Applied Computer Vision

David Vernon
Carnegie Mellon University Africa

vernon@cmu.edu
www.vernon.eu

Lecture 21

Video Image Processing

Moving Object Detection

Motion detection issues, difference images,
background models: static background, running average, selective update, median, running Gaussian average, Gaussian mixture model

Moving Object Detection

Motion detection

Relative motion

- Facilitates segmentation of (moving) foreground from (static) background
- Possibly also segmentation of one moving object from another

Applications

- Motion detection (e.g. security alert)
- Detection and localization of moving object
  - Detection
  - Tracking
  - Recognition
  - Prediction of likely future positions (i.e. expected trajectory)
- Determine 3D structure (so-called structure from motion … camera or object motion)

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Moving Object Detection

Constraints defining objects of interest

Minimum size (in pixels)

Maximum velocity (pedestrians vs. cars)

- constrains change in position between frames

Maximum acceleration

- constrains change in velocity between frames

Common motion

- object move in a coordinated manner

(Near) constant appearance

- constrains change in colour or shape

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Moving Object Detection

Common problems

Illumination & appearance changes

- Gradual (e.g. time of day)
- Sudden (e.g. clouds, lights)
- Shadows
- Weather (e.g. rain, snow)

Background changes (and may need to be updated)

- Objects becoming part of the background (e.g. parked car)
- Objects leaving the background (e.g. parked car)
- Background objects oscillating slightly (e.g. trees or bushes)

Camera configuration

- Mobile or static?
- Pan, tilt, zoom?
- Time interval between frames: constrain allowable/detectable speed of objects

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Moving Object Detection

Difference Images

Image subtraction

background: first frame

Binary image output

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

Grey-level output

$$d(i,j) = |f_k(i,j) - b(i,j)|$$

Options for colour images

- Process each channel separately
- Just process hue

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Moving Object Detection

Difference Images

OpenCV

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

Issues

- Threshold selection
- Sensitivity to threshold
- False positives
  - detected pixels that are not moving
- False negatives
  - undetected pixels that are moving
- May need to perform some morphological processing

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Moving Object Detection

Difference Images

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Moving Object Detection

Difference Images

Moving Object Detection

Need to acquire and maintain the background model/image

"new" background

moving objects

Background

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

"missing" background

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Static background image

Acquire background image by taking a static frame of the scene

Simplest approach

Can’t deal with objects being added or removed from the static scene

Cannot deal with illumination changes

Cannot deal with even minor changes in the camera pose

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Frame difference

background: previous frame

Image subtraction

$$d(i,j) = |\, f_n(i,j) - f_{n\text{-}1}(i,j)\, | > \text{Threshold}$$

- Depends on the speed of the objects and the frame rate
- Sensitive to the Threshold

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

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Running Average

- Incorporate changes to the background
- Use the average of the last k frames
- Approximation to avoid storing k frames:

Learning rate

$$b_{n+1}(i,j) = \propto . f_n(i,j) + (1-\propto) . b_n(i,j)$$

- OpenCV

```
accumulateWeighted(current_frame_grey,
                   running_average_background, 0.01);
```

- For colour images, apply to each channel separately

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Running Average

- Will adapt to changing lighting conditions
- Will also incorporate in moving objects

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

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Selective Update

- To overcome the problem of incorporating foreground object into the background model, only update pixels which are determined to be background

$$b_{n+1}(i,j) = \begin{cases} \propto . f_n(i,j) + (1-\propto) . b_n(i,j) & \dots \text{ if } f_n(i,j) \text{ is background} \\ b_n(i,j) & \dots \text{ if } f_n(i,j) \text{ is foreground} \end{cases}$$

- OpenCV

```
accumulateWeighted(current_frame_grey,
                   running_average_background, 0.01.
                   foreground_mask);
```

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Selective Update

- objects that enter the scene and stop (e.g. a car parking) will not be incorporated into the background model

