image.type() == CV_8UC3 );
    output_image = input_image.clone();
    for (int row=0; row < input_image.rows; row++)
        for (int col=0; col < input_image.cols; col++)
            for (int channel=0; channel <
                 input_image.channels(); channel++)
                output_image.at<Vec3b>(row,col)[channel] = 255 -
                input_image.at<Vec3b>(row,col)[channel];
}
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images

Slide 10

10

## Colour Images – Efficient processing

```
int image_rows = image.rows;
int image_columns = image.cols;
for (int row=0; row < image_rows; row++) {
    uchar* value = image.ptr<uchar>(row);
    uchar* result_value = result_image.ptr<uchar>(row);
    for (int column=0; column < image_columns; column++) {
        *result_value++ = *value++ ^ 0xFF;
        *result_value++ = *value++ ^ 0xFF;
        *result_value++ = *value++ ^ 0xFF;
    }
}
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

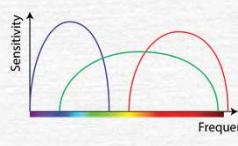
Images

Slide 11

11

## Colour Images – RGB Images

- Red-Green-Blue images
  - Most common
  - Channels correspond roughly to
    - Red (700nm)
    - Green (546nm)
    - Blue (436nm)
  - Channels combined in display



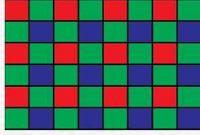

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images

Slide 12

2

## Colour Images – RGB Images

- Converting to Greyscale
  - $Y = 0.299R + 0.587G + 0.114B$
- Camera photosensitive elements
  - Separate Red, Green & Blue elements
  - Sometimes sensitive to all visible wavelengths
  - Bayer pattern:

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images

Slide 13

13

## Colour Images – RGB Images

```
Mat bgr_image, grey_image;
cvtColor(bgr_image, grey_image, CV_BGR2GRAY);
vector<Mat> bgr_images(3);
split(bgr_image, bgr_images);
Mat& blue_image = bgr_images[0];
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images

Slide 14

14

## Colour Images – CMY Images

- Cyan-Magenta-Yellow images
  - Secondary colours
  - Subtractive colour scheme
    - $C = 255 - R$
    - $M = 255 - G$
    - $Y = 255 - B$
  - Often used in printers



CMY is not directly supported in OpenCV.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

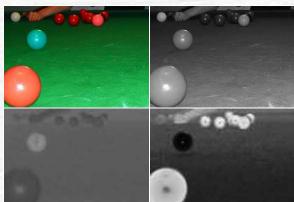
Images

Slide 15

15

## Colour Images – YUV Images

- Used for analogue television signals
  - PAL, NTSC
  - 4 Y to 1 U to 1 V
- Conversion from RGB
  - $Y = 0.299R + 0.587G + 0.114B$
  - $U = 0.492 * (B - Y)$
  - $V = 0.877 * (R - Y)$



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

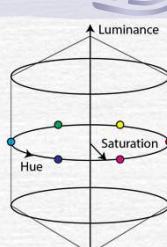
Images

Slide 16

16

## Colour Images – HLS Images

- Hue-Luminance-Saturation images
  - Separates Luminance & Chrominance
  - Values we humans can relate to...
    - Hue 0°..360°
    - Luminance 0..1
    - Saturation 0..1
  - Watch out for circular Hue...



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images

Slide 17

17

## Colour Images – HLS Images

- Conversion from RGB
 
$$L = \frac{\max(R, G, B) + \min(R, G, B)}{2}$$

$$S = \begin{cases} \frac{\max(R, G, B) - \min(R, G, B)}{\max(R, G, B) + \min(R, G, B)} & \text{if } L < 0.5 \\ \frac{\max(R, G, B) - \min(R, G, B)}{2 - (\max(R, G, B) + \min(R, G, B))} & \text{if } L \geq 0.5 \end{cases}$$

$$H = \begin{cases} 60 \cdot (G - B) / S & \text{if } R = \max(R, G, B) \\ 120 + 60 \cdot (B - R) / S & \text{if } G = \max(R, G, B) \\ 240 + 60 \cdot (R - G) / S & \text{if } B = \max(R, G, B) \end{cases}$$

```
cvtColor(bgr_image, hls_image, CV_BGR2HLS);
// Hue ranges from 0 to 179.
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images

Slide 18

## Colour Images – Other colour spaces

- HSV
- YCrCb
- CIE XYZ
- CIE L\*u\*v\*
- CIE L\*a\*b\*
- Bayer

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 19

19

## Noise

- Affects most images
- Degrades the image
- Can cause problems with processing
- Causes?
- Measuring noise:  $S/N \text{ ratio} = \frac{\sum_{(i,j)} f^2(i,j)}{\sum_{(i,j)} v^2(i,j)}$
- Types
  - Gaussian
  - Salt and Pepper
- Correcting noise...

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 20

20

## Noise – Salt and Pepper Noise

- Impulse noise
- Noise is maximum or minimum values

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 21

21

## Noise – Gaussian Noise

- Good approximation to real noise
- Distribution is Gaussian (mean & s.d.)

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 22

22

## Smoothing

- Removing or reducing noise...
- Linear & non-linear transformations

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 23

23

## Smoothing – Averaging Filters (linear)

- Linear transformation (convolution)
- Local neighbourhood
  - $f(i,j) = \sum \sum h(i-m, j-n).g(m,n)$
  - Different masks...
    - Local Average
    - Gaussian
- Acceptable results?
  - Suppression of (small) image noise
  - Blurring of edges

```
blur(image,smoothed_image,Size(3,3));
GaussianBlur(image,smoothed_image,Size(5,5),1.5);
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images Slide 24

24

## Smoothing – Averaging filter examples

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images      Slide 25

25

## Smoothing – Median Filter (non-linear)

- Use the median value... 11 18 20 21 23 25 25 30 250
- Median = 23 Average = 47
- Not affected by noise
- Doesn't blur edges much
- Can be applied iteratively

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images      Slide 26

26

## Smoothing – Median Filter (non-linear)

- Damages thin lines and sharp corners
  - Change region shape
- Computational expensive
  - Standard –  $O(r^2 \log r)$
  - Huang –  $O(r)$
  - Perreault (2007) –  $O(1)$

```
medianBlur(image, smoothed_image, 5);
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images      Slide 27

27

## Smoothing – Effects of mask size

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images      Slide 28

28

## Image Pyramids

- To process images
  - At multiple scales
  - Efficiently
- Technique
  - Smooth image (often Gaussian)
  - Subsample (usually by a factor of 2)

```
pyrDown(image, smaller_image,
        Size( (image1.cols+1)/2, (image1.rows+1)/2));
```

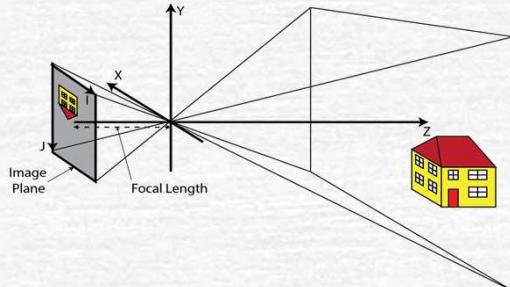
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Images      Slide 29

29

## Camera models – Simple Pinhole Model

$$\begin{bmatrix} i \cdot w \\ j \cdot w \\ w \end{bmatrix} = \begin{bmatrix} f_i & 0 & c_i \\ 0 & f_j & c_j \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



- ⌚ 3-D point  $(x, y, z)$
- ⌚ 2-D image point  $(i, j)$
- ⌚ Scaling factor  $w$
- ⌚ Combination of focal length and image coordinate system ( $f_i$  &  $f_j$ )
- ⌚ Location of the optical centre ( $c_i$  &  $c_j$ )

Images

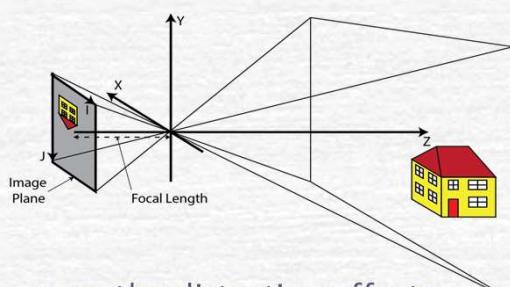
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 3

3

## Camera models – Simple Pinhole Model

$$\begin{bmatrix} i \cdot w \\ j \cdot w \\ w \end{bmatrix} = \begin{bmatrix} f_i & 0 & c_i \\ 0 & f_j & c_j \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



- ⌚ This is what OpenCV uses once the distortion effects are removed.
- ⌚  $i \cdot w = f_i \cdot x + c_i \cdot z \quad w = z$
- ⌚  $i = f_i \cdot (x/z) + c_i$
- ⌚  $j = f_j \cdot (y/z) + c_j$

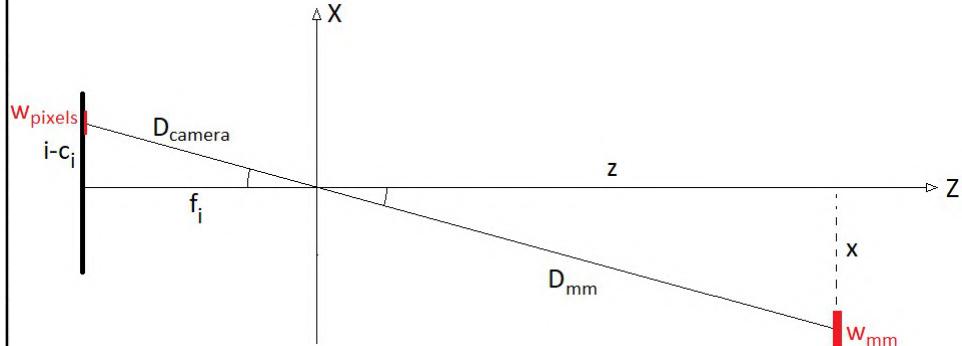
Images

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 4

4

## Camera models – Distance from the camera



- Just considering the  $i$  and  $X$  plane, if we have an object of a known width (note it is a little more complex when  $j$  and  $Y$  are introduced)
- $D_{camera}$  can be computed using Pythagoras for right angled triangles
- $D_{mm} / W_{mm} = D_{camera} / W_{pixels}$
- So  $D_{mm} = D_{camera} * W_{mm} / W_{pixels}$

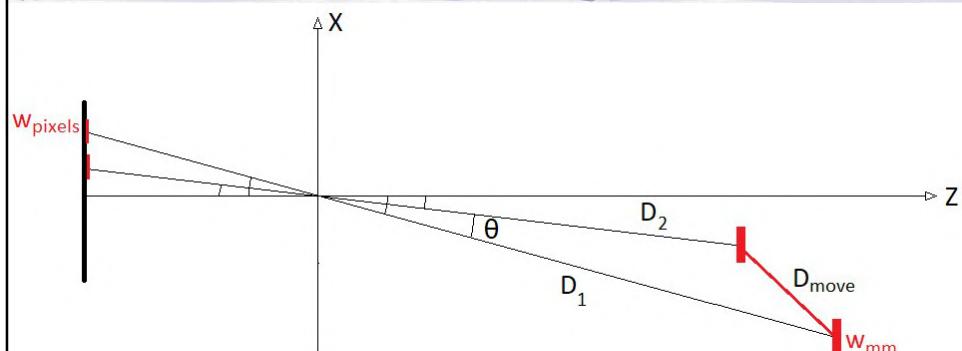
Images

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 5

5

## Camera models – Distance travelled



- We can compute the two distances
- We can compute theta (the difference of the two angles)
- We can compute the distance moved using the **law of cosines**

Images

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 6

6

## Binary

- Thresholding
- Threshold detection
- Variations
- Mathematical Morphology

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 1

1

## Thresholding

- Binary thresholding
  - for all pixels  
 $g(i,j) = 1 \text{ for } f(i,j) \geq T$   
 $= 0 \text{ for } f(i,j) < T$
- Simple scenes?
- Look Up Table
  - for all grey levels  
 $LUT(k) = 1 \text{ for } k \geq T$   
 $= 0 \text{ for } k < T$
- Objects of interest vs. background

```
threshold(gray_image, binary_image, threshold, 255, THRESH_BINARY);
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 2

2

## Thresholding

- Distinct foreground & background needed

How do we determine the best threshold?

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 3

3

## Threshold Detection

- Manual Setting
  - Issue of changing lighting
- Need to determine automatically

For the techniques which follow:

- Image  $f(i,j)$
- Histogram  $- h(g)$  of values in  $f(i,j)$
- Normalised Probability Distribution  $- p(g) = h(g) / \sum_g h(g)$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 4

4

## Threshold Detection – Otsu Thresholding

- Minimize the spread of the pixels...
  - Smallest within class variance

$$\sigma_w^2(T) = w_f(T)\sigma_f^2(T) + w_b(T)\sigma_b^2(T)$$

$$w_f(T) = \sum_{g=0}^{255} p(g) \quad \sigma_f^2(T) = \frac{\sum_{g=0}^{255} p(g) \cdot (g - \mu_f(T))^2}{w_f(T)}$$

$$w_b(T) = \sum_{g=0}^{T-1} p(g) \quad \sigma_b^2(T) = \frac{\sum_{g=0}^{T-1} p(g) \cdot (g - \mu_b(T))^2}{w_b(T)}$$

$$\mu_f(T) = \frac{\sum_{g=0}^{255} p(g) \cdot g}{w_f(T)} \quad \mu_b(T) = \frac{\sum_{g=0}^{T-1} p(g) \cdot g}{w_b(T)}$$

- Largest between class variance

$$\sigma_B^2(T) = w_f(T)w_b(T)(\mu_f(T) - \mu_b(T))^2$$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 5

5

## Threshold Detection – Otsu Thresholding

```
threshold(gray_image, binary_image, threshold, 255, THRESH_BINARY | THRESH_OTSU);
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 6

1

## Variations – Adaptive Thresholding

- The adaptive thresholding algorithm is
  - Divide the image into sub-images,
  - Compute thresholds for all sub-images,
  - Interpolate thresholds for every point using bilinear interpolation.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Binary

Slide 7

7

## Variations – Adaptive Thresholding

- OpenCV version:

```
if ((f(i,j) - (Σa=-m..m, b=-m..m f(i+a,j+b)) / (2m+1)2) > offset)
g(i,j) = 255
else g(i,j) = 0
```

`adaptiveThreshold( gray_image,binary_image,output_value,  
ADAPTIVE_THRESH_MEAN_C,THRESH_BINARY,  
block_size,offset );`

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Binary

Slide 8

8

## Variations – Multi-Spectral Thresholding

MRI scan (3D GRE) (top) and histogram (bottom)

© Guido Gerig, Univ. of North Carolina (Reproduced with permission)

Automatic Estimation of parameters for wm, gm and subcortical structures (Levenberg Marquardt).

Fuzzy Classification of wm, gm, subcort.

WM Classification

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Binary

Slide 9

9

## Cleaning Binary Images

- Cannot use normal smoothing operations
- Need to remove noisy points and smooth binary region boundaries.
- Mostly we use operations originally defined as part of “mathematical morphology” (which treats images as sets).
- Most common operations:
  - Erosion
  - Dilation
  - Opening
  - Closing

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Binary

Slide 10

10

## Mathematical Morphology – Dilation

- Minkowski set addition:
  - $X \oplus B = \{ p \in e^2; p = x+b, x \in X \text{ and } b \in B \}$
  - B is the structuring element
  - B is typically isotropic
- Adds pixels around borders
- Effects:
  - Makes ‘regions’ bigger
  - Fills small holes
  - Joins close ‘regions’

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Binary

Slide 11

11

## Mathematical Morphology – Erosion

- Minkowski set subtraction:
  - $X \ominus B = \{ p \in e^2; p+b \in X \text{ for every } b \in B \}$
  - Again B is typically isotropic
  - Removes border pixels
- Effects:
  - Makes ‘regions’ smaller
  - Removes noise
  - Removes narrow bridges

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Binary

Slide 12

2

## Mathematical Morphology –

**Opening:**  $X \circ D = (X \ominus D) \oplus D$

- Removes noise
- Removes narrow bridges
- Roughly maintains 'region' size
- Smooths shape

**Closing:**  $X \bullet D = (X \oplus D) \ominus D$

- Fills small holes
- Joins close 'regions'
- Roughly maintains 'region' size

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 13

13

## Mathematical Morphology – Application

**Opening:**  $X \circ D = (X \ominus D) \oplus D$

**Closing:**  $X \bullet D = (X \oplus D) \ominus D$

**Isotropic structuring element:**

- Eliminates small image details

**Properties**

- $X \circ D = (X \ominus D) \oplus D$  and  $X \bullet D = (X \oplus D) \ominus D$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 14

14

## Mathematical Morphology – OpenCV Code

```
dilate( binary_image, dilated_image, Mat());
Mat structuring_element( 5, 5, CV_8U, Scalar(1) );
dilate( binary_image, dilated_image, structuring_element);

erode( binary_image, eroded_image, Mat());
Mat structuring_element( 5, 5, CV_8U, Scalar(1) );
erode( binary_image, eroded_image, structuring_element);

Mat five_by_five_element(5, 5, CV_8U, Scalar(1));
morphologyEx( binary_image, opened_image,
    MORPH_OPEN, five_by_five_element );
morphologyEx( binary_image, closed_image,
    MORPH_CLOSE, five_by_five_element );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 15

15

## Mathematical Morphology – Greyscale / Colour

**Grey scale image**

- One set per grey level ( $g$ )
- All points  $\geq g$
- Operations on each set separately

**Colour**

- One set per level per channel
- e.g. RGB image has  $255 \times 3$  sets.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 16

16

## Mathematical Morphology – Local maxima

Can be used to locate local maxima and minima

- Dilate & compare to the original image (to find local maxima)
- Threshold the original image
- Logical AND the results together to find 'high' local maxima

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 17

17

## Mathematical Morphology – Local maxima

```
Mat dilated, thresholded_input, local_maxima, thresholded_8bit;
dilate( input, dilated, Mat());
compare( input, dilated, local_maxima, CMP_EQ );
threshold( input, thresholded_input, threshold, 255, THRESH_BINARY );
thresholded_input.convertTo( thresholded_8bit, CV_8U );
bitwise_and( local_maxima, thresholded_8bit, local_maxima );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 18

## Histograms

- 1D Histograms
- 3D Histograms
- Equalisation
- Histogram Comparison
- Back Projection

Histograms

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 1

1

## 1D Histograms

Determine the frequency of brightness values

- Initialise:  
•  $h(z) = 0$  for all values of  $z$
- Compute:  
• For all pixels  $(i,j)$ :  $h(f(i,j))++$

Histograms

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 2

2

## 1D Histograms

Is this useful?

- Global information
- Useful for classification ?
- Not unique

Histograms

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 3

3

## 1D Histograms

```
Mat display_image;
MatND gray_histogram;
const int* channel_numbers = { 0 };
float channel_range[] = { 0.0, 255.0 };
const float* channel_ranges = channel_range;
int number_bins = 256;
calcHist( &gray_image, 1, channel_numbers, Mat(),
          gray_histogram, 1, &number_bins, &channel_ranges );
OneDHistogram::Draw1DHistogram( &gray_histogram, 1,
                                 display_image );
imshow("Greyscale histogram", display_image);
```

Histograms

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 4

4

## 1D Histograms – Smoothing

- Local minima and maxima are useful.
- But how can we deal with noise though?

10	20	30	25	20	40	50	40	20	20
?	20	25	25	28	37	43	37	27	?

- Smooth the histogram...
- For all values  $v$ :  $h_{\text{new}}(v) = (h(v-1) + h(v) + h(v+1)) / 3$
- What do we do at the ends?
- Do not compute OR Wraparound OR Duplicate values  
OR Reflect values OR Constant values ??

Histograms

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 5

5

## 1D Histograms – Smoothing in OpenCV

10	20	30	25	20	40	50	40	20	20
?	20	25	25	28	37	43	37	27	?

