Video

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

1

Introduction – Objects of interest

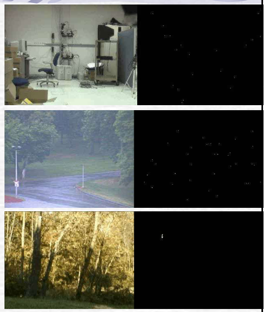
- Reliably detecting moving objects of interest in a scene.
  1. Motion detection
  2. Moving object detection & location
  3. Derivation of 3D object properties

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

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

3

Introduction – Static background

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

```
absdiff( frame, background, difference );
```

  - Background b(i,j) = first frame?

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.

Introduction – Colour

- But how did 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?

Introduction – Static Background

```
absdiff( current_frame, first_frame, difference );
cvtColor( difference, moving_points, CV_BGR2GRAY );
threshold( moving_points, moving_points, 30, 255, THRESH_BINARY );
Mat display_image = Mat::zeros( moving_points.size(), CV_8UC3 );
current_frame.copyTo( display_image, moving_points );
```

Background Models – Median

- Median: Middle value (from an ordered list)

$$h_n(i,j,p) = \sum_{k=(n-m+1)..n} \begin{cases} 1 & \dots \text{ if } (f_k(i,j) = p) \\ 0 & \dots \text{ otherwise} \end{cases}$$

- No. of frames (m)
- Histogram quantisation?
- Computational expense
  - Adding, storing and removing frames
  - Change in median can be tracked inexpensively from frame to frame
  - Can be approximated using aging

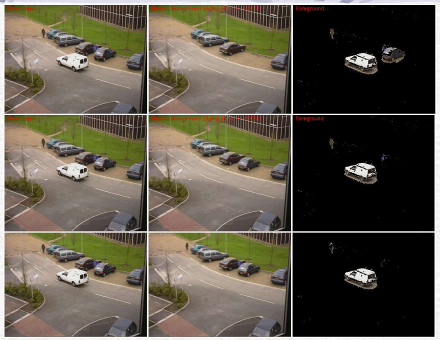
$$h_n(i,j,p) = \sum_{k=1..n} \begin{cases} w_k & \dots \text{ if } (f_k(i,j) = p) \\ 0 & \dots \text{ otherwise} \end{cases}$$

where $w_1 = 1$ and $w_k = w_{k-1} * 1.001$

- Can also use selective update
- Could use the **Mode** instead… (Most common value)

$$b_n(i,j) = p \quad \text{where } h_n(i,j,p) \geq h_n(i,j,q) \text{ for all } q \neq p$$

Background Models – Median Results



Learning Rate = 1.001

Learning Rate = 1.005

Learning Rate = 1.02

Background Models – Median Algorithm

First frame:

$total = 1$

for all pixels (i, j)

$median = f_1(i, j)$

$less\_than\_median(i, j) = 0$

Current frame (n):

$total = total + w_n$

for all pixels (i, j)

if $(median(i, j) > f_n(i, j))$

$less\_than\_median(i, j) = less\_than\_median(i, j) + w_n$

while $(less\_than\_median(i, j) + h_n(i, j, median(i, j)) < {}^{total}/_{2})$

$less\_than\_median(i, j) = less\_than\_median(i, j) + h_n(i, j, median(i, j))$

$median(i, j) = median(i, j) + 1$

while $(less\_than\_median(i, j) > {}^{total}/_{2})$

$median(i, j) = median(i, j) - 1$

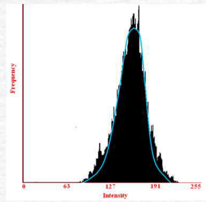
$less\_than\_median(i, j) = less\_than\_median(i, j) - h_n(i, j, median(i, j))$

Background Models – Gaussian Mixture Model

- How to deal with multi-modal background pixels
  - e.g. from trees, water
  - *Stauffer & Grimson, 2000*
  - *Algorithm presented is based on that in Sonka (3rd edition) pp.777-780*
- Model multiple values (3-5) at each point.
- Unsupervised learning…
- Most popular method for background modelling

Background Models – GMM Model details

- Model each pixel $f_n(i,j)$ using k Gaussian distributions each with
  - $\pi_n(i,j,m)$
  - $\mu_n(i,j,m)$
  - $\sigma^2_n(i,j,m)$
    - Must assume different components independent and of equal variance
- Set a learning constant α
  - in the range 0.01 … 0.1