- Moving objects in the first frame are not removed

Background Models

Selective Update (with running average for foreground)

- Alternative: a different (lower) learning rate β for the foreground pixels

$$b_{n+1}(i,j) = \begin{cases} \propto . f_n(i,j) + (1-\propto). b_n(i,j) & \text{... if } f_n(i,j) \text{ is background} \\ \propto/_3 . f_n(i,j) + (1-\propto/_3). b_n(i,j) & \text{... if } f_n(i,j) \text{ is foreground} \end{cases}$$

Background Models

Running Average

Selective Update

Selective Update
(with running average)

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

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Median (middle element of an ordered list or a histogram)

Sum over last m frames

Histogram bin

- Compute the median background image

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

- Decide on the number of frames m
- Decide on the histogram quantisation, i.e. number of bins
- Computationally expensive
  - Adding, storing and removing frames
  - Change in median can be tracked inexpensively from frame to frame
  - Can be approximated using aging

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

where $w_1 = 1$ and $w_k = w_{k-1} * 1.001$ or some other value, e.g. 1.005

  - Can also use selective update

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Median

- Not supported in openCV but relatively easily implemented
- Could use the **Mode** instead… (most common value)

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

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Median

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

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

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

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

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

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

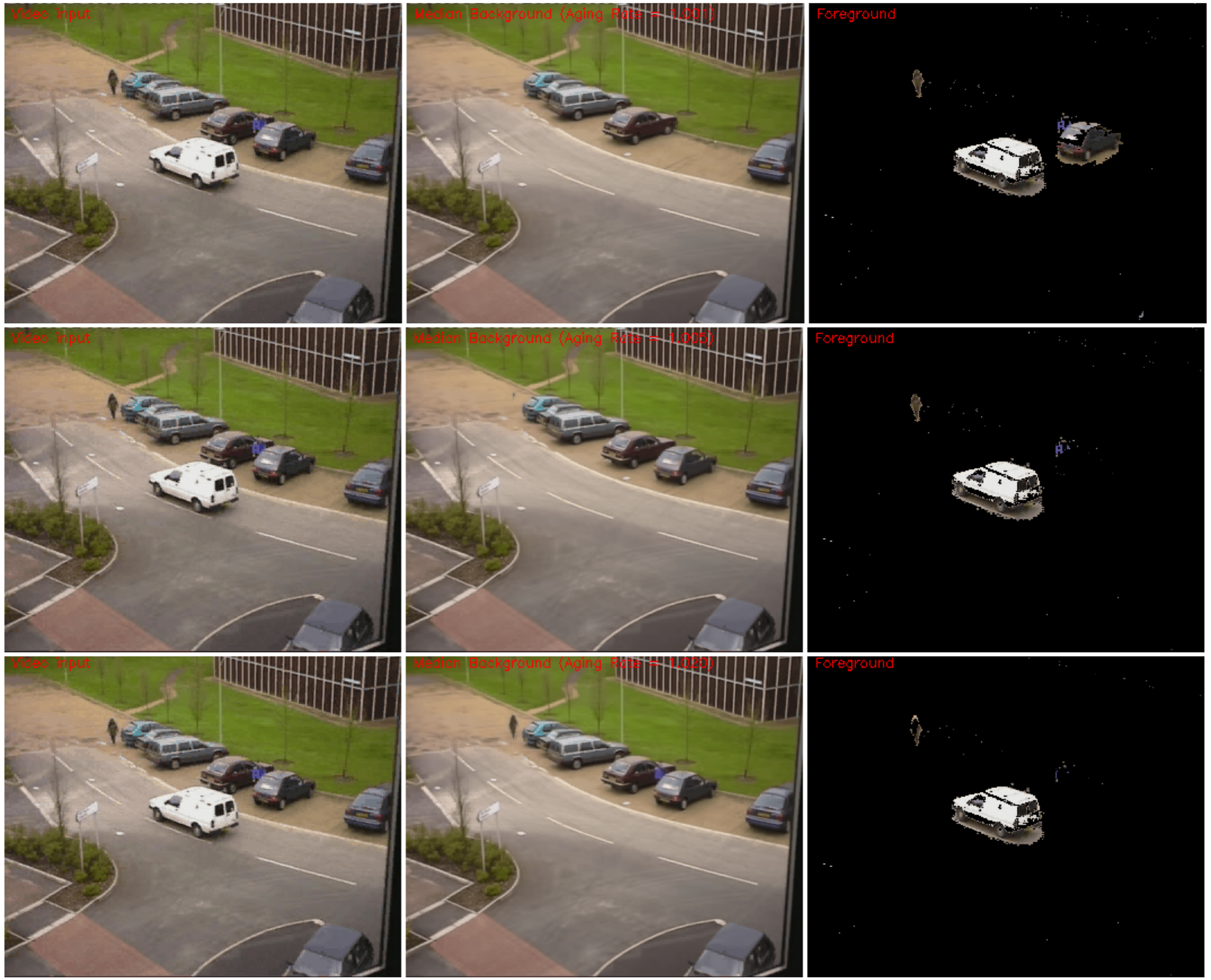
Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Learning Rate = 1.001