```
MatND smoothed_histogram = histogram[channel].clone();
for(int i = 1; i < histogram[channel].rows - 1; ++i)
    smoothed_histogram[channel].at<float>(i) =
        (histogram.at<float>(i-1) + histogram.at<float>(i) +
         histogram.at<float>(i+1)) / 3;
```

Histograms

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 6

6

## 1D Histograms – Colour Histograms

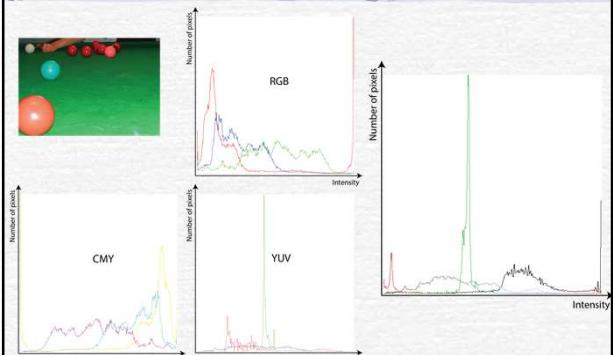
- Determine histograms for each channel independently...
- Choice of colour space...

Histograms

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 7

## 1D Histograms – Colour Histograms



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 8

7

8

## Colour (1D) Histograms in OpenCV

```
MatND* histogram = new MatND[image.channels()];
vector<Mat> channels(image.channels());
split( image, channels );
const int* channel_numbers = { 0 };
float channel_range[] = { 0.0, 255.0 };
const float* channel_ranges = channel_range;
int number_bins = 64;
for (int chan=0; chan < image.channels(); chan++)
    calcHist( &(channels[chan]), 1, channel_numbers, Mat(),
              histogram[chan], 1, &number_bins, &channel_ranges );
```

Histograms

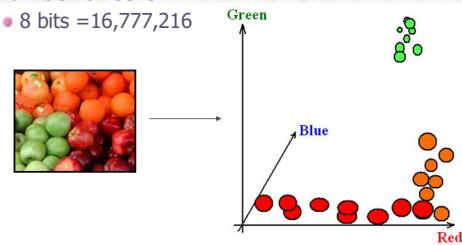
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 9

9

## 3D Histograms

- Channels are not independent
- Better discrimination comes from considering all channels simultaneously
- Number of cells?



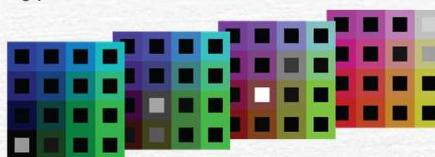
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 10

10

## 3D Histograms

- Reduce quantisation
  - 6 bits = 262,144
  - 4 bits = 4,096
  - 2 bits = 64



Histograms

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 11

11

## 3D Histograms

```
MatND histogram;
int channel_numbers[] = { 0, 1, 2 };
int* number_bins = new int[image.channels()];
for (ch=0; ch < image.channels(); ch++)
    number_bins[ch] = 16;
float ch_range[] = { 0.0, 255.0 };
const float* channel_ranges[] = { ch_range, ch_range, ch_range };
calcHist( &image, 1, channel_numbers, Mat(), histogram,
          image.channels(), a_number_bins, channel_ranges );
```

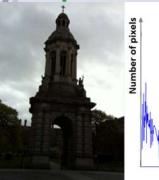
Histograms

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 12

12

## Equalisation



A histogram showing the distribution of luminance values. The x-axis is labeled 'Luminance' and the y-axis is labeled 'Number of pixels'. The distribution is highly skewed towards lower luminance values.

If an image has insufficient contrast...

- Human can distinguish 700-900 greyscales
- How can we improve the contrast?
  - Evenly distribute values on all channels?
  - What happens to colours?

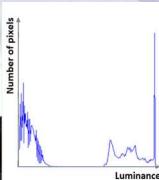
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 13

## Equalisation



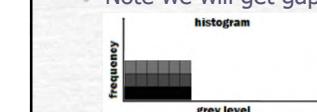
A histogram showing the distribution of luminance values. The x-axis is labeled 'Luminance' and the y-axis is labeled 'Number of pixels'. The distribution is highly skewed towards lower luminance values.



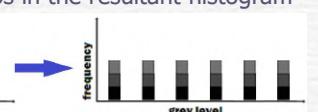
A histogram showing the distribution of luminance values. The x-axis is labeled 'Luminance' and the y-axis is labeled 'Number of pixels'. The distribution is more uniform across the range of luminance values.

Normally equalise only the greyscales / luminance

Note we will get gaps in the resultant histogram



A histogram showing the frequency of each grey level. The x-axis is labeled 'grey level' and the y-axis is labeled 'frequency'. The distribution is uniform across all grey levels.



A histogram showing the frequency of each grey level. The x-axis is labeled 'grey level' and the y-axis is labeled 'frequency'. The distribution is uniform across all grey levels.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 14

13

14

## Equalisation

```
// Create a lookup table to map the luminances
// h[x] = histogram of luminance values image f(i,j).
pixels_so_far = 0
num_pixels = image.rows * image.cols

for input = 0 to 255
  pixels_so_far = pixels_so_far + h[ input ]
  LUT[ input ] = (pixels_so_far*256) / (num_pixels+1)

// Apply the lookup table LUT(x) to the image:
for every pixel f(i,j)
  g(i,j) = LUT[ f(i,j) ]
```

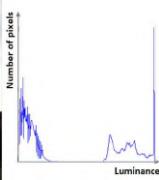
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 15

## Equalisation



A histogram showing the distribution of luminance values. The x-axis is labeled 'Luminance' and the y-axis is labeled 'Number of pixels'. The distribution is highly skewed towards lower luminance values.



A histogram showing the distribution of luminance values. The x-axis is labeled 'Luminance' and the y-axis is labeled 'Number of pixels'. The distribution is more uniform across the range of luminance values.

```
vector<Mat> channels( hls_image.channels() );
split(hls_image, channels);
equalizeHist( channels[1], channels[1] );
merge( channels, hls_image );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 16

15

16

## Histogram Comparison

- To find similar images...
  - Use metadata
  - Compare images?
- One simple possibility is to compare the colour distributions
  - Need a metric for comparisons...

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 17

## Histogram Comparison



Four images with their respective histograms below them. The histograms show the distribution of color channels (Red, Green, Blue, and Alpha) for each image. The images are: a building with a red car, a building with a blue sky, a brick wall, and a golden pavilion.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 18

17

18

## Histogram Comparison – Metrics

- $D_{Correlation}(h_1, h_2) = \frac{\sum_i(h_1(i)-\bar{h}_1)(h_2(i)-\bar{h}_2)}{\sqrt{\sum_i(h_1(i)-\bar{h}_1)^2} \sqrt{\sum_i(h_2(i)-\bar{h}_2)^2}}$
- $D_{Chi-Square}(h_1, h_2) = \sum_i \frac{(h_1(i)-h_2(i))^2}{(h_1(i)+h_2(i))}$
- $D_{Intersection}(h_1, h_2) = \sum_i \min(h_1(i), h_2(i))$
- $D_{Bhattacharyya}(h_1, h_2) = \sqrt{1 - \frac{1}{\sqrt{h_1.h_2.N^2}} \sum_i \sqrt{h_1(i).h_2(i)}}$
- where
  - $N$  is the number of bins in the histograms,
  - $\bar{h}_k = \sum_i(h_k(i))/N$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 19

## Histogram Comparison – Earth Mover

- An alternative to the metrics is to compute the Earth Mover's Distance...
  - Minimum cost for turning a distribution into another distribution
- 1D solution:
  - $EMD(-1) = 0$
  - $EMD(i) = \text{histogram}_1(i) + EMD(i-1) - \text{histogram}_2(i)$
  - Earth Mover's Distance =  $\sum_i |EMD(i)|$
- Colour EMD is harder to compute...

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 20

19

20

## Histogram Comparison

```
normalize( histogram1, histogram1, 1.0);
normalize( histogram2, histogram2, 1.0);
double matching_score = compareHist( histogram1,
                                      histogram2, CV_COMP_CORREL);

We can also use Chi-Square (CV_COMP_CHISQR),
Intersection (CV_COMP_INTERSECT) or
Bhattacharyya (CV_COMP_BHATTACHARYYA) metrics or
alternatively can use the Earth Mover's Distance function
(EMD())
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 21

## Histogram Back Projection

- An approach to selecting colours (based on samples):
  - Obtain a representative sample set of the colours.
  - Histogram those samples.
  - Normalize that histogram so that the maximum value is 1.0.
  - Back project the normalized histogram onto any image  $f(i,j)$ .
  - This will provide a 'probability' image  $p(i,j)$  which indicates the similarity between  $f(i,j)$  and the sample set.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 22

21

22

## Histogram Back Projection in OpenCV

```
calcHist( &hls_samples_image, 1, channel_numbers, Mat(),
          histogram,image.channels(),number_bins,channel_ranges);
normalize( histogram, histogram, 1.0);
Mat probabilities = histogram.BackProject( hls_image );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 23

## Region Segmentation

- Binary Regions
- k Means clustering
- Watershed Segmentation
- Mean Shift Segmentation

Regions

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 1

1

## Segmentation

- Split images into smaller parts
  - Preferably corresponding to (parts of) objects.

- Two main approaches

- Region based
  - Edge based

- Video segmentation

- Breaking video into clips
  - Segmenting objects/regions consistently over time

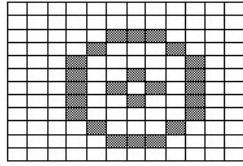
Regions

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 2

2

## Connectivity – Paradoxes



- Use pixel Adjacency to build contiguous regions
  - Objects
  - Background
  - Holes

Regions

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 3

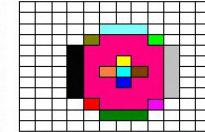
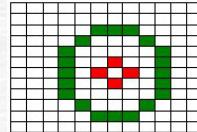
3

## Connectivity – 4 adjacency & 8 adjacency

- Have to use either 4-adjacency or 8-adjacency
  - Label each non-zero pixel...

3	2	1
4	8	0
5	6	7

3	2	1
4	8	0
5	6	7



Regions

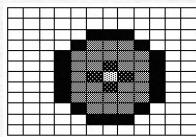
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 4

4

## Connectivity –

- What do we actually want?



- One possibility

- Treat background using 4-adjacency
  - Treat object using 8-adjacency
  - Treat holes using 4-adjacency
  - Treat objects in holes using 8-adjacency
  - ...

Regions

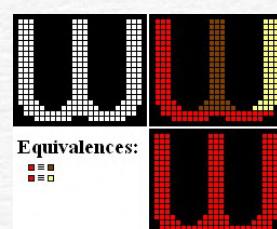
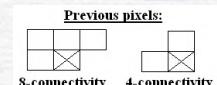
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 5

5

## Connectivity – Connected Components Analysis

- Search image row by row
  - Label each non-zero pixel
  - If previous pixels are all background
    - Assign New Label
  - Otherwise
    - Pick any label from the previous pixels
    - If any of the other previous pixels have a different label
      - Note equivalence
- Relabel equivalent labels.



Equivalences:



Regions

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 6

6

1

## Connectivity – Extracting regions

```
00000000000000000000
JJJJJJJJJJJJJJJJJJJJ
WWWWWWWWWWWWWWWWWW
BABABABABABABABABAB
```

```
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
findContours( binary_image, contours, hierarchy,
CV_RETR_TREE, CV_CHAIN_APPROX_NONE );
```

Regions Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 7

## Connectivity – Labelling regions

```
00000000000000000000
JJJJJJJJJJJJJJJJJJJJ
WWWWWWWWWWWWWWWWWW
BABABABABABABABABAB
```

```
for (int contour=0; (contour < contours.size()); contour++)
{
    Scalar colour( rand()&0xFF,rand()&0xFF,rand()&0xFF );
    drawContours( contours_image, contours, contour, colour,
CV_FILLED, 8, hierarchy );
}
```

Regions Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 8

7

8

## k means Clustering

- We'd like to
  - identify significant colours in images
    - Concise descriptions
    - Object tracking
  - reduce the number of colours in any image
    - Compression
- How do we find the best colours?
- k means clustering
  - Creates k clusters of pixels
  - Unsupervised learning

Regions Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 9

## k means Clustering – Algorithm

- Number of clusters (k) is known in advance (or determine the k with the maximum confidence)
- Initialise the k cluster exemplars either randomly or use the first k patterns or ...
- 1<sup>st</sup> pass: Allocate patterns to the closest existing cluster exemplar and recompute the exemplar as the centre of gravity
- 2<sup>nd</sup> pass: Using the final exemplars from the first pass allocate all patterns cluster exemplars.

Regions Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 10

9

10

## k means Clustering

(a) Initial state with 2 red exemplars and 2 green arrows pointing to them.  
(b) After 1st pass, 2 new blue exemplars are shown.  
(c) After 2nd pass, 2 new green exemplars are shown.  
(d) After 3rd pass, 2 new blue exemplars are shown.  
(e) After 4th pass, 2 new green exemplars are shown.  
(f) After 5th pass, 2 new blue exemplars are shown.  
(g) After 6th pass, 2 new green exemplars are shown.  
(h) After 7th pass, 2 new blue exemplars are shown.  
(i) After 8th pass, 2 new green exemplars are shown.  
(j) After 9th pass, 2 new blue exemplars are shown.

Regions Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 11

## k means Clustering

Different values of k (10, 15 & 20 random exemplars):

- Not all clusters end up with patterns
- More exemplars generally gives a more faithful representation

Regions Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 12

11

12

## k means Clustering

- Choosing the best number of exemplars
  - Evaluate the resulting clusters

Davies-Bouldin index measures cluster separation:

$$DB = 1/k \sum_{i=1}^k \max_{j \neq i} \frac{(\sigma_i + \sigma_j)}{\delta_{ij}}$$

OR Check that distributions are normal

Regions Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 13

13

## k means Clustering

- Using random exemplars gives non-deterministic results (30 random exemplars each time):

Regions Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 14

14

## k means Clustering

```
// Store the image pixels as an array of samples
Mat samples{image.rows*image.cols, 3, CV_32F};
float* sample = samples.ptr<float>(0);
for(int row=0; row<image.rows; row++)
  for(int col=0; col<image.cols; col++)
    for (int channel=0; channel < 3; channel++)
      samples.at<float>(row*image.cols+col,channel) =
        (uchar) image.at<Vec3b>(row,col)[channel];
// Apply k-means clustering, determining the cluster
// centres and a label for each pixel.
....
```

Regions Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 15

15

## k means Clustering

```
Mat labels, centres;
kmeans(samples, k, labels, TermCriteria( CV_TERMCRIT_ITER |
  CV_TERMCRIT_EPS, 0.0001, 10000), iterations,
  KMEANS_PP_CENTERS, centres );
// Use centres and label to populate result image
Mat& result_image = Mat( image.size(), image.type() );
for(int row=0; row<image.rows; row++)
  for(int col=0; col<image.cols; col++)
    for (int channel=0; channel < 3; channel++)
      result_image.at<Vec3b>(row,col)[channel] =
        (uchar) centres.at<float>(*labels.ptr<int>(row*image.cols+col)), channel);
```

Regions Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 16

16

## Watershed segmentation

- In geology watersheds separate different catchment basins.
  - Where the rain gets caught
- In computer vision we
  - Identify all minima.
  - Label as different regions.
  - Flood from the minima extending the regions.
  - Where regions meet we have watershed lines.
- Minimum what?
  - Greyscale
  - Gradient
  - Inverse of the chamfer distance

Images from <http://uk.mathworks.com/company/newsletters/volume16/3/watershed.html> © 1994-2017 The MathWorks, Inc.