Background Models – GMM Model Update

- For a new sample $f_n(i,j)$
  - Select **the best close Gaussian distribution**
    - **close** = within $2.5\ \sigma_n(i,j,m)$ of $\mu_n(i,j,m)$
  - If there is a best close Gaussian l

$$\pi_{n+1}(i,j,m) = \alpha * O_n(i,j,m) + (1-\alpha) * \pi_n(i,j,m)$$

where $O_n(i,j,m) = 1$ for **the close Gaussian distribution** and 0 otherwise

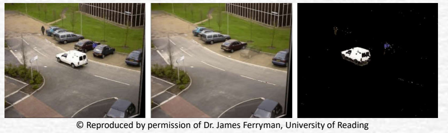
$$\mu_{n+1}(i,j,m) = \mu_n(i,j,m) + O_n(i,j,m) * (\alpha / \pi_{n+1}(i,j,m)) * (f_n(i,j) - \mu_n(i,j,m))$$

$$\sigma^2_{n+1}(i,j,m) = \sigma^2_{n+1}(i,j,m) + O_n(i,j,m) * (\alpha / \pi_{n+1}(i,j,m)) * ((f_n(i,j) - \mu_n(i,j,m))^2 - \sigma^2_n(i,j,m))$$

  - If there is no close Gaussian (replace one…)
    - $x = argmin_m(\pi_n(i,j,m))$
    - $\mu_{n+1}(i,j,x) = f_n(i,j)$
    - $\sigma^2_{n+1}(i,j,x) = 2.max_m\ \sigma^2_n(i,j,m)$
    - $\pi_{n+1}(i,j,x) = ½.min_m\ \pi_n(i,j,m)$

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
    - $B = argmin_b((\Sigma_{b=1..m}\ \pi_{n+1}(i,j,m)) > T)$
- 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.



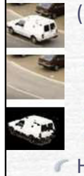
© Reproduced by permission of Dr. James Ferryman, University of Reading

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.

Background Models – Shadow Detection

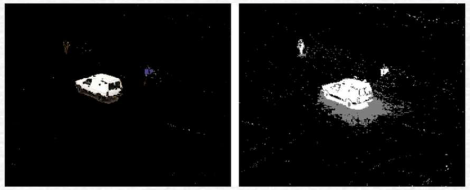
$$SP_k(i,j) = \begin{cases} 1 \ldots \text{ if } \left(\alpha < \frac{f_k^V(i,j)}{B_k^V(i,j)} < \beta\right) and \left(\left(f_k^S(i,j) - B_k^S(i,j)\right) < \tau_S\right) and \left(\left|f_k^H(i,j) - B_k^H(i,j)\right| < \tau_H\right) \\ 0 \ldots \text{ otherwise} \end{cases}$$



- 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)

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

Background Models – Shadow Detection



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

17

Background Models – Shadow Detection – Code

```
Ptr<BackgroundSubtractorMOG2> gmm =
                            createBackgroundSubtractorMOG2();
gmm->apply(current_frame, foreground_mask);
gmm( current_frame, foreground_mask );
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 );
```


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

19

Tracking – Topics

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

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.

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.

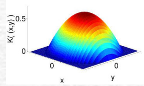
Tracking – Mean Shift

- Model the "object"

$$\hat{q}_u \triangleq C \sum_{i=1}^{n} k\left(\left\|\frac{\mathbf{x}_i}{h_q}\right\|^2\right) \delta\left[b\left(\mathbf{x}_i\right) - u\right]$$

  - 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

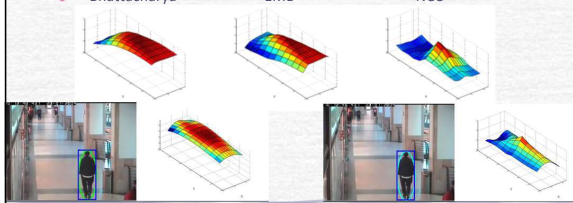


Tracking – Mean Shift

- Model candidates regions…

$$\hat{p}_u(\mathbf{y}) \triangleq C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h_p}\right\|^2\right) \delta\left[b\left(\mathbf{x}_i\right) - u\right]$$