Learning Rate = 1.005

Learning Rate = 1.02



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

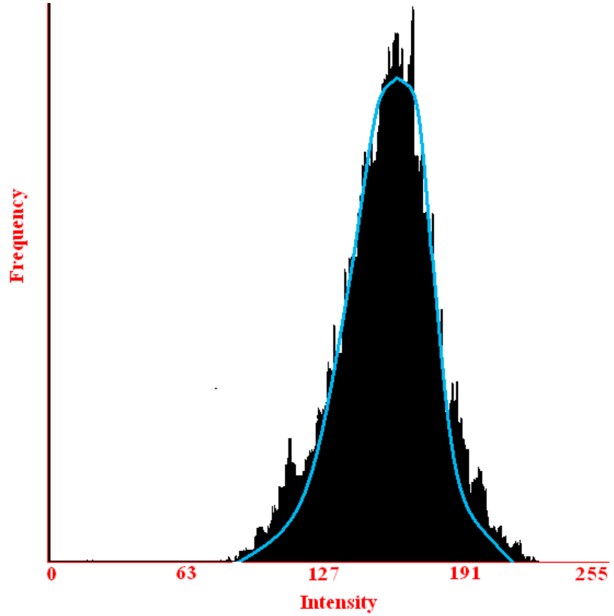
Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Background Models

Running Gaussian Average

- Pixel values change from frame to frame
  - sampling noise
  - variations in illumination
  - ...
- Model the noise as a Gaussian distribution:
  - PDF with mean and standard deviation μ, σ
- Define a point as foreground if it is more than some multiple k of σ from the mean μ