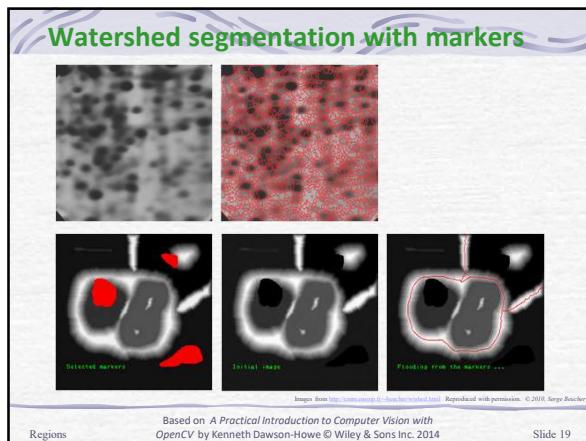
Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 17

17

## Watershed segmentation example

Images from <http://www.cse.lehigh.edu/~sander/watershed.html> Reproduced with permission. © 2010, Serge Beucher

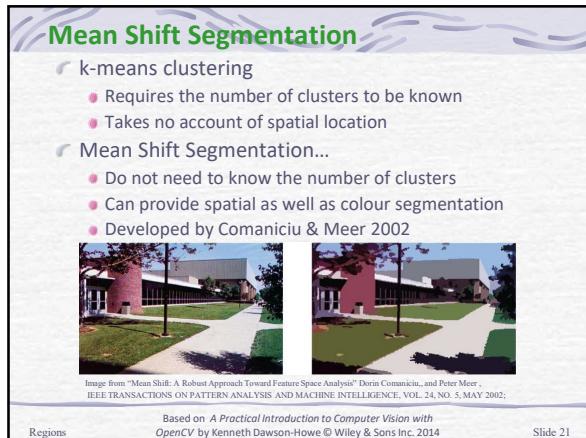
Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 18



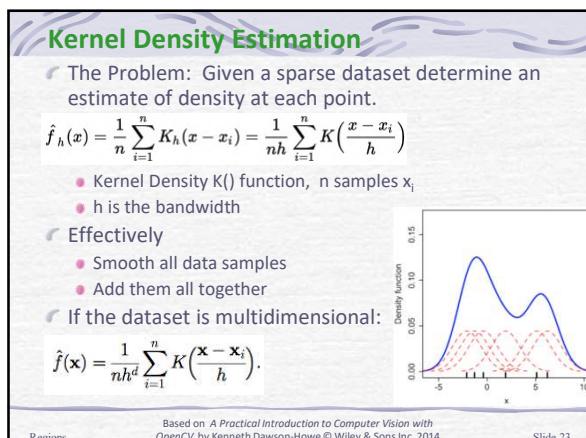
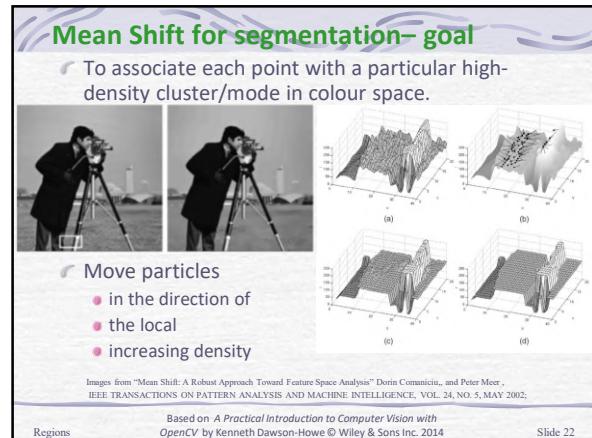
19



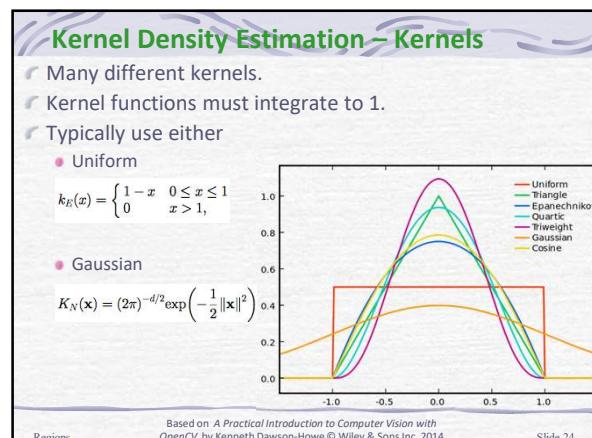
20



21



23



24

## Mean Shift – Basic Algorithm

- For each particle (pixel)
  - Estimate the local kernel density and more importantly the direction of local increasing density (the *mean shift vector*)
  - Shift the particle to the new mean.
  - Re-compute until the location stabilizes.
- Finally identify which pixels ended up in the same location
  - Mark these as members of the same cluster.
  - Determine the local mean of similar particles

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Regions Slide 25

25

## Mean Shift – The local kernel density

- We limit the points included in the kernel density estimate based on distance and on similar to the current point.

$$K_{h_s, h_r}(\mathbf{x}) = \frac{C}{h_s^d h_r^d} k\left(\frac{\|\mathbf{x}^s\|^2}{h_s}\right) k\left(\frac{\|\mathbf{x}^r\|^2}{h_r}\right)$$

- We must use both a spatial kernel and a colour kernel.
- Both can be Gaussian
- Spatial kernel limits/weights the region to consider around the current point
- Colour/range kernel limits/weights the colour/intensity of the points to be included in the mean.

Maths from "Mean Shift: A Robust Approach Toward Feature Space Analysis" Dorin Comaniciu, and Peter Meer IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 5, MAY 2002;

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Regions Slide 26

26

## Mean Shift – The mean shift vector

- For radially symmetric kernels  $K(\mathbf{x}) = c_{k,d} k(\|\mathbf{x}\|^2)$
- So the KDE becomes  $\hat{f}_{h,K}(\mathbf{x}) = \frac{c_{k,d}}{nh^d} \sum_{i=1}^n k\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)$  as  $\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$
- We are interested in the rate of change...

$$\hat{\nabla} f_{h,K}(\mathbf{x}) \equiv \nabla \hat{f}_{h,K}(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^n (\mathbf{x} - \mathbf{x}_i) k'\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)$$

- Set  $g(x) = -k'(x)$  so

$$= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^n (\mathbf{x}_i - \mathbf{x}) g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)$$

$$= \frac{2c_{k,d}}{nh^{d+2}} \left[ \sum_{i=1}^n g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right) \right] \left[ \frac{\sum_{i=1}^n \mathbf{x}_i g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^n g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)} - \mathbf{x} \right]$$

- Define the mean shift vector as

$$m_{h,G}(\mathbf{x}) = \frac{\sum_{i=1}^n \mathbf{x}_i g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^n g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)} - \mathbf{x}$$

Maths from "Mean Shift: A Robust Approach Toward Feature Space Analysis" Dorin Comaniciu, and Peter Meer IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 5, MAY 2002;

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Regions Slide 27

27

## Mean Shift – Spatial kernel functions

Varying the Spatial radius and varying the colour radius

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Regions Slide 28

28

## Mean Shift – Pros and cons

- + Do not need to know the number of clusters *a priori*.
- + Provides spatial as well as colour segmentation.
- Selection of kernel widths can be very hard.
- It is quite slow particularly if there are a lot of clusters.

Images from "Mean Shift: A Robust Approach Toward Feature Space Analysis" Dorin Comaniciu, and Peter Meer IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 5, MAY 2002; From <http://www.comaniciu.net/Papers/MRRobustApproach.pdf>

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Regions Slide 29

29

## Recognition

- Template Matching
  - Chamfer Matching
- Statistical Pattern Recognition (SPR)
- Support Vector Machines (SVM)
- Cascade of Haar classifiers (Haar)
- Principal Components Analysis (PCA)

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

1

## Template Matching - Topics

- Applications
- Matching criteria
- Control strategies
- Chamfer Matching

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

2

## Template – Applications for template matching

- Searching
  - Locating objects

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

3

## Template – Applications for template matching

- Recognition

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

4

## Template – Applications for template matching

- Visual inspection
  - Golden template matching
- Matching
  - Stereo Vision
  - Tracking

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

5

## Template – Matching Algorithm

- Basic Algorithm
  - Inputs – Image & Object
  - For every possible position of the object in the image
    - Evaluate a match criterion
  - Search for local maxima of the match criterion above some threshold
- Problems
  - ‘Every possible position’?
  - ‘Match criterion’
  - ‘Local maxima above some threshold’

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

6

## Template – Matching criteria

$D_{SquareDifferences}(i, j) = \sum_{(m,n)} (f(i + m, j + n) - t(m, n))^2$

$D_{NormalisedSquareDifferences}(i, j) = \frac{\sum_{(m,n)} (f(i + m, j + n) - t(m, n))^2}{\sqrt{\sum_{(m,n)} f(i + m, j + n)^2 \sum_{(m,n)} t(m, n)^2}}$

$D_{CrossCorrelation}(i, j) = \sum_{(m,n)} f(i + m, j + n) \cdot t(m, n)$

$D_{NormalisedCrossCorrelation}(i, j) = \frac{\sum_{(m,n)} f(i + m, j + n) \cdot t(m, n)}{\sqrt{\sum_{(m,n)} f(i + m, j + n)^2 \sum_{(m,n)} t(m, n)^2}}$

Recognition Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 7

7

## Template – Matching criteria example

- Simple example
- Matching criteria
- Boundaries...

0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	1	1	1	1	0
0	0	0	1	1	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0

Matching Space

Recognition Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 8

8

## Template Matching in OpenCV

```
Mat matching_space;
matching_space.create(
    search_image.cols-template_image.cols+1,
    search_image.rows-template_image.rows+1, CV_32FC1 );
matchTemplate( search_image, template_image,
                matching_space, CV_TM_CCORR_NORMED );
// Other measures: CV_TM_CCORR, CV_TM_SQDIFF,
//                  CV_TM_SQDIFF_NORMED
```

Recognition Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 9

9

## Finding local maxima/minima

- Local maxima – Dilate + Look for unchanged values + Threshold
- Local minima – Erode + Look for unchanged values + Threshold

```
Mat dilated, thresholded_input, local_maxima, thresholded_8bit;
dilate( input, dilated, Mat());
compare( input, dilated, local_maxima, CMP_EQ );
threshold( input, thresholded_input, threshold, 255, THRESH_BINARY );
thresholded_input.convertTo( thresholded_8bit, CV_8U );
bitwise_and( local_maxima, thresholded_8bit, local_maxima );
```

Recognition Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 10

10

## Template – Control Strategies for Matching

- Goal: Localise close copies
  - Size, orientation
  - Geometric distortion
- Use an image hierarchy
  - Low resolution first
  - Limit higher resolution search
- Search higher probability locations first
  - Known / learnt likelihood
  - From lower resolution

Recognition Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 11

11

## Template – Chamfer matching

- Template matching requires very close matches
- Objects often appear very slightly different
  - Orientation
  - Noise
  - Sampling
- We want a more flexible approach

Recognition Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 12

2

### Template – Chamfering

Compute chamfered image for a binary edge image.

```

For every point
    If (edge point)
        Set c(i, j) = 0
    Else Set c(i, j) = ∞
For j = min to max
    For i = min to max
        c(i, j) = minq∈AL [distance( (i, j), q ) + f( q )]
For j = max to min
    For i = max to min
        c(i, j) = minq∈BR [distance( (i, j), q ) + f( q )]

```

Canny( gray\_image, edge\_image, 100, 200, 3);  
threshold(edge\_image,edge\_image,127,255,THRESH\_BINARY\_INV);  
distanceTransform(edge\_image,chamfer\_image,CV\_DIST\_L2,3);

Chamfer Image

Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 13

13

### Template – Chamfer matching

Video Input      Static Background      Difference

Edges from difference      Chamfer image      Result

Modus      Degree of fit      Local minima

Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 14

14

### Template – Chamfering for matching

Compare with boundary template

- Sum of chamfer values for boundary points
- Low values best

3.8 2.8 2.4 2 1 0 1 2	1 1 1 1	27.4 19.6 12.8 8 27.4
3.4 2.4 1.4 1 1 0 1 1.4	1 1 1	23.2 16.8 10.4 5 9.4
3 2 1 0 0 0 0 1	1 1 1	21 14 8 0 6
3 2 1 0 0 1 0 1	1 1 1 1	23.2 16.8 10.4 5 9.4
3 2 1 0 1 1 0 1	1 1 1 1 1	3.4 2.4 1.4 1 1 1 1 1.4
3 2 1 0 1 1 0 1	1 1 1 1 1	
3 2 1 0 0 0 0 1	1 1 1 1 1	
3.4 2.4 1.4 1 1 1 1 1.4		

Matching Space

Template

Chamfer Image

Chamfer matching is a simple routine to write (provided in TIPS);

Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 15

15

### Chamfer matching example

Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 16

16

### SPR – Statistical Pattern Recognition

Features

Classification

	Rectangularity	Elongatedness
Albatross	0.28	3.63
Vulture	0.45	5.56
Hawk	0.39	2.39
Falcon	0.33	2.11

	Rectangularity	Elongatedness
Falcon	0.49	5.29
Vulture	0.30	3.67
Albatross	0.33	3.63
Hawk	0.41	2.38

Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 17

17

### SPR – Features: Minimum Bounding Rectangle

Turn rectangle through discrete steps

Only one quadrant

Metrics:

- Length to Width ratio
- Length / Width
- Rectangularity
- Area / (Length \* Width)
- Convex Hull to Minimum Bounding Rectangle area ratio
- Area inside convex hull / (Length \* Width)

RotatedRect min\_bounding\_rectangle =  
minAreaRect(contours[contour\_number]);

Based on A Practical Introduction to Computer Vision with OpenCV by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 18

## SPR – Features: Elongatedness

- CANNOT BE the ratio of the length and width of the minimum bounding rectangle
- Ratio of region area divided by the square of it's thickness

Erosion       $\text{elongatedness} = \frac{\text{area}}{(2d)^2}$

	Rectangularity	Elongatedness
Circle	0.79	0.78
Square	1.00	1.00
Rectangle	1.00	1.85

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 19

19

## SPR – Features: Moments & Moment invariants

- Moments measure the distribution of shape  $M_{xy} = \sum_i \sum_j i^x j^y f(i, j)$
- Central moments:

$$\mu_{00} = M_{00} \quad \mu_{01} = 0 \quad \mu_{10} = 0$$

$$\mu_{11} = M_{11} - \frac{M_{10}M_{01}}{M_{00}} \quad \mu_{20} = M_{20} - \frac{M_{10}M_{10}}{M_{00}} \quad \mu_{02} = M_{02} - \frac{M_{01}M_{01}}{M_{00}}$$

$$\mu_{21} = M_{21} - \frac{2M_{10}M_{11}}{M_{00}} - \frac{M_{01}M_{20}}{M_{00}} + \frac{2M_{10}M_{10}M_{01}}{M_{00}M_{00}} \quad \mu_{03} = M_{03} - \frac{3M_{01}M_{02}}{M_{00}} + \frac{2M_{01}M_{01}M_{01}}{M_{00}M_{00}}$$

$$\mu_{12} = M_{21} - \frac{2M_{01}M_{11}}{M_{00}} - \frac{M_{10}M_{02}}{M_{00}} + \frac{2M_{10}M_{01}M_{01}}{M_{00}M_{00}} \quad \mu_{30} = M_{30} - \frac{3M_{10}M_{20}}{M_{00}} + \frac{2M_{10}M_{10}M_{10}}{M_{00}M_{00}}$$

- Scale invariant moments:

$$\eta_{xy} = \frac{\mu_{xy}}{\mu_{00}^{(1+x+y)/2}}$$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 20

20

## SPR – Features: Moment invariants

- Moment invariants / Hu moments:

$$I_1 = \eta_{20} + \eta_{02}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4(\eta_{11})^2$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)$$

$$I_6 = (\eta_{20} - \eta_{02})(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)$$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 21

21

## SPR – Features: Moments and Moment invariants

Shape	Perimeter	Area	Bounding Box	Convex Hull	Hu Moments	Hole Areas			
0	80	651	555	0.17	0.00	0.00	175		
1	101	274	522	429	0.47	0.13	0.01		
2	111	347	570	528	0.42	0.07	0.00		
3	121	351	522	530	0.40	0.06	0.00		
4	85	347	592	442	0.20	0.01	0.00	60	
5	122	351	570	504	0.43	0.06	0.00		
6	102	392	620	534	0.23	0.01	0.00	76	
7	94	267	551	394	0.46	0.09	0.04		
8	65	438	651	571	0.20	0.01	0.00	45	62
9	101	389	589	524	0.23	0.01	0.00	74	

Moments contour\_moments;  
double hu\_moments[7];  
contour\_moments = moments( contours[contour] );  
HuMoments( contour\_moments, hu\_moments );

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 22

22

## SPR – Features: Convex hull

- Smallest convex region

Algorithm

- Start with
  - Any point on convex hull
  - Previous vector direction
- Search all other boundary points
  - Find point with least angle to previous vector
- Switch to new point and vector
- Go to 2 unless new point = start point.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 23