- Matching to find the best new location…
  - Compare distributions directly.
  - Move to the mode in matching space.
  - Bhattacharya EMD NCC



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

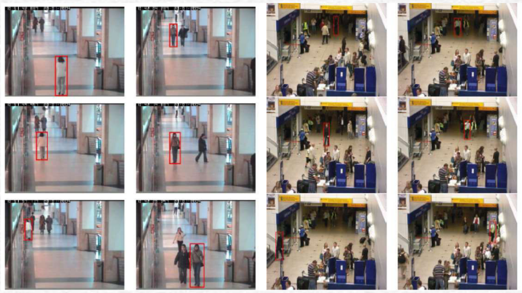
$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\mathbf{y_0})}}\,\delta\left[b\left(\mathbf{x}_i\right) - u\right] \qquad \mathbf{y_1} = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i w_i}{\sum_{i=1}^{n_h} w_i}$$

    - 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

25

Tracking – Mean Shift – Test dataset

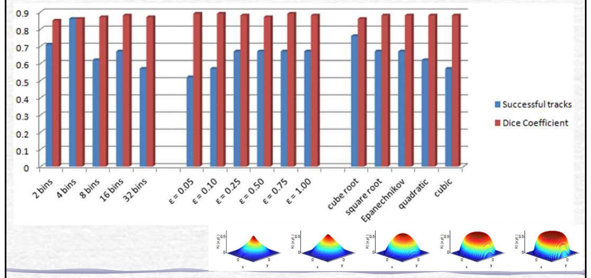
- CAVIAR (14) and PETS (7) tracking scenarios



26

Tracking – Mean Shift – Changing parameters

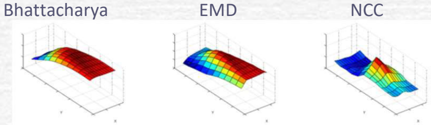
- Number of bins per channel
- Convergence parameter (ε)
- Kernel type



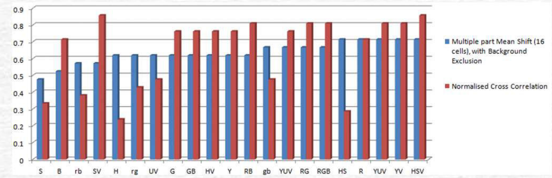
27

Tracking – Mean Shift – The best technique?

- Bhattacharya EMD NCC



- Colour Space



28

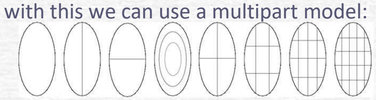
Tracking – Mean Shift – Background Exclusion

- 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.

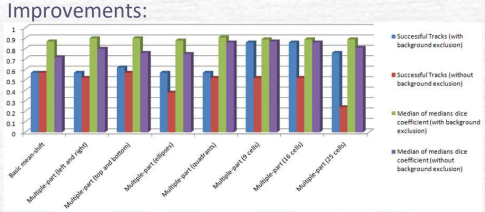
- Analogous to background subtraction
  - No image differencing.

Tracking – Mean Shift – Multipart models

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



- Improvements:



Tracking – Mean Shift – In OpenCV

- Back-projects 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.

Tracking – Mean Shift – In OpenCV

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

Tracking – Dense Optical Flow

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

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

Final Year Project by Gavin Corkery

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

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

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…

$$f_{t+\Delta t}(i+\Delta i, j+\Delta j) = f_t(i,j) + \frac{\partial f}{\partial i}\Delta i + \frac{\partial f}{\partial j}\Delta j + \frac{\partial f}{\partial t}\Delta t$$

- Hence (given our previous equation

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

- So

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

- And reorganising

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

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 );
```


36

Tracking – Dense Optical Flow – Problems

- Optical flow is the apparent motion of points within a scene.
  - Based on brightness patterns
  - Needs texture

Q. What happens if the brightness changes?
- i.e. brightness constancy does not hold.

A. Perhaps look at optical flow in gradient space.

Q. What if a point moves differently to all its neighbours?

A. Use Region based optical flow

Tracking – Dense Optical Flow – Problems

Q. What happens to a rotating sphere? What about a "barber pole"?

A. We get the wrong motion. i.e. it fails.

Q. What happens if the motion is too large?

A. Use Iterative Refinement (Lucas-Kanade)

Tracking – Feature-based Optical Flow

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