$$|f_n(i,j) - \mu_n(i,j)| > k\sigma_n(i,j)$$



Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

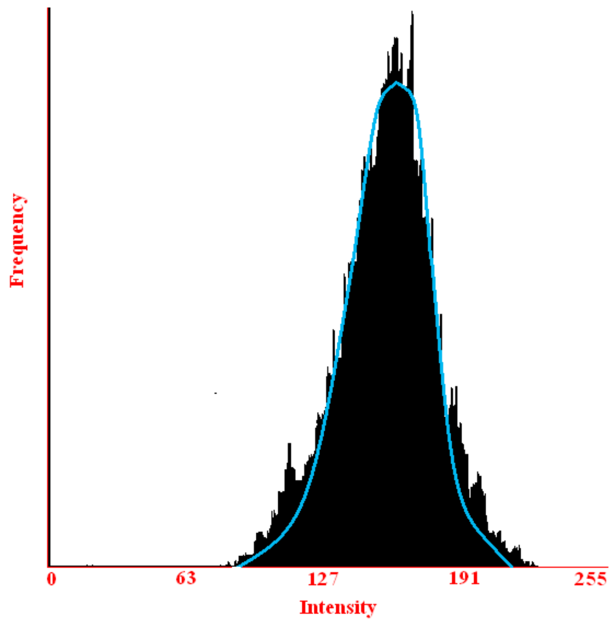
Background Models

Running Gaussian Average

- Update the Gaussian distribution in a manner similar to the running average

$$\mu_{n+1}(i,j) = \alpha f_n(i,j) + (1-\alpha)\mu_n(i,j)$$
$$\sigma^2_{n+1}(i,j) = \alpha(f_n(i,j) - \mu_n(i,j))^2 + (1-\alpha)\sigma^2_n(i,j)$$

- Can also use selective update if necessary so that the foreground objects don't pollute the Gaussian distribution



Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

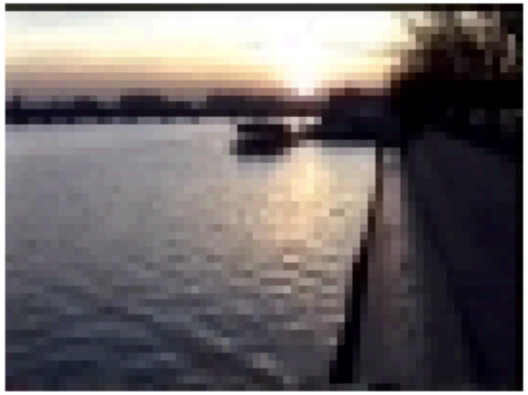
Background Models

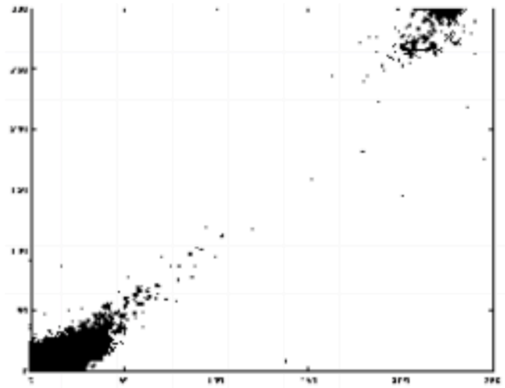
Gaussian Mixture Model

- None of the techniques so far are able to deal with background objects which move slightly
  - ripples on water
  - leaves and branches on trees,
- Stauffer & Grimson (1998) proposed to model each point of the background using a mixture of Gaussian distributions
  - Typically 3 - 5 distributions per pixel
  - For example
    - If a pixel corresponded to a leaf on a tree, and the leaf is moved by the wind, periodically revealing the sky, then two distributions would model the leaf and the sky respectively
    - If a pixel corresponded to a rippling water, then two distributions would model the water with and without a reflection

Background Models

Gaussian Mixture Model





Bi-modal distribution of a single pixel's values resulting from specularities on the surface of water

Credit: Stauffer and Grimson, 1998

Background Models

Gaussian Mixture Model

- Fit multiple Gaussian distributions to the historical pixel data for each point
  - Model each pixel $f_n(i, j)$ using K Gaussian distributions $N_K = N(\mu_K, \sigma_K)$
- At any frame n, each Gaussian distribution k $(1 \leq k \leq K)$ has a weight $\omega_{k,n}$ depending on how frequently the distribution been matched in past frames

Background Models

Gaussian Mixture Model

When a new frame is acquired, each pixel $f_n(i, j)$ is checked against the existing K Gaussian distributions currently modelling that pixel

- If the pixel value is $< 2.5\ \sigma$ from mean μ, then that distribution is updated
- Otherwise, a new distribution is created:
  - The least probable distribution is replaced
  - A new distribution is initialized with
    - mean equal to the pixel value
    - high variance
    - low weight