23

## SPR – Features: Convex hull

```
vector<vector<Point>> hulls(contours.size());
for (int contour=0; (contour<contours.size()); contour++)
{
    convexHull(contours[contour], hulls[contour]);
}
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 24

4

Shape	Height	Width	Hull / Width	Hull / Box	# holes	Hole Areas	# concavities	Concavity Details
0	2.46	0.85	1	4.1	0			
1	4.71	0.85	1	4.1	0	2.04@243	0.03@261	
2	2.83	0.85	0	5.36@213	2.5@45			
3	2.83	0.87	0	3.1@195	2.7@154			
4	2.83	0.84	0	0.6	0	0.00@10	0.00@10	
5	2.54	0.86	0	4.1@1.304	2.4@302			
6	2.13	0.79	1	1.5	3	3.1@317	0.03@94	
7	2.36	0.57	0	9	4.3@205	0.1@294		
8	2.83	0.80	2	1.8	1.6	0.3@205	0.2@355	
9	2.62	0.81	1	1.9	4	3.0@143	0.1@240	

Shape	Height	Width	Hull / Width	Hull / Box	# holes	Hole Areas	# concavities	Concavity Details
0	2.62	0.81	1	1.8	0	3.2@137	0.03@66	
5	2.54	0.85	0	3.0@140	2.4@302			
0	2.54	0.87	1	4.5	3	0.4@171	0.1@116	
3	2.67	0.81	0	3.1@184	0.4@343			
7	2.49	0.81	0	1.3@104	0.1@104			
5	2.63	0.81	2	1.6	1.5	0.3@354	0.3@203	
2	2.43	0.84	0	3.1@209	2.1@204			
9	2.62	0.83	0	3	0.0@143	2.4@300		

25

## SPR – Features: Concavities and Holes

26

- Perimeter length
- Approximately the number of boundary elements  
`contours[contour].size()`
- Should really take into account direction
- Circularity =  $(4 * \pi * \text{Area}) / (\text{Perimeter length})^2$

Shape	Perimeter	Area	Bounding Box	Convex Hull	Hu Moments	Hole Areas	
0	80	406	651	551	0.17	0.00	0.00
1	101	274	532	429	0.47	0.13	0.01
2	111	347	570	528	0.42	0.07	0.00
3	121	351	522	530	0.40	0.06	0.00
4	85	347	592	442	0.20	0.01	0.00
5	122	351	570	504	0.43	0.06	0.00
6	102	392	620	534	0.37	0.00	0.00
7	24	262	557	446	0.46	0.09	0.04
8	65	438	592	571	0.20	0.01	45
9	101	389	589	524	0.23	0.01	0.00

27

- SPR – Probability Review – Basics
  - $P(A) = \lim_{n \rightarrow \infty} N(A) / n$
  - Probability of two events A and B:
    - Independent:  
 $P(AB) = P(A)P(B)$
    - Dependent:  
 $P(AB) = P(A|B)P(B)$
    - Conditional Probability  $P(A|B)$
- Typical problem:
  - Given some evidence  $x$  from an unknown object, what class  $W_i$  is the object?
  - Training
    - A-priori* probability –  $p(x | W_i)$
    - Relative probability –  $p(W_i)$
    - A-posteriori* probability –  $p(W_i | x)$

28

## SPR – Probability Review – Bayes Theorem

- For two classes  $A$  and  $B$  the  $a$ -posteriori probability is:
$$P(B|A) = P(A|B)P(B) / P(A)$$
- Where  $W_i$  forms a partitioning of the event space:
$$p(W_i | x) = \frac{p(x | W_i)p(W_i)}{\sum_j p(x | W_j)p(W_j)}$$

29

# SPR – Probability Density Functions

30

## SPR – Classification

- Object recognition
  - Classes ( $w_1, w_2, \dots, w_R$ )
- Classifier
  - Input Pattern / features ( $x_1, x_2, \dots, x_n$ )
- Feature space
  - Choosing the features (Example)
  - Clusters in feature space
- Separability
  - Hyper-surfaces
  - Linear separability
  - Inseparable classes
- Probabilistic Classifier

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition

Slide 31

31

## SPR – Probabilistic classifier

- Optimal classifier
- Based on Probability Density Functions
  - Remove normalisation from Bayes rule...  $\sum_j p(x|W_j)p(W_j)$
- Resultant Discrimination function:  $g_r(x) = p(x|W_r)p(W_r)$
- $g_r(x) \geq g_s(x)$
- Unknown class:  $g_r(x) > \text{threshold}$
- Mean Loss Function:  $J(q) = \int_x \sum_{s=1..R} \lambda[d(x, q)|W_s] p(x|W_s) P(W_s) dx$   
where  $\lambda[W_r|W_s] = \begin{cases} 0 & \text{if } r = s \\ 1 & \text{otherwise} \end{cases}$
- Advantages / Disadvantages
  - Accuracy
  - Extensive training

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition

Slide 32

32

## SPR – Classifier learning

- Training set
  - Must be representative
  - Must be inductive
  - Can use Kernel Density Estimation
- Training set size
  - Training set provides the unknown statistical information
  - Size will typically have to be increased several times
- Learning strategies
  - Supervised:
    - Probability density estimation – estimating  $p(x|w_r)$  &  $P(w_r)$
    - Training set includes class specification for every instance
  - Unsupervised:
    - Cluster Analysis
    - Look for similarities in feature space

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition

Slide 33

33

## SPR: Real example

	Shape	Height/Width	Ht/Width	# holes	Hole Areas	Concavity Details
0	2.46	0.85	1	4.1	0	
1	4.71	0.58	0		2	0.4@243 0.03@261
2	2.83	0.85	0		5	3.6@213 2.5@45
3	2.83	0.87	0		3	5.7@185 0.7@354
4	2.38	0.62	0	0.6	3	0.9@180 0.1@27
5	2.38	0.62	0		4	1.1@140 2.4@302
6	2.13	0.79	1	1.5	3	3.1@317 0.03@94
7	2.36	0.57	0		9	4.3@205 0.1@294
8	2.83	0.80	1	1.81	2	0.3@205 0.2@355
9	2.62	0.81	1	1.9	4	3.0@143 0.1@240

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition

Slide 34

34

## Support Vector Machines

- Popular technique for two-class problem
- Consider two linearly separable classes in  $n$ -dimensional feature space
- Many hyper-surfaces usually exist
- Optimal classification
  - Maximise the margin between the classes.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe 2015

Recognition

Slide 35

35

## Support Vector Machines – Example

From *Vision-Based Traffic Data Collection Sensor for Automotive Applications* by Llorente DF, Sánchez S, Ocaña M, Sotelo MA - (2010)  
[http://openi.nlm.nih.gov/detailedresult.php?img=3270873\\_sensors-10-00860f12&req=4](http://openi.nlm.nih.gov/detailedresult.php?img=3270873_sensors-10-00860f12&req=4)

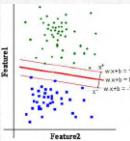
Recognition

Slide 36

36

## Support Vector Machines – Formulation

- Given many  $n$ -dimensional training samples  $x_i$  with associated class identifiers  $\omega_i = \{-1, 1\}$
- Define the separating hyper-planes  $w \cdot x + b = 0$   
 $w \cdot x + b = 1$        $w \cdot x + b = -1$
- Constraint:  $\omega_i(w \cdot x_i + b) \geq 1$
- If  $x^+$  is a point on  $w \cdot x + b = 1$  and  $x^-$  is the nearest point on  $w \cdot x + b = -1$   
then  $\lambda w = x^+ - x^-$
- Multiply by  $w$   $\lambda w \cdot w = x^+ \cdot w - x^- \cdot w = (1 - b) - (-1 - b) = 2$   
Therefore  $\lambda w \cdot w = \lambda \|w\|^2 = 2$  as  $\|w\| = \sqrt{w \cdot w}$  and...
- $\|x^+ - x^-\| = \|\lambda w\| = \left\| \frac{2 \cdot w}{\|w\|^2} \right\| = \left\| \frac{2 \cdot w}{w \cdot w} \right\| = \left\| \frac{2}{\|w\|} \right\| = \frac{2}{\|w\|}$
- Hence we must minimise  $\|w\|$  to maximise the separation, subject to  $\omega_i(w \cdot x_i + b) \geq 1$



Recognition

© Kenneth Dawson-Howe 2015

Slide 37

37

## Support Vector Machines – Lagrangian Optimisation

- This is a minimisation problem subject to constraints which is amenable to solution by Lagrangian optimisation

$$L(w, b, \alpha) = \frac{\|w\|^2}{2} - \sum_{i=1}^n \alpha_i [\omega_i (w \cdot x_i + b) - 1]$$

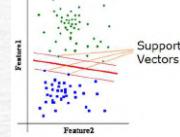
- We determine the Lagrangian multipliers ( $\alpha_i$ ) will be zero except for the support vectors.

- Classification is simply based on  $x_i$

entering an unknown feature vector

into the original equation  $f(x_i) = w \cdot x_i + b$

and classifying based on whether the value is positive or negative (preferably  $\geq 1$  or  $\leq -1$ ).



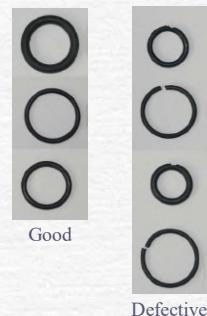
Recognition

© Kenneth Dawson-Howe 2015

Slide 38

38

## Support Vector Machines – Example



Recognition

Slide 39

39

## SVM Example – Training

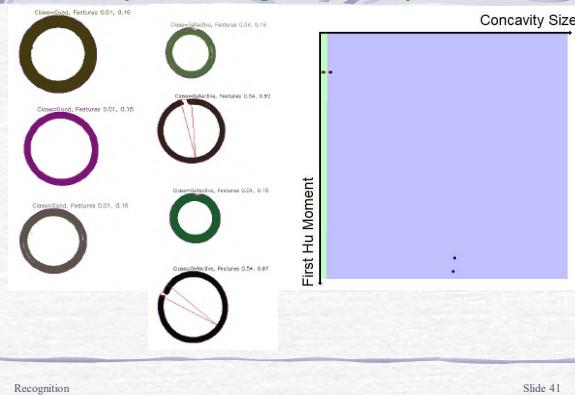
```
Ptr<SVM> svm = ml::SVM::create();
svm->setType(SVM::C_SVC);
svm->setKernel(SVM::LINEAR);
Mat labelsMat(number_of_samples, 1, CV_32SC1, labels);
Mat trainingDataMat(number_of_samples, 2, CV_32FC1,
                     training_data);
Ptr<ml::TrainData> tData = ml::TrainData::create(trainingDataMat,
                                                 ml::SampleTypes::ROW_SAMPLE, labelsMat);
svm->train(tData);
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 40

40

## SVM Example – Training & Feature Space



Recognition

Slide 41

41

## SVM – Support vectors

```
Mat support_vectors = svm->getSupportVectors();
for (int support_vector_index = 0; support_vector_index <
support_vectors.rows; ++support_vector_index)
{
    const float* v = support_vectors.ptr<float>(support_vector_index);
    circle(feature_space, Point((int)v[0],(int)v[1]), 3, Scalar(0,0,255));
}
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 42

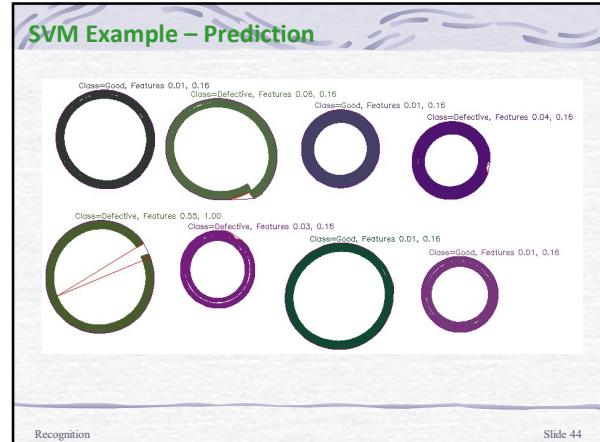
42

### SVM Example – Prediction

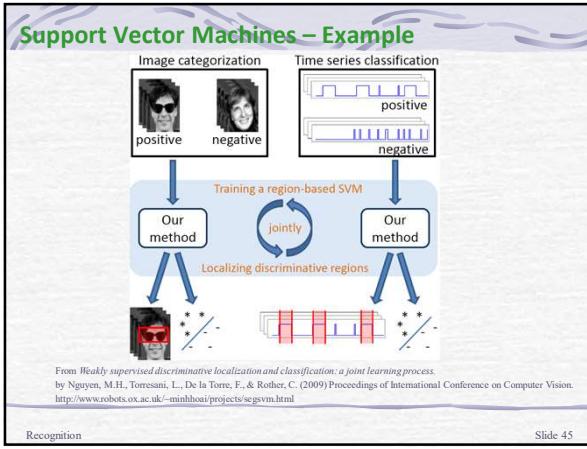
```
// Try to predict the class
Mat sampleMat = (Mat<float>(1,2) << feature[0], feature[1]);
float prediction = svm->predict(sampleMat);
class_id = (prediction > 0.0) ? 1 : (prediction < 0.0) ? 2 : 0;
```

Recognition Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014 Slide 43

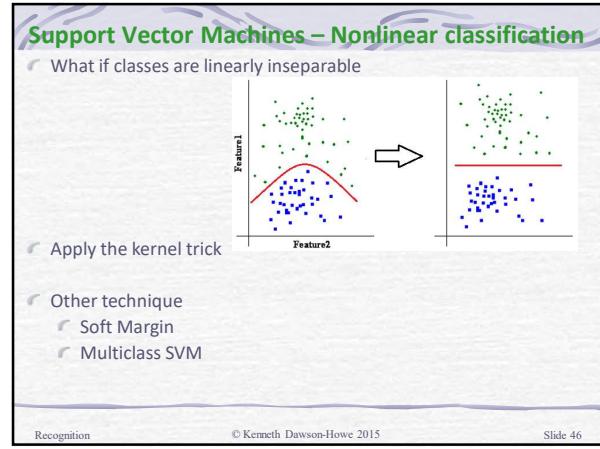
43



44



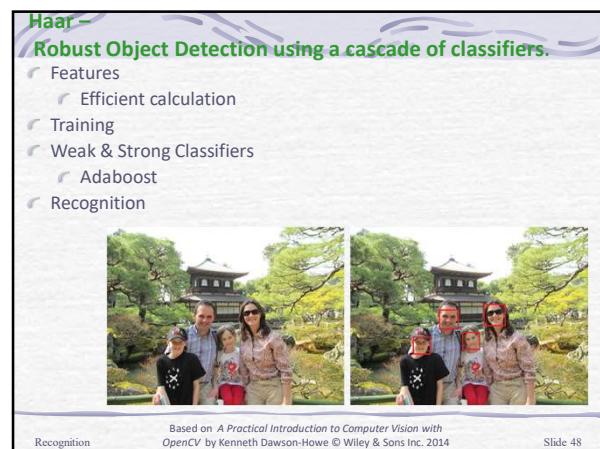
45



46



47



## Haar – In OpenCV.

```
CascadeClassifier cascade;
if( !cascade.load(
    "haarcascades/haarcascade_frontalface_alt.xml" )
{
    vector<Rect> faces;
    equalizeHist( gray_image, gray_image );
    cascade.detectMultiScale( gray_image, faces, 1.1, 2,
        CV_HAAR_SCALE_IMAGE, Size(30, 30) );
}
```

Recognition

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 49

49

## Haar – Overview

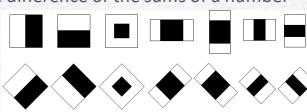
- ❖ Training using a number of positive and negative samples
- ❖ Uses simple (Haar like) features
  - ❖ Efficient calculation...
- ❖ Selects a large number of these features during training to create strong classifiers
- ❖ Links a number of strong classifiers into a cascade for recognition
  - ❖ Efficiency...
- ❖ Can work at different scales

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 50

50

## Haar – Features

- ❖ Features determined as the difference of the sums of a number of rectangular regions
 
- ❖ Place the mask in a specific location and at a specific scale
- ❖ Subtract the sum of the 'white pixels' from the sum of the 'black pixels'
 
- ❖ Why does this work?

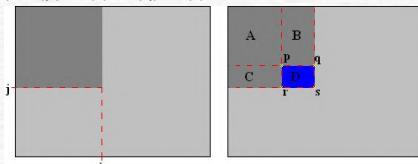
Recognition

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 51

51

## Haar – Efficient Calculation – Integral Image

- ❖ Integral Image:
  - ❖ Every point  $ii(i,j) = \sum_{i'=0..i} \sum_{j'=0..j} \text{image}(i',j')$
  - ❖ Sum of points in rectangle D:
    - ❖  $\text{sum}(D) = ii(p) + ii(s) - ii(q) - ii(r)$
  - ❖ Features can be computed at any scale for the same cost

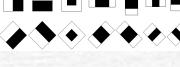
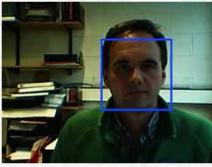
An **integral** function is provided which computes the integral image both normally and at 45°

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 52

52

## Haar – Training

- ❖ Number of possible features.
  - ❖ the variety of feature types
  - ❖ allowed variations in size and position
 
- ❖ Training must
  - ❖ Identify the best features to use at each stage
  - ❖ To do this positive and negative samples are needed...
 

Recognition

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 53

53

## Haar – Weak & Strong Classifiers

- ❖ Weak Classifier
  - ❖  $p_j \text{feature}_j(x) < p_j \vartheta_j$
  - ❖ Tune threshold ( $\vartheta_j$ )
- ❖ Strong Classifiers
  - ❖ Combine a number of weak classifiers...
  - ❖ E.g. 100% true positives with only 40% false positives using only two face features...
 

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 54

## Haar – Strong Classifiers – AdaBoost

Given  $n$  example images  $x_1..x_n$  together with classifications  $y_1..y_n$   
where  $y_i = 0, 1$  for negative and positive examples respectively.

Initialise weights  $w_{t,0} = 1 / (2 * (m * (1 - y_j) + l * y_j))$   
where  $m$  and  $l$  are the number of negative and positive examples respectively.

For  $t=1,..,T$

1. Normalize the weights (i.e. for all  $i$ ):  $w_{t,i} = w_{t,i} / (\sum_{j=1..n} w_{t,j})$
2. For each feature,  $j$ , train a weak classifier  $h_j(x)$  and evaluate the error taking into account the weights:  $\epsilon_j = \sum_i w_{t,i} |h_j(x_i) - y_i|$
3. Select the classifier,  $h_j(x)$ , with the lowest  $\epsilon_j$ , save as  $c_t(x)$  with error  $E_t$
4. Update the weights (i.e. for all  $i$ ):  $w_{t+1,i} = w_{t,i} \beta_t^{(1-\epsilon_t)}$   
where  $\epsilon_t = |c_t(x_i) - y_i|$  and  $\beta_t = E_t / (1-E_t)$

The final strong classifier is:  $h(x) = 1 \text{ if } \sum_{t=1..T} \alpha_t c_t(x) \geq \sum_{t=1..T} \alpha_t$ ,  
0 otherwise  
where  $\alpha_t = \log 1/\beta_t$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 55

55

## Haar – Classifier cascade

- Object recognition is possible with a single strong classifier
- To improve detection rates AND to reduce computation time:
  - A cascade of strong classifiers can be used
  - Each stage either accepts or rejects
  - Only those accepted pass to the next stage
    - Efficient computation...
- Strong classifiers trained using AdaBoost
  - On the remaining set of data

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 56

56

## Haar – Recognition

Face recognition

- 38 stages
- 6000+ features
- 4916 positive samples
- 9544 negative samples
- Scale independence

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Recognition      Slide 57

57

## Principal Component Analysis (PCA)

- Statistical Technique for
  - Data analysis
  - Data compression
  - Recognition
- Analyses data covariance
- Identifies principal directions

**For example:**

- Given 2 dimensional data
- and  $N$  samples/vectors
- Find
  - the mean sample
  - the direction of maximum covariance
  - the direction orthogonal to this

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe, 2015

Recognition      Slide 58

58

## PCA – Background – Eigenvalues and Eigenvectors

- Consider a square matrix  $A$
- The eigenvalues of  $A$  are the roots of the *characteristic equation*:  $\det(A - \lambda I) = |A - \lambda I| = 0$
- For each eigenvalue  $\lambda$  there will be an eigenvector  $x$  such that  $Ax = \lambda x$ 
  - Many possible values of  $x$
- An  $n \times n$  matrix will have  $n$  eigenvalues
- Consider  $n=2$ . There will be 2 eigenvalues:  $\lambda_1, \lambda_2$ 
  - with corresponding eigenvectors  $x_1, x_2$
  - where  $Ax_1 = \lambda_1 x_1$  and  $Ax_2 = \lambda_2 x_2$

Recognition      © Kenneth Dawson-Howe, 2015      Slide 59

59

## PCA – Background – Eigenvalues and Eigenvectors

- Combining we get  $A[x_1 \ x_2] = [x_1 \ x_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$
- and  $A\Phi = \Phi\Lambda$
- where  $\Phi = [x_1 \ x_2]$  and  $\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$
- If we normalise the eigenvectors then  $\Phi\Phi^T = \Phi^T\Phi = I$ 
  - so  $\Phi^T A \Phi = \Phi^T \Phi \Lambda = \Lambda$
  - and  $A = A\Phi\Phi^T = \Phi\Lambda\Phi^T$

Recognition      © Kenneth Dawson-Howe, 2015      Slide 60

60

10

## PCA – Theory – Organise Data

- In PCA we have  $N$  samples/vectors in some  $n$ -dimensional space
- Combine into a data matrix  $D$
- Each row is a sample
- Determine the mean of the samples
- Compute mean-centred data  $U$

$$D = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_n^1 \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_1^N & x_2^N & \dots & x_n^N \end{bmatrix}$$

$$\mu = \left[ \frac{\sum_{i=1}^N x_1^i}{N} \quad \frac{\sum_{i=1}^N x_2^i}{N} \quad \dots \quad \frac{\sum_{i=1}^N x_n^i}{N} \right]$$

$$U = D - \begin{bmatrix} \mu \\ \mu \\ \vdots \\ \mu \end{bmatrix} = \begin{bmatrix} x_1^1 - \frac{\sum_{i=1}^N x_1^i}{N} & x_2^1 - \frac{\sum_{i=1}^N x_2^i}{N} & \dots & x_n^1 - \frac{\sum_{i=1}^N x_n^i}{N} \\ x_1^2 - \frac{\sum_{i=1}^N x_1^i}{N} & x_2^2 - \frac{\sum_{i=1}^N x_2^i}{N} & \dots & x_n^2 - \frac{\sum_{i=1}^N x_n^i}{N} \\ \vdots & \vdots & & \vdots \\ x_1^N - \frac{\sum_{i=1}^N x_1^i}{N} & x_2^N - \frac{\sum_{i=1}^N x_2^i}{N} & \dots & x_n^N - \frac{\sum_{i=1}^N x_n^i}{N} \end{bmatrix}$$

Recognition

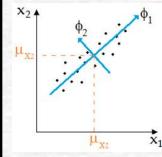
© Kenneth Dawson-Howe, 2015

Slide 61

61

## PCA – Theory – Covariance matrix

- Using the mean-centred data  $U$ , compute the covariance matrix  $\Sigma$
- We can then determine  $\Phi^T \Sigma \Phi = \Lambda$
- where the eigenvectors are in an orthogonal matrix  $\Phi$
- and the eigenvalues are in an ordered diagonal matrix  $\Lambda$



- We can consider  $\Phi$  as a linear transformation
- e.g. in our example it transforms  $[x_1, x_2]$  to  $[\phi_1, \phi_2]$
- The amplitudes of the eigenvalues in  $\Lambda$  are proportional to the percentage of overall variance accounted for by the associated eigenvector.

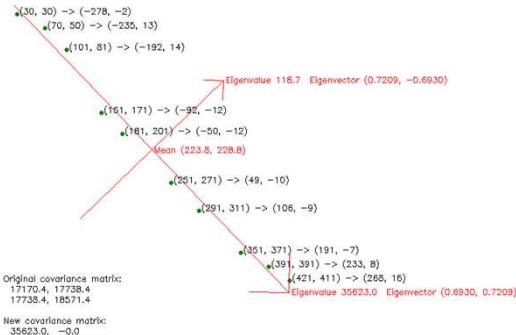
Recognition

© Kenneth Dawson-Howe, 2015

Slide 62

62

## PCA – Example



Recognition

© Kenneth Dawson-Howe, 2015

Slide 63

63

## PCA – Basics