Background Models

Gaussian Mixture Model

The prior weights of the K distributions at frame n are adjusted as follows

$$\omega_{k,n} = (1 - \alpha)\, \omega_{k,n-1} + \alpha\, M_{k,n}$$

where $M_{k,n}$ $= 1$ for distribution that matched
$= 0$ for all other distributions

α is the learning rate

After this, the weights are re-normalized

The mean μ and standard deviation σ of the unmatched distributions remain the same

Background Models

Gaussian Mixture Model

The mean μ and standard deviation σ of the distribution k that matches the new observation are updated:

$$\mu_n = (1- \rho\,)\ \mu_{n\text{-}1} + \rho f_n(i, j)$$

$$\sigma^2{}_n = (1- \rho\,)\ \sigma^2{}_{n\text{-}1} + \rho\,(f_n(i, j) - \mu_n)^2$$

where

$$\rho = \alpha\ N_k\,(f_n(i, j))$$

The value of the Gaussian distribution for that pixel value
(i.e. the probability of that value occurring)

Background Models

Gaussian Mixture Model

Identify the background distributions (and label the pixel accordingly)

- Define T, a proportion of the frames in which background pixels should be visible (e.g. 70%)

- Order the distributions by value of ω_k / σ

- Choose the first B distributions as the background model where

$$B = argmin_b \left(\sum_{k=1}^{b} \omega_k > T \right)$$

where T is a measure of the minimum portion of the data that should be accounted for by the background

e.g. 70% allows for a multi-modal model of the background

Background Models

Gaussian Mixture Model

Background Models

Gaussian Mixture Model

openCV supports several variants of the GMM background model

GMM

Andrew B Godbehere, Akihiro Matsukawa, and Ken Goldberg. Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In American Control Conference (ACC), 2012, pages 4305–4312. IEEE, 2012.

MOG

Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In Video-Based Surveillance Systems, pages 135–144. Springer, 2002.

MOG2

Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In Video-Based Surveillance Systems, pages 135–144. Springer, 2002.

Demos

The following code is taken from the backgroundModelStatic project in the lectures directory of the ACV repository

See:

```
backgroundModelStatic.h
backgroundModelStaticImplementation.cpp
backgroundModelStaticApplication.cpp
```

```
/*
  Example use of openCV to extract the background and foreground images of a dynamic scene: static model
  (image difference between the first & current frames)
  ----------------------------------------------------------------------------------------------------------------------------
  Implementation file

  David Vernon
  27 October 2017
*/

#include "backgroundModelStatic.h"

/*
 * function backgroundModelStatic
 * Trackbar callback - threshold user input
*/

void backgroundModelStatic(int, void*) {

   extern Mat first_frame;
   extern Mat current_frame;
   extern char* input_window_name;
   extern char* background_window_name;
   extern char* foreground_window_name;
   extern int thresholdValue;

   Mat difference_frame;
   Mat binary_difference;
   Mat thresholded_image;

   if (thresholdValue < 1)  // the trackbar has a lower value of 0 which is invalid
      thresholdValue = 1;
```

```
    /* -------------------------------------------------------------------------------------
     * Adapted from code provided as part of "A Practical Introduction to Computer Vision with OpenCV"
     * by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014.  All rights reserved.
     */

    absdiff(current_frame,first_frame,difference_frame);

    cvtColor(difference_frame, thresholded_image, CV_BGR2GRAY);

    threshold(thresholded_image,thresholded_image,thresholdValue,255,THRESH_BINARY);

    binary_difference.setTo(Scalar(0,0,0));

    current_frame.copyTo(binary_difference, thresholded_image);

    /* ------------------------------------------------------------------------------------- */

    imshow(background_window_name, first_frame);
    imshow(foreground_window_name, binary_difference);
}
```

Demos

The following code is taken from the backgroundModelRunningAverage project in the lectures directory of the ACV repository

See:

```
backgroundModelRunningAverage.h
backgroundModelRunningAverageImplementation.cpp
backgroundModelRunningAverageApplication.cpp
```

```
/*
  Example use of openCV to extract the background and foreground images of a dynamic scene: Running Average Model
  ----------------------------------------------------------------------------------------------------------------
  Implementation file

  David Vernon
  27 October 2017
*/

#include "backgroundModelRunningAverage.h"

/*
 * function backgroundModelRunningAverage
 * Trackbar callback - threshold user input
*/

void backgroundModelRunningAverage(int, void*) {

   extern Mat current_frame;
   extern Mat running_average_background;
   extern char* input_window_name;
   extern char* background_window_name;
   extern char* foreground_window_name;
   extern int thresholdValue;

   Mat difference_frame;
   Mat binary_difference;
   Mat thresholded_image;

   if (thresholdValue < 1)  // the trackbar has a lower value of 0 which is invalid
      thresholdValue = 1;
```

```
    /* ---------------------------------------------------------------------------------------
     * Adapted from code provided as part of "A Practical Introduction to Computer Vision with OpenCV"
     * by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014.  All rights reserved.
     */

    // Running Average (three channel version)

    Mat temp_running_average_background;
    Mat selective_running_average_background;
    Mat running_average_foreground_mask;
    Mat running_average_difference;
    Mat running_average_foreground_image;
    double running_average_learning_rate = 0.01;

    vector<Mat> input_planes(3);
    split(current_frame,input_planes);

    vector<Mat> running_average_planes(3);
    split(running_average_background,running_average_planes);
    accumulateWeighted(input_planes[0], running_average_planes[0], running_average_learning_rate);
    accumulateWeighted(input_planes[1], running_average_planes[1], running_average_learning_rate);
    accumulateWeighted(input_planes[2], running_average_planes[2], running_average_learning_rate);
    merge(running_average_planes,running_average_background);
    running_average_background.convertTo(temp_running_average_background,CV_8U);
    absdiff(temp_running_average_background,current_frame,running_average_difference);
    split(running_average_difference,running_average_planes);

    // Determine foreground points as any point with a difference of more than threshold (DV) on any one channel:
    threshold(running_average_difference,running_average_foreground_mask,thresholdValue,255,THRESH_BINARY); // DV replaced literal 30
    split(running_average_foreground_mask,running_average_planes);
    bitwise_or( running_average_planes[0], running_average_planes[1], running_average_foreground_mask );
    bitwise_or( running_average_planes[2], running_average_foreground_mask, running_average_foreground_mask );
    running_average_foreground_image.setTo(Scalar(0,0,0));
    current_frame.copyTo(running_average_foreground_image, running_average_foreground_mask);

    /* --------------------------------------------------------------------------------------- */

    imshow(background_window_name, temp_running_average_background);
    imshow(foreground_window_name, running_average_foreground_image);
}
```

Demos

The following code is taken from the backgroundModelSelectiveUpdate project in the lectures directory of the ACV repository