```
float samples[]][2] = { {30,30}, {70,50}, ..., {421,411}, {391,391} };
int number_of_samples = sizeof(samples)/sizeof(int[2]);
Mat samples_matrix(number_of_samples, 2, CV_32FC1, samples);
...
PCA pca(samples_matrix, Mat(), 0, 2 );
Mat eigenvalues = pca.eigenvalues;
Mat eigenvectors = pca.eigenvectors;
Mat mean = pca.mean;
```

Recognition

© Kenneth Dawson-Howe 2015

Slide 64

64

## PCA – Basics

```
Mat transformed_samples_matrix = samples_matrix.clone();
for (int sample_no=0; sample_no<number_of_samples; sample_no++)
{
    Mat sample = samples_matrix.row(sample_no);
    Mat transformed_sample;
    pca.project(sample, transformed_sample);
    transformed_sample.row(0).copyTo(
        transformed_samples_matrix.row(sample_no));
}

Mat covariance,average,new_covariance,new_average;
calcCovarMatrix(samples_matrix,covariance,average,
    CV_COVAR_NORMAL | CV_COVAR_ROWS);
calcCovarMatrix(transformed_samples_matrix,new_covariance,
    new_average,CV_COVAR_NORMAL | CV_COVAR_ROWS);
```

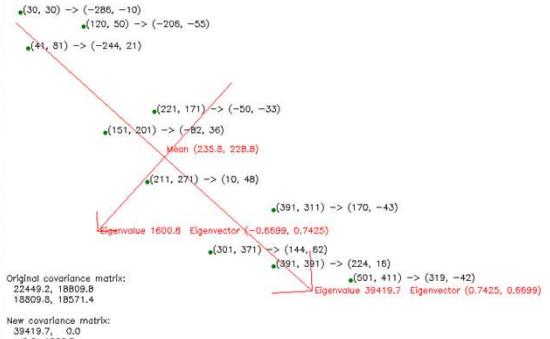
Recognition

© Kenneth Dawson-Howe 2015

Slide 65

65

## PCA – Example



Recognition

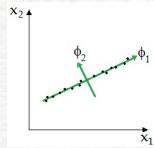
© Kenneth Dawson-Howe, 2015

Slide 66

66

## PCA – Compression

- If the variance on an axis is very small
- Variance caused by noise?
- Drop this dimension?



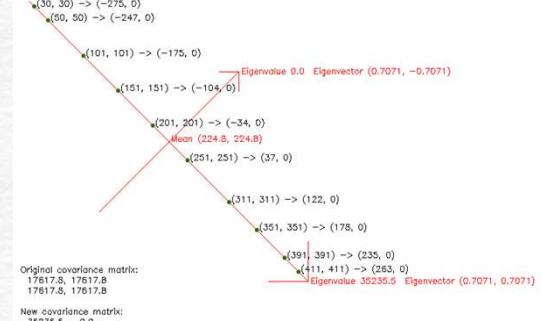
Recognition

© Kenneth Dawson-Howe, 2015

Slide 67

67

## PCA – Example



Recognition

© Kenneth Dawson-Howe, 2015

Slide 68

68

## PCA – Face Recognition

- Common example application of PCA
- Each face image is treated as a vector with  $n=\text{rows} \times \text{columns}$  dimensions!



$$\begin{aligned}
 &= \begin{matrix} \text{Binary data representation of the face image} \\ \dots \end{matrix} \\
 &= [9c, a1, 97, af, \dots, 74, 6f]_{900}
 \end{aligned}$$

- In the following example...  $n = 10304$

Recognition

© Kenneth Dawson-Howe, 2015

Slide 69

69

## PCA – Face Recognition

- Create a matrix  $D$ 
  - where each face image is a row
  - $D$  is a  $N \times n$  matrix
- Create mean centred data  $U$ 
  - Determine the mean of the rows (face images)
  - Subtract from all of the rows
- Compute the covariance matrix  $\Sigma = U^T U / (N - 1)$
- Solve for Eigenvectors and Eigenvalues as before.
- Normalise the Eigenvectors
- Select  $m$  Eigenvectors (with the largest  $m$  Eigenvalues)
  - Usually 20-50 for face recognition.
- For an unknown image find its location in PCA space
  - Look for the closest match  $p_\phi = (p_x - \mu_x) \cdot \Phi_{pca}^T$

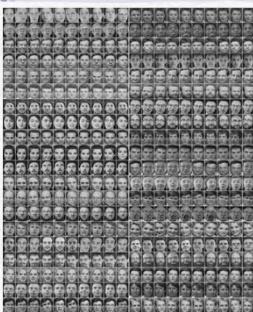
Recognition

© Kenneth Dawson-Howe, 2015

Slide 70

70

## PCA – Face Recognition – Database



Original images provided by AT&T Laboratories Cambridge  
<http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>

Recognition

© Kenneth Dawson-Howe, 2015

Slide 71

71

## PCA – Face Recognition – Some known faces



Original images provided by AT&T Laboratories Cambridge  
<http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>

Recognition

© Kenneth Dawson-Howe, 2015

Slide 72

12

## PCA – Face Recognition – Recognised faces

Original images provided by AT&T Laboratories Cambridge (Composite recognised images) © Kenneth Dawson-Howe  
http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Recognition      © Kenneth Dawson-Howe, 2015      Slide 73

73

## PCA – Face Recognition – Training & Prediction

```
vector<Mat> known_images;      vector<int> known_labels;
Mat current_image, unknown_image;
int current_label;
...
known_images.push_back(current_image);
known_labels.push_back(current_label);
...
Ptr<FaceRecognizer> recogniser = createEigenFaceRecognizer();
recogniser->train(known_images, known_labels);
...
int predicted_label = 0;      double confidence = 0.0;
recogniser->predict(unknown_image, predicted_label, confidence);
```

Recognition      © Kenneth Dawson-Howe 2015      Slide 74

74

## PCA – Image Reconstruction using Eigenvectors

- We can reconstruct images from PCA space  
 $p_x = p_{\phi} \cdot \Phi_{pca}^T + \mu_x$
- Using some number ( $m$ ) of principal components
- So that some percentage of variance is accounted for  
 $\% \text{ of Total Variance} = 100 \cdot \frac{\sum_{i=1}^m \lambda_i}{\sum_{j=1}^n \lambda_j}$

**Figure 3.23:** 32 original images of a boy's face, each 321 × 261 pixels. © Cengage Learning 2015.

**Figure 3.24:** Reconstruction of the image from four basis vectors  $b_i$ ,  $i = 1, \dots, 4$  which can be displayed as images. The linear combination was computed as  $q_1 b_1 + q_2 b_2 + q_3 b_3 + q_4 b_4 = 0.078 b_1 + 0.062 b_2 - 0.182 b_3 + 0.179 b_4$ . © Cengage Learning 2015. © Cengage Learning Engineering. Reproduced with permission. From *Image Processing, Analysis and Machine Vision* by Milan Sonka, Vaclav Hlavac and Roger Boyle

Recognition      Slide 75

75

## Edges

- Edge Detection
  - 1<sup>st</sup> derivative
  - 2<sup>nd</sup> derivative
- Contour Segmentation
- Hough Transform
- Least Squared Error
- Random Sample Consensus (RANSAC)

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 1

1

## Edges

- An approach to segmentation.
- The analysis of the discontinuities in an image.



- No correct answer?
- An alternative to region based processing.

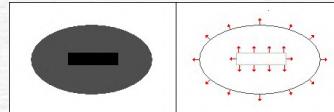
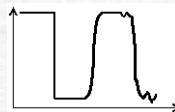
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 2

2

## Edge Detection – What is an edge?

- Where brightness changes abruptly
- Edges have
  - Magnitude (Gradient)
  - Direction (Orientation)
- Edge Profiles
  - Step
  - Real
  - Noisy

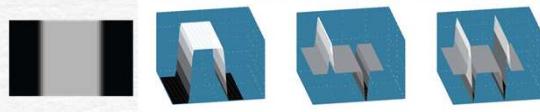



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 3

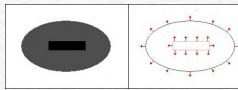
3

## Edge Detection – 1<sup>st</sup> derivative definitions



- Calculus...
  - Rate of change in two directions
  - Vector variable:
    - Gradient Magnitude
    - Orientation (0° is East)

$$\nabla f(i,j) = \sqrt{\left(\frac{\delta f(i,j)}{\delta i}\right)^2 + \left(\frac{\delta f(i,j)}{\delta j}\right)^2}$$

$$\phi(i,j) = \arctan\left(\frac{\delta f(i,j)}{\delta j}, \frac{\delta f(i,j)}{\delta i}\right)$$


Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 4

4

## Edge detection – Digital images

- Derivatives work on continuous functions
- Discrete domain
  - Differences
  - Orthogonal

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 5

5

## Edge detection – 1<sup>st</sup> derivative – Roberts

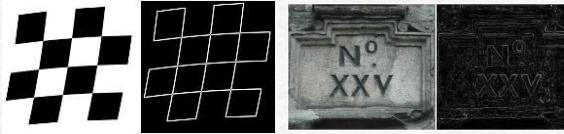
$$\delta_1(i,j) = f(i,j) - f(i+1,j+1)$$

$$\delta_2(i,j) = f(i,j+1) - f(i+1,j)$$

- Convolution Masks

$$h_1(i,j) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad h_2(i,j) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

- Sensitivity to Noise
- Binary Images



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 6

6

## Edge detection – Digital images

Need to define the partial derivatives so that they:

1. Cross at a single middle point
2. Preferably cross at the centre of a pixel
3. Evaluate points which are not too close together
4. Deal with some degree of image noise

Edges

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 7

7

## Edge detection – 1<sup>st</sup> derivative – Compass

Compass edge detectors:

- Partial derivatives defined for a number of orientations (typically 8)
- Prewitt:

$$h_1(i,j) = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, h_2(i,j) = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}, h_3(i,j) = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, h_4(i,j) = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$h_5(i,j) = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, h_6(i,j) = \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}, h_7(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, h_8(i,j) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & -1 & -1 \end{bmatrix}$$

- Use  $h_3(i,j)$  and  $h_1(i,j)$  to derive gradient and orientation



Edges

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

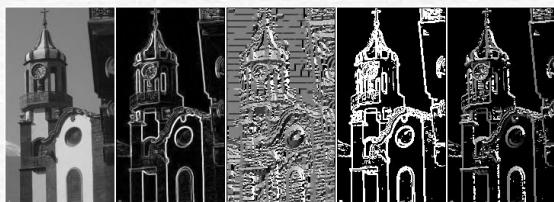
Slide 8

8

## Edge detection – 1<sup>st</sup> derivative – Compass

Sobel

$$h_1(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, h_3(i,j) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$



Edges

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 9

9

## Edge detection – 1<sup>st</sup> derivative – Compass

$$\nabla f(i,j) = \sqrt{\left(\frac{\delta f(i,j)}{\delta i}\right)^2 + \left(\frac{\delta f(i,j)}{\delta j}\right)^2} \quad \phi(i,j) = \arctan\left(\frac{\delta f(i,j)}{\delta j}, \frac{\delta f(i,j)}{\delta i}\right)$$

$$\nabla f(i,j) = \left| \frac{\delta f(i,j)}{\delta i} \right| + \left| \frac{\delta f(i,j)}{\delta j} \right|$$

```
Mat horizontal_derivative, vertical_derivative;
Sobel( gray_image, horizontal_derivative, CV_32F,1,0 );
Sobel( gray_image, vertical_derivative, CV_32F,0,1 );
Mat abs_gradient, l2norm_gradient, orientation;
abs_gradient = abs(horizontal_derivative) +
                abs(vertical_derivative);
cartToPolar(horizontal_derivative,vertical_derivative,
            l2norm_gradient,orientation);
```

Edges

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 10

10

## Edge detection – 1<sup>st</sup> derivative – Thresholding

Simple thresholding

- Too many points
- Too few points



Edges

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 11

11

## Edge detection – 1<sup>st</sup> derivative – Non maxima suppression

- Use orientation information

- Algorithm:

- Quantise edge orientations

- For all points (i,j)

- Look at the 2 points orthogonal to edge

- if  $\text{gradient}(i,j) < \text{gradient}(\text{either of these 2 points})$

- output(i,j) = 0

- else output(i,j) = gradient(i,j)

- Now thresholding can be used... or hysteresis thresholding (detailed later...)



Edges

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 12

12

2

### Edge detection – 1<sup>st</sup> derivative – NMS

```

nms_result = gradients.clone();
for (int row=1; row < gradients.rows-1; row++)
    for (int column=1; column < gradients.cols-1; column++)
    {
        float curr_gradient = gradients.at<float>(row,column);
        float curr_orientation = orientations.at<float>(row,column);
        // Determine which neighbours to check
        int direction = (((int) (16.0*(curr_orientation)/(2.0*PI))+15)%8)/2;
        float gradient1 = 0.0, gradient2 = 0.0;
        switch(direction)
        {
            case 0:
                gradient1 = gradients.at<float>(row-1,column-1);
                gradient2 = gradients.at<float>(row+1,column+1);
                break;
        }
    }
}

```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 13

13

### Edge detection – 1<sup>st</sup> derivative – NMS

```

case 1:
gradient1 = gradients.at<float>(row-1,column);
gradient2 = gradients.at<float>(row+1,column);
break;
case 2:
gradient1 = gradients.at<float>(row-1,column+1);
gradient2 = gradients.at<float>(row+1,column-1);
break;
case 3:
gradient1 = gradients.at<float>(row,column+1);
gradient2 = gradients.at<float>(row,column-1);
break;
}
if ((gradient1 > curr_gradient) || (gradient2 > curr_gradient))
nms_result.at<float>(row,column) = 0.0;
}

```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 14

14

### Edge detection –Canny

- Combination of first and second derivative detectors
- Second derivative intended to improve localisation
- Thresholding incorporated so result is binary
- (Optional?) Analysis at multiple scales

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 15

15

### Edge detection – Canny algorithm

- Convolve image with Gaussian
- Compute the first derivative gradients and orientations
- Use the 2<sup>nd</sup> derivative zero-crossing to suppress non-maximas
- Threshold edges with hysteresis
  - Apply a double threshold to find strong & weak edge points.
  - Remove any weak edges points not connected to strong edge points.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 16

16

### Edge detection – Canny – uses 2<sup>nd</sup> derivative?

- According to the original paper Canny is supposed to be a combination of first and second derivative detectors
- Second derivative intended to improve localisation

Uses a filter like the Laplacian (of Gaussian)

$$h(i,j) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Need to find zero-crossings.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 17

17

### Edge detection –Canny

- Also according to the original paper the image is supposed to be analysed at multiple scales
- Smooth image using different Gaussians
- Look for correspondences across scale to identify significant discontinuities

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges Slide 18

### Edge detection – advanced – Canny

```
Canny( gray_image, binary_edges, 100, 200 );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 19

### Multispectral edge detection

- Grey levels are ordered, colours are not.
- Multispectral edge detection is addressed in 3 ways:
  - Output fusion
    - Combine edges
  - Multidimensional gradient methods
    - Combine gradients
  - Vector methods
    - Distance between colours in 3D

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 20

19

20

### Contour Segmentation – Topics

- Edge data representations
- Border Detection
- Line segment extraction

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 21

### Contour Segmentation – Boundary Chain Codes

- Each chain contains
  - Start point
  - A list of orientations

Features!
 

- Orientation & Scale dependent
- Slightly position dependent
- Can be smoothed to reduce boundary noise
- Difficulties obtaining consistent representation from image.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 22

21

22

### Contour Segmentation – Boundary Chain Codes

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 23

### Contour Segmentation – Boundary Chain Codes

```
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
findContours( binary_edge_image, contours, hierarchy,
              CV_RETR_CCOMP, CV_CHAIN_APPROX_NONE );

for (int contour_number=0;
     (contour_number<contours.size()); contour_number++)
{
    Scalar colour( rand()&0xFF, rand()&0xFF, rand()&0xFF );
    drawContours( display_image, contours, contour_number,
                  colour, 1, 8, hierarchy );
}
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 24

23

24

### Contour Segmentation

Represent an edge image as a graph:

- Node  $n_i$ 
  - Corresponds to pixel  $x_i$
  - Associated cost  $s(x_i)$  and orientation  $\phi(x_i)$
- $n_i, n_j$  are connected by an arc if
  - Pixels  $x_i, x_j$  are 8-connected neighbours
  - $\phi(x_i)$  and  $\phi(x_j)$  match the local border:
    - $x_i$  must be one of 3 neighbours in the direction  $[\phi(x_i)-\pi/4, \phi(x_i)+\pi/4]$
    - $|\phi(x_i) - \phi(x_j)| < \pi/2$

When expanding edge chains consider

- Strength of edges
  - E.g.  $(\max_{\text{image}} s(x_k)) - s(x_i)$
- Border curvature
  - $\text{diff}[\phi(x_i) - \phi(x_j)]$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 25

25

### Contour Segmentation

Purpose: Segment all edge chains in an image

Algorithm:

- Search for the strongest (unused) node in the graph. If none go to step 5.
- Expand all the edges in front of the specified edge
- Expand all the edges behind of the specified edge
- If the edge chain consists of > 3 pixels store it. Go to step 1.
- Modify edge chains to fill small breaks and to remove duplicate contours.
- Repeat step 5 until stable.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 26

26

### Contour Segmentation – Segment sequences

Boundary

- Sequence of segments

Segment

- Start and end points:  $x_1$  and  $x_2$
- Type: straight line, some type of curve, etc.
- Other parameters

Accurate polygonal representation

- Must define acceptable tolerances
- Vertices – where?

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 27

27

### Contour Segmentation – Straight line extraction example

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 28

28

### Contour Segmentation – Obtaining polygonal segments

Recursive boundary splitting

- Split at furthest point
- Keep going until with tolerance

```
vector<vector<Point>> approx_polyline(Mat contours.size());
for (int contour_number=0;
     (contour_number<contours.size()); contour_number++)
    approxPolyDP( Mat(contours[contour_number]),
                  approx_polyline[contour_number], 3, true );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 29

29

### Hough transform

Direct transformation from image space to probability of the existence of some feature...

- Lines
- Circles
- Generalised shapes

© John McDonald, National University of Ireland, Maynooth. From J. McDonald, J. Franz and R. Shorten, Application of the Hough Transform to Lane Detection in Motorway Driving Scenarios. Published in Proc. of the Irish Signals and Systems Conference, 2001. R. Shorten, T. Ward, T. Lycaght (Eds). (Reproduced with permission)

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 30

30

### Hough Transform – Line detection

Line equation:

- $j = m \cdot i + c$
- Lines in Hough space
- What about  $i = c$
- $r = i \cdot \cos \theta + j \cdot \sin \theta$
- Sinusoidal curves

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 31

31

### Hough Transform – Line detection

$r = i \cdot \cos \theta + j \cdot \sin \theta$

Transform

- From Image space ( $x, y$ )
- To Hough space ( $r, \theta$ )

Algorithm

- Initialise accumulator to 0
- For every edge point
  - Increment cells in accumulator corresponding to all possible lines
- Search for Maximums

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 32

32

### Hough Transform – Line detection

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 33

33

### Hough Transform – Circle detection

Equation of a circle  $(i-a)^2 + (j-b)^2 = r^2$

- Assume constant radius  $r$

Transform

- From Image space ( $x, y$ )
- To Hough space ( $a, b$ )

Algorithm

- Initialise accumulator to 0
- For every edge point
  - Increment cells in accumulator corresponding to all possible circle centers
- Search for Maximums

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 34

34

### Hough Transform – Circle detection

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 35

35

### Hough transform – in OpenCV

Hough for lines:

```
vector<Vec2f> hough_lines;
HoughLines(binary_edge_image, hough_lines, 1, PI/200.0, 60);
```

Probabilistic Hough for line segments:

```
vector<Vec4i> line_segments;
HoughLinesP(binary_edge_image, line_segments, 1.0,
           PI/200.0, 20, 20, 5);
```

Hough for circles:

```
vector<Vec3f> circles;
HoughCircles(gray_image, circles, CV_HOUGH_GRADIENT,
             2, 20, 300, 20, 5, 15);
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 36

36

## Hough Transform – Resolution of Hough Space

- Circle
  - (a, b) could be anywhere up to r outside the image space
  - Can detect the circle centre to higher precision than the image points
- Lines
  - $-\text{dist}(0,0, M,N) \leq r \leq +\text{dist}(0,0, M,N)$
  - $-\pi \leq \theta \leq \pi$
  - Precision of r and  $\theta$  application dependent

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 37

37

## Hough Transform – Efficient Implementation

- Use edge orientations
  - Restrict the mapping into space
- Use half the accumulator
  - For lines only
- Multi-resolution
  - Process at a small resolution
  - Higher resolution to find accurate data
- Problem
  - What if the size of the circle is unknown
  - 3-D Accumulator??

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Edges

Slide 38

38

## Other techniques for estimating pose (lines ++)

- Common to have a number of data points which may
  - form part of a feature (such as a straight line) OR
  - represent possible matches with some known object features
- We need to determine the best possible position or pose of this feature or object
- Techniques:
  - Least squared error solution
  - RANSAC

Recognition

© Kenneth Dawson-Howe 2015

Slide 39

39

## Least squared error solution

- Linear fit which best matches the data.
- For a straight line it minimises the sum of the vertical residuals

Line equation:  $y = m \cdot x + c$

First compute the slope  $m$

Then the intercept  $c$

Recognition

© Kenneth Dawson-Howe 2015

Slide 40

40

## Least squared error solution

Given  $(x_i, y_i)$  where  $i = 1..N$

$$\mu_x = \frac{1}{N} \sum_{i=1}^N x_i \quad \mu_y = \frac{1}{N} \sum_{i=1}^N y_i$$

$$\sigma_x = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu_x)^2} \quad \sigma_y = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \mu_y)^2}$$

Pearson's correlation coefficient:  $\rho_{xy} = \frac{\text{COV}_{xy}}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y)}{\sigma_x \sigma_y}$

$$m = \rho_{xy} \frac{\sigma_y}{\sigma_x} \quad c = \mu_y - m \cdot \mu_x$$

Recognition

© Kenneth Dawson-Howe 2015

Slide 41

41

## Least squared error solution – Problems

- Assumes line is not vertical
- Assumes that error distribution is normal
- Assumes that the points which should be included in the regression are known (i.e. segmented)
- Assumes no significant outliers

Figure 10.7: Influence of an outlier in least squares line fitting. With 6 valid data points and 1 gross outlier (white), the best line is shown in solid. Least squares, followed by discarding the worst outlier, reaches the dotted line after 3 discards [Fischler and Bolles, 1981]. © Cengage Learning 2015.

© Cengage Learning Engineering. Reproduced with permission. From *Image Processing, Analysis and Machine Vision* by Milan Sonka, Vaclav Hlavac and Roger Boyle

Recognition

© Kenneth Dawson-Howe 2015

Slide 42

42

## Random Sample Consensus (RANSAC)

- Uses the minimum number of data points ( $m$ ) to determine the model
  - For a straight line  $m=2$

### Technique

- Randomly select the minimum number ( $m$ ) of data points from the  $N$  data points  $(x_i, y_i)$  where  $i = 1..N$
- Determine the model from the selected data points
- Determine how many data points are within some tolerance of the model – the consensus set
- If the consensus set is not big enough (i.e. is smaller than some pre-set threshold) go back to step 1 (OR fail if tried too often).
- If the consensus set was big enough, re-compute the model using all points in the consensus set.

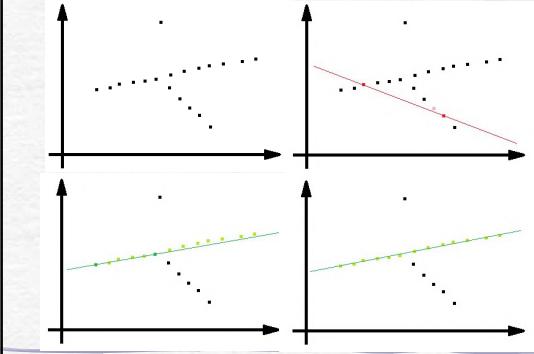
Recognition

© Kenneth Dawson-Howe 2015

Slide 43

43

## RANSAC Example



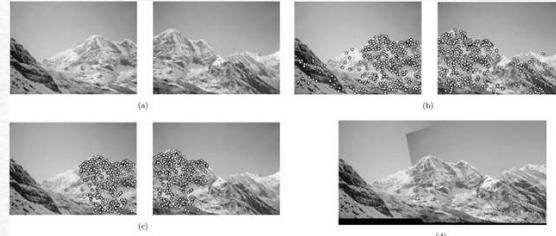
Recognition

© Kenneth Dawson-Howe 2015

Slide 44

44

## RANSAC Example



**Figure 10.8:** Use of RANSAC in panoramic stitching (the original images are, of course, in color): (a) An image pair that overlaps. (b) Each dot represents an ‘interest point’, whose vector matches a point in the other image. (c) The illustrated dots are the RANSAC ‘inliers’: they conform to the hypothesis of the best-performing candidate homography that is tried. (d) Prior to smoothing, the resulting overlap. Courtesy of D. Lowe, M. Brown, University of British Columbia.

© Cengage Learning Engineering. Reproduced with permission. From *Image Processing, Analysis and Machine Vision* by Milan Sonka, Vaclav Hlavac and Roger Boyle

Recognition

Slide 45

45

## RANSAC Examples



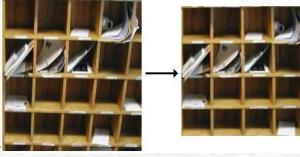
Recognition

Slide 46

46

## Geometric Operations

- Geometric transformations
- Pixel co-ordinate transformations
- Brightness interpolation
- Camera Model



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Geometric

Slide 1

1

## Geometric transformations

- Computer graphics
  - Introduce distortion
- Image processing
  - Image mosaicing
  - Matching/Registering image
  - Eliminating distortion
  - Simplifying further processing
    - e.g. OCR

[Examples](#)

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

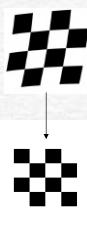
Geometric

Slide 2

2

## Geometric transformations - Specification

- Problem:
  - Distorted image  $f(i, j)$
  - Corrected image  $f'(i', j')$
- Mapping:  $i = T_i(i', j')$      $j = T_j(i', j')$



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

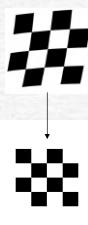
Geometric

Slide 3

3

## Geometric transformations - Specification

- Application steps:
  - Define the transformation
    - Known in advance
    - Determine through correspondences
      - Image to known
      - Image to image
  - Apply the transformation
    - For every point in the output image
      - Determine where it came from using  $T$
      - Interpolate a value for the output point



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Geometric

Slide 4

4

## Output → Input

- A backwards transformation?
- Why aren't we mapping from the input image to the output?
  - For each input point
    - Determine the real coordinates of the location to which it is mapped.
  - For each output point
    - Find the nearest transformed point(s)
    - Interpolate a value from this point(s)

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Geometric

Slide 5

5

## Affine transformations

- Definition
- Known transformations...
  - E.g. Translation
- Unknown transformations...
  - Requires at least 3 observations

```
Mat affine_matrix( 2, 3, CV_32FC1 );
...
warpAffine(image, result, affine_matrix, image.size());
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Geometric

Slide 6

6

## Simple affine transformations

Geometric

- Rotation

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

- Change of scale

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

- Skewing

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 1 & \tan \phi & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 7

7

## More affine transformations

Geometric

- Panoramic distortion

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 8

8

## Unknown affine transformations

Geometric

- Given...

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

- If we have 3 observations...  $(i_1, j_1) \leftrightarrow (i'_1, j'_1)$   
 $(i_2, j_2) \leftrightarrow (i'_2, j'_2)$   
 $(i_3, j_3) \leftrightarrow (i'_3, j'_3)$

- We can reorganise..

$$\begin{bmatrix} i_1 \\ j_1 \\ i_2 \\ j_2 \\ i_3 \\ j_3 \end{bmatrix} = \begin{bmatrix} i'_1 & j'_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & i'_1 & j'_1 & 1 \\ i'_2 & j'_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & i'_2 & j'_2 & 1 \\ i'_3 & j'_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & i'_3 & j'_3 & 1 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{01} \\ a_{02} \\ a_{10} \\ a_{11} \\ a_{12} \end{bmatrix}$$

- And take the inverse to compute the a coefficients

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 9

9

## Example affine transformations

Geometric

Point2f source [3], destination [3];

```
affine_matrix = getAffineTransform(source,destination);
```

- Can we use more observations?

- Better estimate of the a coefficients
- More rows in the matrices
- Must use the pseudo inverse

$$\begin{bmatrix} i_1 \\ j_1 \\ i_2 \\ j_2 \\ i_3 \\ j_3 \end{bmatrix} = \begin{bmatrix} i'_1 & j'_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & i'_1 & j'_1 & 1 \\ i'_2 & j'_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & i'_2 & j'_2 & 1 \\ i'_3 & j'_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & i'_3 & j'_3 & 1 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{01} \\ a_{02} \\ a_{10} \\ a_{11} \\ a_{12} \end{bmatrix}$$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 10

10

## Perspective transformations

Geometric

- Perspective projection

- Planar surface
- Not parallel to the image plane.
  - Cannot be corrected with the affine transformation
- Need a perspective transformation

$$\begin{bmatrix} i.w \\ j.w \\ w \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & 1 \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 11

11

## Perspective transformations – maths

Geometric

- From

$$\begin{bmatrix} i.w \\ j.w \\ w \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & 1 \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

- We know  $i.w = p_{00} \cdot i' + p_{01} \cdot j' + p_{02}$   
 $w = p_{20} \cdot i' + p_{21} \cdot j' + 1$
- Hence  $i = p_{00} \cdot i' + p_{01} \cdot j' + p_{02} - p_{20} \cdot i \cdot i' - p_{21} \cdot i \cdot j'$
- And similarly  $j = p_{10} \cdot i' + p_{11} \cdot j' + p_{12} - p_{20} \cdot j \cdot i' - p_{21} \cdot j \cdot j'$
- If we observe 4 mappings...

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 12

2

### Perspective transformation – combined equations

Then

$$\begin{bmatrix} i_1 \\ j_1 \\ i_2 \\ j_2 \\ i_3 \\ j_3 \\ i_4 \\ j_4 \end{bmatrix} = \begin{bmatrix} i'_1 & j'_1 & 1 & 0 & 0 & 0 & -i_1 i'_1 & -i_1 j'_1 \\ 0 & 0 & 0 & i'_1 & j'_1 & 1 & -j_1 i'_1 & -j_1 j'_1 \\ i'_2 & j'_2 & 1 & 0 & 0 & 0 & -i_2 i'_2 & -i_2 j'_2 \\ 0 & 0 & 0 & i'_2 & j'_2 & 1 & -j_2 i'_2 & -j_2 j'_2 \\ i'_3 & j'_3 & 1 & 0 & 0 & 0 & -i_3 i'_3 & -i_3 j'_3 \\ 0 & 0 & 0 & i'_3 & j'_3 & 1 & -j_3 i'_3 & -j_3 j'_3 \\ i'_4 & j'_4 & 1 & 0 & 0 & 0 & -i_4 i'_4 & -i_4 j'_4 \\ 0 & 0 & 0 & i'_4 & j'_4 & 1 & -j_4 i'_4 & -j_4 j'_4 \end{bmatrix} \begin{bmatrix} p_{00} \\ p_{01} \\ p_{02} \\ p_{10} \\ p_{11} \\ p_{12} \\ p_{20} \\ p_{21} \end{bmatrix}$$

- Multiply by the inverse of the square matrix...
- More observations gives
  - A more accurate transformation
  - Matrix becomes non-square...
  - Pseudo inverse required

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Geometric

Slide 13

13

### Perspective transformations – example



```
Point2f source [4], destination [4];
// Assign values to source and destination points.
perspective_matrix = getPerspectiveTransform( source, destination );
warpPerspective( image, result, perspective_matrix,
result.size() );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Geometric

Slide 14

14

### More complex transformations

- Approximation by a polynomial

$$i = T_i(i', j') = a_{00} + a_{10}i' + a_{01}j' + a_{11}i'j' + a_{02}(j')^2 + a_{20}(i')^2 + a_{12}i'(j')^2 + a_{21}(i')^2j' + a_{22}(i')^2(j')^2 + \dots$$

$$j = T_j(i', j') = b_{00} + b_{10}i' + b_{01}j' + b_{11}i'j' + b_{02}(j')^2 + b_{20}(i')^2 + b_{12}i'(j')^2 + b_{21}(i')^2j' + b_{22}(i')^2(j')^2 + \dots$$

- No. of correspondences.
- Distribution of points
- Solve simultaneous linear equations
- Least squared error solution

- Even more complex
  - Partition the image.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Geometric

Slide 15

15

### Brightness interpolation

- Locations are not integer coordinates

- Interpolate output pixel value from
  - The nearby pixels in the original (uncorrected) image

- Loss of accuracy
  - Complexity of interpolation scheme

- Interpolation schemes:
  - Nearest neighbour
  - Bilinear interpolation
  - Bicubic interpolation

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Geometric

Slide 16

16

### Nearest neighborhood interpolation

- Simple formulation:  $f'(i', j') = f(\text{round}(i), \text{round}(j))$

- Error is perceptible
  - Step edges



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Geometric

Slide 17

17

### Bi-linear interpolation

$$f'(i', j') = (\text{trunc}(i) + 1 - i)(\text{trunc}(j) + 1 - j)f(\text{trunc}(i), \text{trunc}(j)) + (i - \text{trunc}(i))(\text{trunc}(j) + 1 - j)f(\text{trunc}(i) + 1, \text{trunc}(j)) + (\text{trunc}(i) + 1 - i)(j - \text{trunc}(j))f(\text{trunc}(i), \text{trunc}(j) + 1) + (i - \text{trunc}(i))(j - \text{trunc}(j))f(\text{trunc}(i) + 1, \text{trunc}(j) + 1)$$

- Brightness function is bilinear
- Four neighbours contribute

- Error is perceptible
  - Blurring



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

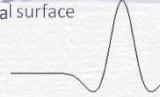
Geometric

Slide 18

18

## Bi-cubic interpolation

- Approximate using a bi-cubic polynomial surface
- No step-like boundary problems
- Copes with linear interpolation blurring
- Often used in raster displays...



Geometric

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 19

19

## Brightness interpolation – OpenCV code

```
int interpolation_scheme = INTER_LINEAR;
// Nearest neighbour interpolation: INTER_NEAREST
// Bilinear interpolation: INTER_LINEAR
// Bicubic interpolation: INTER_CUBIC
warpAffine( image, result, affine_matrix, result_size,
            interpolation_scheme );
warpPerspective( image, result, perspective_matrix,
                  result_size, interpolation_scheme );
```

Geometric  
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

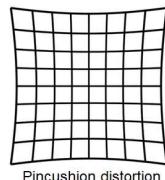
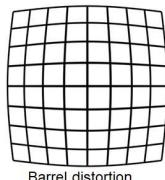
Slide 20

20

## Camera models – Distortion

### Radial distortion

- Radially symmetric from the optical axis
  - Change in magnification
    - Barrel distortion
    - Pincushion distortion
  - Caused by the lenses
- $$i' = i(1 + k_1r^2 + k_2r^4 + k_3r^6)$$
- $$j' = j(1 + k_1r^2 + k_2r^4 + k_3r^6)$$
- $$r = \sqrt{i^2 + j^2}$$



Geometric

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 21

21

## Camera models – Distortion

### Tangential distortion

- Uneven magnification from one side to the other
- Lenses not parallel to the image plane

$$i' = i + (2p_1ij + p_2(r^2 + 2i^2))$$

$$j' = j + (2p_2ij + p_1(r^2 + 2j^2))$$

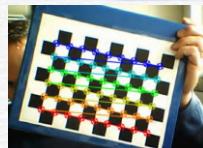
Geometric  
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 22

22

## Camera models – Removing distortion

- Calibrate using multiple images
  - Known object
  - Different positions
  - Computes camera matrix
  - & distortion parameters
- Can be used to remove distortion
  - Mainly using the distortion parameters



```
calibrateCamera( object_points, image_points,
                 image_size, camera_matrix, distortion_coefficients,
                 rotation_vectors, translation_vectors );
...
undistort( camera_image, corrected_image,
           camera_matrix, distortion_coefficients );
```

Geometric

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 23

23

## Video

- Introduction
  - Static Background
- Background Models
  - Median
  - Gaussian Mixture Model
  - Shadow Detection
- Tracking
  - Exhaustive Search, Mean Shift, Optical Flow, Feature Point Tracking

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 1

1

## Introduction – Objects of interest

- Reliably detecting moving objects of interest in a scene.

1. Motion detection
2. Moving object detection & location
3. Derivation of 3D object properties

- When is an object of interest?
  - Size
  - Max and min velocity and acceleration
  - Assumptions:
    - Mutual correspondence
    - Common motion

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 2

2

## Introduction – Common problems

- Illumination & appearance changes
  - Gradual (e.g. time of day)
  - Sudden (e.g. clouds, lights)
  - Shadows
  - Weather (e.g. rain, snow)
- Background changes
  - Objects becoming part of the background
  - Objects leaving the background
  - Background objects oscillating slightly
- Setup
  - Camera motion
  - Frame rate
  - Field of view
  - Distance to objects
  - Location of camera

© Ahmed Elgammal (Reproduced with permission)  
from [ftp://www.umiacs.umd.edu/pub/elgammal/video/index.html](http://www.umiacs.umd.edu/pub/elgammal/video/index.html)

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 3

3

## Introduction – Static background



- Image subtraction  $d(i,j) = |f_k(i,j) - b(i,j)|$ 
  - Difference of current frame and background image..
  - Background  $b(i,j)$  = first frame?

`absdiff( frame, background, difference );`

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 4

4

## Introduction – Thresholding

- Threshold result

$$d(i,j) = \begin{cases} 0 & \text{if } |f_k(i,j) - b(i,j)| < T \\ 1 & \text{otherwise} \end{cases}$$


- Threshold T sensitivity?
  - Too high → False negatives
  - Too low → False positives
  - Just right??
- High contrast required.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 5

5

## Introduction – Colour

- But how do we deal with colour?

1. Average difference of all channels?
2. Just process hue channel?
  - What about white/grey/black?
3. Threshold channels separately
  - Any one channel or all channels?
  - Which colour model?



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 6

1



## Background Models – GMM Model Update

- For a new sample  $f_n(i,j)$
  - Select the best close Gaussian distribution
    - $\text{close}$  = within  $2.5 \sigma_n(i,j,m)$  of  $\mu_n(i,j,m)$
  - If there is a best close Gaussian /
- $$\pi_{n+1}(i,j,m) = \alpha * O_n(i,j,m) + (1-\alpha) * \pi_n(i,j,m)$$
- where  $O_n(i,j,m) = 1$  for the close Gaussian distribution and 0 otherwise
- $$\mu_{n+1}(i,j,m) = \mu_n(i,j,m) + O_n(i,j,m) * (\alpha / \pi_{n+1}(i,j,m)) * (f_n(i,j) - \mu_n(i,j,m))$$
- $$\sigma^2_{n+1}(i,j,m) = \sigma^2_n(i,j,m) +$$
- $$O_n(i,j,m) * (\alpha / \pi_{n+1}(i,j,m)) * ((f_n(i,j) - \mu_n(i,j,m))^2 - \sigma^2_n(i,j,m))$$
- If there is no close Gaussian (replace one...)
    - $x = \arg\min_m(\pi_n(i,j,m))$
    - $\mu_{n+1}(i,j,x) = f_n(i,j)$
    - $\sigma^2_{n+1}(i,j,x) = 2 \cdot \max_m \sigma^2_n(i,j,m)$
    - $\pi_{n+1}(i,j,x) = \frac{1}{N} \cdot \min_m \pi_n(i,j,m)$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 13

Video

## Background Models – GMM Moving Points

- Identifying background distributions
  - Define  $T$ , a proportion of the frames in which background pixels should be visible.
  - Order the Gaussians by  $\pi_{n+1}(i,j,m) / \sigma_{n+1}(i,j,m)$
  - Gaussians 1... $B$  are considered background where
  - $B = \arg\min_B (\sum_{m=1..M} \pi_{n+1}(i,j,m)) > T$
- Just check if best close Gaussian (or the new Gaussian distribution) is a background distribution
- Finally use morphological dilations and erosions to remove small regions and fill in holes.



© Reproduced by permission of Dr. James Ferryman, University of Reading

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 14

13

## Background Models – GMM issues

- GMM issues:
- may fail under fast variations,
  - low sensitivity around Gaussian tails,
  - less frequent events produce low probability & high variance,
  - needs to compute floating point probabilities.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 15

Video

## Background Models – Shadow Detection

$$SP_k(i,j) = \begin{cases} 1 & \dots \text{if } \left( \alpha < \frac{f_k^S(i,j)}{B_k^S(i,j)} < \beta \right) \text{ and } \left( |f_k^S(i,j) - B_k^S(i,j)| < \tau_S \right) \text{ and } \left( |f_k^H(i,j) - B_k^H(i,j)| < \tau_H \right) \\ 0 & \dots \text{otherwise} \end{cases}$$

Compare current frame with background image... (Prati 2003)



- Intensity / luminance / value drops
- Intensity / luminance / value has not dropped too much
- Saturation does not increase too much
- Hue does not change too much

Hue unpredictable & Change in Luminance can be small (Tattersall 2003)

$$SP_k(i,j) = \begin{cases} 1 & \dots \text{if } \left( \lambda < \frac{f_k^S(i,j)}{B_k^S(i,j)} < 1.0 \right) \text{ and } \left( |f_k^S(i,j) - B_k^S(i,j)| < \tau_S \right) \\ 0 & \dots \text{otherwise} \end{cases}$$

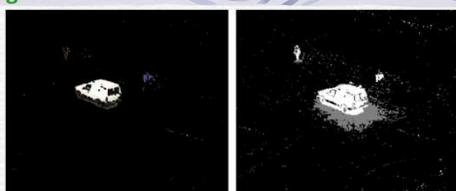
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 16

15

16

## Background Models – Shadow Detection



© Reproduced by permission of Dr. James Ferryman, University of Reading

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 17

Video

## Background Models – Shadow Detection – Code

```
Ptr<BackgroundSubtractorMOG> gmm =
    createBackgroundSubtractorMOG2();
gmm->apply(current_frame, foreground_mask);
gmm->getBackgroundImage( mean_background_image );
threshold( foreground_mask, moving_points, 150, 255, THRESH_BINARY );
...
threshold( foreground_mask, changing_points, 50, 255, THRESH_BINARY );
absdiff( moving_points, changing_points, shadow_points );
...
Mat mean_background_image;
gmm->getBackgroundImage( mean_background_image );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 18

17

18

## Tracking – Introduction

- Used in video surveillance, sports video analysis, vehicle guidance systems, etc.
- A hard task because objects
  - may be undergoing complex motion
  - may change shape
  - may be occluded
  - may change appearance due to lighting/weather
  - may physically change appearance

Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 19

19

## Tracking – Topics

- Exhaustive search
- Mean Shift
- Optical Flow
- Feature based tracking

Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 20

20

## Tracking – Exhaustive Search

- Extract object to be tracked from frame
- Compare in all possible positions in future frame(s)
  - Use a similarity metric
    - E.g. normalised cross correlation
  - Just pick the best match?
- Need extra degrees of freedom for scale and orientation.
- May fail if object motion is too complex.
- Template matching and chamfer matching support this type of tracking.

Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 21

21

## Tracking – Mean Shift

- Tracks “objects” in video by
  - Searching *locally* for the most similar region.
  - Using a histogram to represent the “object”.
  - Using (iterative) gradient ascent to locate the best match.
- Mean shift applied to tracking in 2002 by Comaniciu and Meer.
  - Also used for segmentation.
- Histogram of
  - Colours
  - Oriented gradients
  - Texture
  - etc.



Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 22

22

## Tracking – Mean Shift

- Model the “object”
 
$$\hat{q}_u \triangleq C \sum_{i=1}^n k \left( \frac{\|x_i\|^2}{h_q} \right) \delta [b(x_i) - u]$$
  - Probability Density Function (histogram) of colours.
    - Limit the number of bins
  - Typically an elliptical region is used.
  - Weight the values relative to their location.
    - Epanechnikov kernel



Video

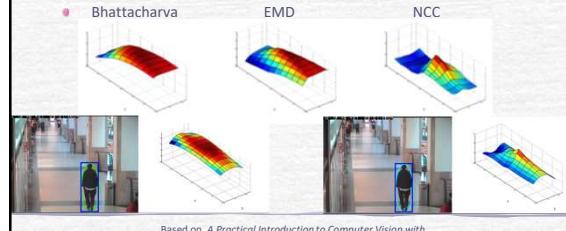
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 23

23

## Tracking – Mean Shift

- Model candidate regions...
 
$$\hat{p}_u(y) \triangleq C_h \sum_{i=1}^{n_h} k \left( \frac{\|y - x_i\|^2}{h_p} \right) \delta [b(x_i) - u]$$
- Matching to find the best new location...
  - Compare distributions directly.
  - Move to the mode in matching space.
  - Bhattacharva EMD NCC



Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 24

24

## Tracking – Mean Shift

### Mean shift approach

- Consider the gradient of the similarity function (the Bhattacharyya coefficient)...
- Gradient of superposition of kernels, centered at each data point is equivalent to convolving the data points with the gradient of the kernel

$$w_i = \sum_{u=1}^m \sqrt{\frac{q_u}{p_u(y_0)}} \delta [b(x_i) - u] \quad y_1 = \frac{\sum_{i=1}^{n_h} x_i w_i}{\sum_{i=1}^{n_h} w_i}$$

- Derived from the Bhattacharyya similarity measure
- Assumes Epanechnikov kernels
- Move in the direction of the highest gradient

### Iterate until convergence

- Separation between  $y_0$  and  $y_1$  less than ε

Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 25

## Tracking – Mean Shift – Test dataset

### CAVIAR (14) and PETS (7) tracking scenarios



Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

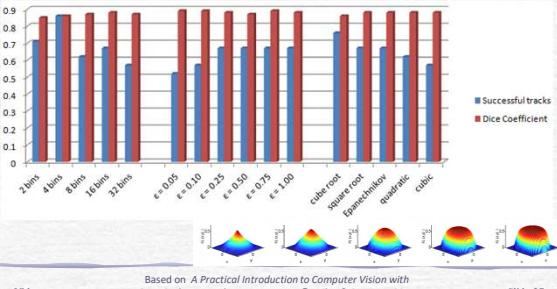
Slide 26

25

26

## Tracking – Mean Shift – Changing parameters

- Number of bins per channel
- Convergence parameter ( $\epsilon$ )
- Kernel type



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 27

Video

## Tracking – Mean Shift – Background Exclusion

- If a background model of the scene is available
  - Favour image regions which are similar to the object model
  - AND dissimilar to the corresponding background region.



### Analogous to background subtraction

- No image differencing.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

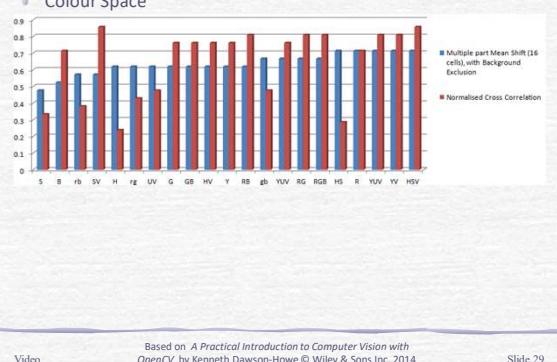
Slide 28

27

28

## Tracking – Mean Shift – The best technique?

### Colour Space



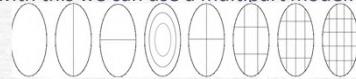
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 29

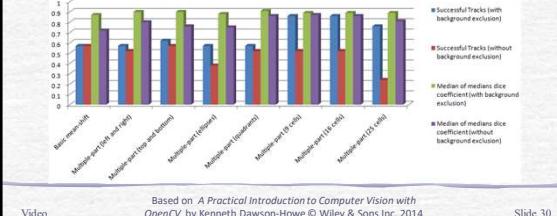
Video

## Tracking – Mean Shift – Multipart models

- Histograms have a lack of spatial structure
- To deal with this we can use a multipart model:



### Improvements:



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 30

29

30

## Tracking – Mean Shift – In OpenCV

- Back-project a histogram of the object into the current frame
- Searches for a region of the same size within the back projection looking for the highest (weighted) sum.



Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 31

## Tracking – Mean Shift – In OpenCV

```
Rect position(starting_position);
TermCriteria criteria( cv::TermCriteria::MAX_ITER, 5, 0.01 );
meanShift( back_projection_probabilities, position, criteria);
```



Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

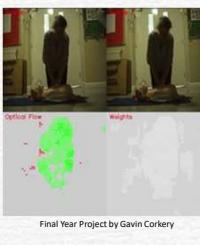
Slide 32

31

32

## Tracking – Dense Optical Flow

- Compute a motion field (known as optical flow) for the entire image
- Direction & Magnitude



Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 33

## Tracking – Dense Optical Flow

- Based on the brightness constancy constraint
  - Object points will have the same brightness over a short period of time  $f_t(i, j) = f_{t+\Delta t}(i + \Delta i, j + \Delta j)$

- Need to find the displacement  $(\Delta i, \Delta j)$  which will minimise the residual error

$$\varepsilon(\Delta i, \Delta j) = \sum_{i=i_{current}-w}^{i_{current}+w} \sum_{j=j_{current}-w}^{j_{current}+w} f_t(i, j) - f_{t+\Delta t}(i + \Delta i, j + \Delta j)$$

Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 34

33

34

## Tracking – Dense Optical Flow

- To compute the optical flow  $\left(\frac{\Delta i}{\Delta t}, \frac{\Delta j}{\Delta t}\right)$ , assuming that the displacement is small...
 
$$f_{t+\Delta t}(i + \Delta i, j + \Delta j) = f_t(i, j) + \frac{\partial f}{\partial t} \Delta t + \frac{\partial f}{\partial i} \Delta i + \frac{\partial f}{\partial j} \Delta j$$

- Hence (given our previous equation  $f_t(i, j) = f_{t+\Delta t}(i + \Delta i, j + \Delta j)$ )
 
$$\frac{\partial f}{\partial i} \Delta i + \frac{\partial f}{\partial j} \Delta j + \frac{\partial f}{\partial t} \Delta t = 0$$

- So
 
$$\frac{\partial f}{\partial i} \Delta t + \frac{\partial f}{\partial j} \Delta t + \frac{\partial f}{\partial t} = 0$$

- And reorganising

$$\begin{bmatrix} \frac{\partial f}{\partial i} & \frac{\partial f}{\partial j} \end{bmatrix} \begin{bmatrix} \frac{\Delta i}{\Delta t} \\ \frac{\Delta j}{\Delta t} \end{bmatrix} = -\frac{\partial f}{\partial t}$$

Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 35

## Tracking – Dense Optical Flow – Code

```
calcOpticalFlowFarneback (previous_gray_frame, gray_frame,
                           optical_flow, 0.5, 3, 15, 3, 5, 1.2, 0);
cvtColor (previous_gray_frame, display, CV_GRAY2BGR);
for (int row = 4; row < display.rows; row+=8)
    for(int column = 4; column < display.cols; column+=8)
    {
        Point2f& flow = optical_flow.at<Point2f>(row,column);
        line (display, Point(column,row), Point(
            cvRound(column+flow.x), cvRound(row+flow.y)),
            passed_colour);
    }
gmm.getBackgroundImage( mean_background_image );
```

Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 36

35

36

## Tracking – Dense Optical Flow – Problems

- ➊ Optical flow is the apparent motion of points within a scene.
  - Based on brightness patterns
  - Needs texture
- ➋ What happens if the brightness changes?
  - i.e. brightness constancy does not hold.
- ➌ Perhaps look at optical flow in gradient space.
- ➋ What if a point moves differently to all its neighbours?
- ➌ Use Region based optical flow

Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 37

## Tracking – Dense Optical Flow – Problems

- ➋ What happens to a rotating sphere? What about a “barber pole”?



- ➌ We get the wrong motion. i.e. it fails.

- ➋ What happens if the motion is too large?

- ➌ Use Iterative Refinement (Lucas-Kanade)

Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 38

## Tracking – Feature-based Optical Flow

- ➊ We cannot accurately compute optical flow for constant regions or along edges.
- ➋ Often better to compute optical flow just for features... (e.g. Lucas Kanade feature tracker)...



© Reproduced by permission of Dr. James Ferryman, University of Reading

Video

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 39

39

## Learning & Evaluation

- Data / Experience
  - Unsupervised
  - Supervised
- Performance Evaluation
  - Ground Truth
  - Metrics
  - Training vs. Validation

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 1

1

## Machine learning

An algorithm that can learn from data.

- Data / Experience
  - Unsupervised / Unlabelled.
  - Supervised / Labelled.
- Some task
  - Often classification
- Performance improved by providing more data
  - Measure accuracy or error rate

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 2

2

## Unsupervised machine learning

### k-means.

- Data – Unlabelled image data
- Task – Grouping pixels to reduce the number of colours
- Performance – Very application/image dependent



© Reproduced by permission of Dr. James Ferryman, University of Reading

### Gaussian Mixture Model

- Data – Unlabelled video data.
- Task – Identification of moving objects.
- Performance
  - Accuracy based on hand labelled ground truth.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 3

3

## Supervised machine learning

### Back-projection.

- Data – Labelled sample pixel values
- Task – Classifying pixels.
- Performance
  - Accuracy based on hand labelled ground truth.



### Statistical Pattern Recognition / Support Vector Machines

- Data – Labelled samples (feature vectors).
- Task – Recognition of objects/shapes.
- Performance
  - Accuracy based on hand labelled ground truth.



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 4

4

## Evaluating Computer Vision Performance

- Does it work in real time?
  - What is real time?
  - Does it matter?
- What is the correct answer?
  - Ground truth.
  - How do we generate ground truth?
- How do we assess how well we have done?
  - Success/failure metrics?
  - Overlap?
  - How do we deal with changing results over time?

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 5

5

## Performance – Ground Truth

- What should any computer vision system achieve?
  - The correct answer is easy for some tasks. e.g.
    - Is this picture me? →
    - Determine the state of the traffic lights. →
- For some tasks it is harder.
  - Classify the object in this image. →
  - Count the people. →
  - Drive from A to B. →

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 6

6

1

### Performance – Ground Truth

- What should any computer vision technique achieve?
  - The correct answer is easy for some techniques. e.g.
    - Find skin pixels
  - For some techniques it is harder.
    - Find the edges.
- The usefulness of a technique depends on the application...
  -

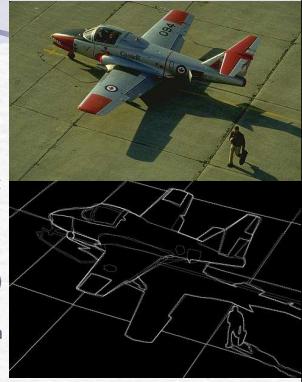
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 7

7

### Ground Truth – Example

- How should an image be segmented?
  - Significant edges/regions?
  - Impossible to get agreement?
- Berkeley segmentation dataset
  - 1,000 images
  - 12,000 segmentations (colour & grayscale)
  - Hand-labelled (30 subjects)
- Is this a good way to evaluate a segmentation technique?



From the public "Berkeley Segmentation dataset"

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 8

8

### Ground Truth – Bounding boxes

- Boxes incorporate parts of other objects/background.
- Labour intensive for video.
- CAVIAR:
  - Shopping centre (Lisbon)
  - Bounding boxes for all people.
  - Some have head, hands, and feet annotated.
  - Manually annotated.

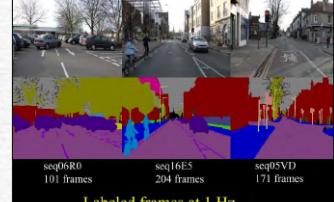
From the public dataset CAVIAR

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 9

9

### Ground Truth – Pixel based

- Better validity.
- Very expensive to create.
- CAMVID:
  - 32 classes
  - 10 minutes @30/15Hz
  - 700 frames
  - Manually annotated.
  - 2<sup>nd</sup> person inspection.

Labeled frames at 1 Hz

From the public dataset CAMVID

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 10

10

### Performance – Simple Metrics

- Compute
  - TP
  - TN
  - FP
  - FN
- and then compute...
 
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total Samples}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}}$$

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

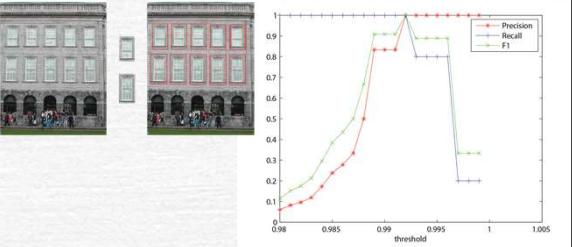
Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 11

11

### Performance – Tuning

- We can tune performance by altering parameters
  - For example altering the threshold used for accepting a match in a template matching example



Precision  
Recall  
F1

threshold

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 12

12

2

## Performance – Precision-Recall Curves

- Alternative (common) visualisation
- Select the parameters which result in the point on the PR curve which is closest to P=1.0, R=1.0

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 13

13

## Performance – Overlap & mAP

- Intersection over Union (IoU)

$$IoU(Detected, GroundTruth) = \frac{\text{Detected} \cap \text{GroundTruth}}{\text{Detected} \cup \text{GroundTruth}}$$

- Values of IoU > 0.5 are often considered good matches.

When considering multiple classes...

- Mean Average Precision (mAP)

$$mAP = \frac{1}{|\text{classes}|} \sum_{\text{classes}} \frac{\#TP(c)}{\#TP(c) + \#FP(c)}$$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 14

14

## Performance – Confusion Matrix

- It is helpful to know how things are misclassified
- Actual class
- Determined class

	Ground Truth				
Result	Walking	Running	Standing	Dancing	Hopping
Walking	0.9	0.0	0.0	0.0	0.0
Running	0.0	0.7	0.0	0.1	0.1
Standing	0.3	0.0	1.0	0.0	0.0
Dancing	0.0	0.2	0.0	0.4	0.0
Hopping	0.0	0.1	0.0	0.5	0.9

- If considering a binary classification
- Confusion matrix shows TP, FP, FN, TN

	A	Not A
A	TP	FP
Not A	FN	TN
Face	3	0
Not Face	1	65532

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 15

15

## Training & Validation

- Typically datasets are divided into two parts:
- Training set.
- Validation/Testing set.

- This can be
- Static
- Random
- Leave p-out cross validation
  - Leave p samples out – for testing. Train on the rest.
  - Do multiple rounds of training and testing using different partitions.
  - Average results.
  - Intended to determine how the results will generalise.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 16

16

## Outcome

- So what does testing tell me?
  - How well the system/technique works on the test data (images/videos)...
  - Actually, how close the system/technique gets to the manually created ground truth...
- How representative is the test set?
  - What conditions matter?
  - Can we ever be sure that a system will work in general?

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 17

17

## Summary

- Learning (Data – Task – Performance)
  - Unsupervised
  - Supervised
- Ground Truth
  - Expensive to obtain.
  - Often unclear what is the right answer.
- Metrics
  - Precision, Recall, F1-measure
  - Tuning
  - IoU, mAP, Confusion matrix
- Training vs. Validation
  - Leave p-out cross validation

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Learning & Evaluation Slide 18

18

# Features

- Introduction
- Moravec Corner Detection
- Harris/Plessey Corner Detection
- Scale Invariant Feature Transform (SIFT)

Features

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Slide 1

1

Given two images (a) and (b) taken at different times determine the movement of edge points from frame to frame...

(a)

(b)

(c)

2

The image contains five diagrams labeled (a) through (e).

- (a) A circle divided horizontally into two equal halves: white on top and dark red on bottom.
- (b) The same circle, but the top half is dark red and the bottom half is white.
- (c) A circle divided vertically into two equal halves: white on left and dark red on right. A yellow question mark is located near the center.
- (d) A circle centered over a dark red rectangle. Three yellow arrows point from the circle's circumference to the rectangle's surface.
- (e) A circle centered over a dark red rectangle. Three yellow arrows point from the circle's circumference to the rectangle's surface. A green line segment connects the center of the circle to the center of the rectangle.

Below the diagrams, the text "The Aperture Problem." is displayed.

3

## Introduction – Using Features / Corners instead

- Use corners / image features / interest points
  - Corner = intersection of two edges
  - Interest point = any feature which can be robustly detected
- 
- Reduces number of points
- Easier to establish correspondences
- Spurious features

4

## Introduction – Steps for corner detection

1. Determine cornerness values.
  - For each pixel
  - Main difference
  - Produces a Cornerness map.
2. Non-maxima suppression.
  - Multiple responses
  - Compare to local neighbours
3. Threshold the cornerness map.
  - Significant corners.



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features

Slide 5

5

The figure illustrates the Moravec corner detection process. It consists of three panels: the original image of a license plate with the number '03' and the letters 'IRL'; a processed image showing a grid of local patches; and the final result where the corners of the digits are highlighted in red.

6

### Moravec – Binary Example

Corner:

Edge:

Minimum difference: 2      Minimum difference: 0

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 7

7

### Moravec – Flaws.

- Anisotropic response
- Diagonal lines
- Smoothing

- Noisy response
- Larger area
- Smoothing

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 8

8

### Harris / Plessey corner detection

- Difference – Cornerness determination
- Uses
  - Partial derivatives
  - Gaussian weighting
  - Matrix Eigenvalues

```
Ptr<FeatureDetector> harris_detector =
    GFTTDetector::create( 1000, 0.01, 10, 3, true );
vector<KeyPoint> keypoints;
harris_detector->detect( gray_image, keypoints );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 9

9

### Harris / Plessey corner detection

- Consider the intensity variation for an arbitrary shift  $(\Delta i, \Delta j)$  as
$$SSD_W(\Delta i, \Delta j) = \sum_{(i,j) \in W} (f(i,j) - f(i - \Delta i, j - \Delta j))^2$$
- If  $f(i - \Delta i, j - \Delta j) \approx f(i,j) + \left[ \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} \right] [\Delta i] [\Delta j]$
- Then  $SSD_W(\Delta i, \Delta j) = \sum_{(i,j) \in W} \left( f(i,j) - f(i,j) - \left[ \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} \right] [\Delta i] \right)^2$

$$SSD_W(\Delta i, \Delta j) = \sum_{(i,j) \in W} \left( \left[ \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} \right] [\Delta i] \right)^2$$

$$SSD_W(\Delta i, \Delta j) = \sum_{(i,j) \in W} \left( \begin{bmatrix} \delta f(i,j) \\ \delta f(i,j) \end{bmatrix} \begin{bmatrix} \delta f(i,j) \\ \delta f(i,j) \end{bmatrix}^\top \right) \begin{bmatrix} [\Delta i] \\ [\Delta j] \end{bmatrix}$$

$$SSD_W(\Delta i, \Delta j) = [\Delta i] \begin{bmatrix} \sum_{(i,j) \in W} \left( \frac{\delta f(i,j)}{\delta i} \right)^2 & \sum_{(i,j) \in W} \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} \\ \sum_{(i,j) \in W} \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} & \sum_{(i,j) \in W} \left( \frac{\delta f(i,j)}{\delta j} \right)^2 \end{bmatrix} \begin{bmatrix} [\Delta i] \\ [\Delta j] \end{bmatrix}$$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 10

10

### Harris / Plessey corner detection

- From the matrix we can compute the Eigenvalues
  - Both high => corner
  - One high => edge
  - None high => constant region
- Harris & Stephens proposed the following cornerness measure:
$$M = \begin{bmatrix} \sum_{(i,j) \in W} \left( \frac{\delta f(i,j)}{\delta i} \right)^2 & \sum_{(i,j) \in W} \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} \\ \sum_{(i,j) \in W} \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} & \sum_{(i,j) \in W} \left( \frac{\delta f(i,j)}{\delta j} \right)^2 \end{bmatrix} = \begin{bmatrix} 0.04 & 0.12 & 0.04 \\ 0.12 & 0.36 & 0.12 \\ 0.04 & 0.12 & 0.04 \end{bmatrix}$$

$$\det(M) = \lambda_1 \lambda_2 = AC + B^2$$

$$\text{trace}(M) = \lambda_1 + \lambda_2 = A + C$$

$$C(i,j) = \det(M) - k(\text{trace}(M))^2$$

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 11

11

### Harris / Plessey corner detection

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

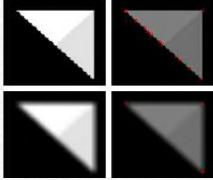
Features      Slide 12

12

2

## Harris / Plessey – Pros and Cons

- Cons:**
  - More expensive computationally
  - Sensitive to noise
  - Somewhat anisotropic
- Pros:**
  - Very repeatable response
  - Better detection rate



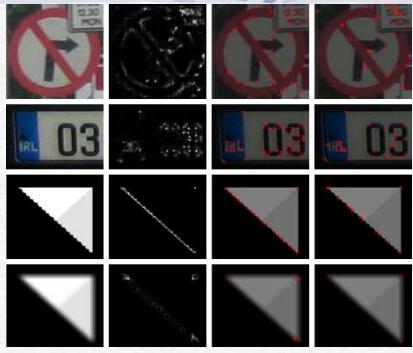
```
Mat display_image;
drawKeypoints( image, keypoints, display_image,
Scalar( 0, 0, 255 ) );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 13

13

## Moravec & Harris/Plessey



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 14

14

## Scale Invariant Feature Transform (SIFT)

- FROM “Distinctive Image Features from Scale-Invariant Keypoints”,  
by David G. Lowe in International Journal of Computer Vision, 60, 2 (2004), pp.91-110
- Motivation:**
  - Providing repeatable robust features for
    - Tracking,
    - Recognition,
    - Panorama Stitching, etc.
- Features:**
  - Invariant to scaling & rotation.
  - Partly invariant to illumination and viewpoint changes



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 15

15

## SIFT – Overview & Contents

- Scale Space Extrema Detection
  - Scale Space
  - Difference of Gaussian
  - Locate Extrema
- Accurate Keypoint Location
  - Sub-pixel locate
  - Filter response – remove low contrast and features primarily along an edge
- Keypoint Orientation assignment
- Keypoint Descriptors
- Matching Descriptors – including dropping poor ones
- Recognition

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 16

16

## SIFT – OpenCV code

```
// The following code is valid up to v2.4.11:
Ptr<FeatureDetector> feature_detector =
    FeatureDetector::create("SIFT");
vector<KeyPoint> keypoints;
feature_detector->detect( gray_image, keypoints );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 17

17

## SIFT – 1. Scale Space Extrema Detection

- For scale invariance, consider the image at multiple scales
 
$$L(x,y,\sigma) \quad L(x,y,k\sigma) \quad L(x,y,k^2\sigma) \quad L(x,y,k^3\sigma)$$

- $L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$
- Applied in different octaves of scale space
- Each octave corresponds to a doubling of  $\sigma$ 


Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 18

18

### SIFT – 1. Space Extrema Detection

- Stable keypoint locations defined to be at extrema in the Difference of Gaussian (DoG) functions across scale space...
- $D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma)$
- Extrema...
  - Centre point is Min or Max of Local 3x3 region in current DoG and in adjacent scales
  - IN any octave

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 19

19

### SIFT – 2. Accurate Keypoint Location

- Originally location and scale taken from central point
- Locate keypoints more precisely
  - Model data locally using a 3D quadratic
  - Locate interpolated maximum/minimum



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 20

20

### SIFT – 2. Accurate Keypoint Location

- Discard low contrast keypoints
  - If the local contrast is too low discard the keypoint
  - Evaluated from the curvature of the 3D quadratic...



Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 21

21

### SIFT – 2. Accurate Keypoint Location

- Discard poorly localised keypoints (e.g. along an edge)

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad \text{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta$$

$$\text{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta \quad \alpha = r\beta$$

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$


Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 22

22

### SIFT – 3. Keypoint Orientation

- For scale invariance, the keypoint scale is used to select the smoothed image with the closest scale
- For orientation invariance we describe the keypoint wrt. the principal orientation
  - Create an orientation histogram (36 bins)
  - $m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$
  - $\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y)))$
  - Weight by gradient magnitude
  - Sample points around the keypoint
  - Highest peak + peaks within 80%
    - Oriented keypoint(s)
  - Stable results when considered under varying noise.

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 23

23

### SIFT – 4. Keypoint Description

- Could sample image intensity at relevant scale
  - Match using normalized cross correlation
- Sensitive to
  - affine transformations,
  - 3D viewpoint changes and
  - non-rigid deformations
- A better approach (Edelman et al. 1997)
  - Based on a model of biological vision
  - Consider gradients at particular orientations and spatial frequencies
  - Location not required to be precise

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features      Slide 24

24

### SIFT – 4. Keypoint Description

- Use blurred image at closest scale
- Sample points around the keypoint
- Compute gradients and orientations
- Rotate by keypoint orientation
- Divide region into subregions
- Create histograms (8 bins) for subregions
  - Weight by gradient
  - Weight by location (Gaussian)
  - Distribute into bins (trilinear interpolation)

Images derived from those by ©David Lowe reproduced with permission

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features

Slide 25

25

### SIFT – 5. Matching Keypoints

- Nearest neighbour matching
  - Euclidean distance between keypoints
  - Use a global distance threshold?
  - Better to...
    - Compare distance to closest neighbour to distance to 2<sup>nd</sup> closest neighbour (from a different object)
    - Distance ratio > 0.8 eliminates
      - 90% false matches
      - 5% correct matches

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features

Slide 26

26

### SIFT – 5. Matching Keypoints

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features

Slide 27

27

### SIFT – OpenCV code

```
// The following code is valid up to v2.4.11:
SiftFeatureDetector sift_detector;
vector<KeyPoint> keypoints1, keypoints2;
sift_detector.detect( gray_image1, keypoints1 );
sift_detector.detect( gray_image2, keypoints2 );
// Extract feature descriptors
SiftDescriptorExtractor sift_extractor;
Mat descriptors1, descriptors2;
sift_extractor.compute( gray_image1, keypoints1, descriptors1 );
sift_extractor.compute( gray_image2, keypoints2, descriptors2 );
...
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features

Slide 28

28

### SIFT – OpenCV code

```
...
// Match descriptors.
BFMatcher sift_matcher(NORM_L2);
vector< DMatch > matches;
matcher.match( descriptors1, descriptors2, matches );
// Display SIFT matches
Mat display_image;
drawMatches( gray_image1, keypoints1, gray_image2,
            keypoints2, matches, display_image );
```

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features

Slide 29

29

### SIFT – 6. Recognition

- Match at least 3 features
- Cluster matches
  - Hough transform
    - Location (2D)
    - Scale
    - Orientation
  - Really a 6D problem
    - Use broad bins
      - 30° for orientation
      - Factor of 2 for scale
      - 0.25 times image dimension for location
  - Consider all bins with at least 3 entries
    - Determine affine transformation

Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

Features

Slide 30

30