See:

```
backgroundModelSelectiveUpdate.h
backgroundModelSelectiveUpdateImplementation.cpp
backgroundModelSelectiveUpdateApplication.cpp
```

```
/*
  Example use of openCV to extract the background and foreground images of a dynamic scene: Selective Update Model
  -----------------------------------------------------------------------------------------------------------------
  Implementation file

  David Vernon
  27 October 2017
*/

#include "backgroundModelSelectiveUpdate.h"

/*
 * function backgroundModelSelectiveUpdate
 * Trackbar callback - threshold user input
*/

void backgroundModelSelectiveUpdate(int, void*) {

   extern Mat selective_running_average_background;
   extern Mat current_frame;
   extern char* input_window_name;
   extern char* background_window_name;
   extern char* foreground_window_name;
   extern int thresholdValue;

   Mat difference_frame;
   Mat binary_difference;
   Mat thresholded_image;

   if (thresholdValue < 1)  // the trackbar has a lower value of 0 which is invalid
      thresholdValue = 1;
```

```
/* ---------------------------------------------------------------------------------------------
 * Adapted from code provided as part of "A Practical Introduction to Computer Vision with OpenCV"
 * by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014.  All rights reserved.
 */

// Selective Running Average

Mat temp_selective_running_average_background;
Mat selective_running_average_foreground_mask;
Mat selective_running_average_difference;
Mat selective_running_average_foreground_image;
double running_average_learning_rate = 0.01;

vector<Mat> input_planes(3);
split(current_frame,input_planes);

// Running Average with selective update
vector<Mat> selective_running_average_planes(3);

// Find Foreground mask
selective_running_average_background.convertTo(temp_selective_running_average_background,CV_8U);
absdiff(temp_selective_running_average_background,current_frame,selective_running_average_difference);
split(selective_running_average_difference,selective_running_average_planes);

// Determine foreground points as any point with an average difference of more than a given threshold over all channels:
Mat temp_sum = (selective_running_average_planes[0] + selective_running_average_planes[1] +
                selective_running_average_planes[2])/3;
threshold(temp_sum,selective_running_average_foreground_mask,thresholdValue,255,THRESH_BINARY_INV); // DV replaced literal 30
                                                                                                     // with thresholdValue
```

```
    // Update background
    split(selective_running_average_background,selective_running_average_planes);
    accumulateWeighted(input_planes[0], selective_running_average_planes[0], running_average_learning_rate,
                       selective_running_average_foreground_mask);
    accumulateWeighted(input_planes[1], selective_running_average_planes[1], running_average_learning_rate,
                       selective_running_average_foreground_mask);
    accumulateWeighted(input_planes[2], selective_running_average_planes[2], running_average_learning_rate,
                       selective_running_average_foreground_mask);

    invertImage(selective_running_average_foreground_mask,selective_running_average_foreground_mask);

    accumulateWeighted(input_planes[0], selective_running_average_planes[0], running_average_learning_rate/3.0,
                       selective_running_average_foreground_mask);
    accumulateWeighted(input_planes[1], selective_running_average_planes[1], running_average_learning_rate/3.0,
                       selective_running_average_foreground_mask);
    accumulateWeighted(input_planes[2], selective_running_average_planes[2], running_average_learning_rate/3.0,
                       selective_running_average_foreground_mask);

    merge(selective_running_average_planes,selective_running_average_background);

    selective_running_average_foreground_image.setTo(Scalar(0,0,0));
    current_frame.copyTo(selective_running_average_foreground_image, selective_running_average_foreground_mask);

    /* ------------------------------------------------------------------------------------------------ */

    imshow(background_window_name, temp_selective_running_average_background);
    imshow(foreground_window_name, selective_running_average_foreground_image);
}
```

Demos

The following code is taken from the backgroundModelMedian project in the lectures directory of the ACV repository

See:

```
backgroundModelMedian.h
backgroundModelMedianImplementation.cpp
backgroundModelMedianApplication.cpp
```

```
/*
  Example use of openCV to extract the background and foreground images of a dynamic scene: Median Model
  ---------------------------------------------------------------------------------------------------------
  Implementation file

  David Vernon
  27 October 2017
*/

#include "backgroundModelMedian.h"

/*
 * function backgroundModelMedian
 * Trackbar callback - threshold user input
*/

void backgroundModelMedian(int, void*) {

   extern Mat current_frame;
   extern char* input_window_name;
   extern char* background_window_name;
   extern char* foreground_window_name;
   extern int thresholdValue;

   /* Bug: there is no way to reinstantiate median_background or change its properties in the event that     */
   /* if the dimensions of the current_frame change such as will be the case when a new video file is opened */
   /* Consequently, we only run this example with one video file                                              */
   static MedianBackground median_background( current_frame, (float) 1.005, 1 );

   Mat median_background_image;
   Mat median_foreground_image;
   Mat median_difference;
```

```
    if (thresholdValue < 1)  // the trackbar has a lower value of 0 which is invalid
       thresholdValue = 1;

    /* ----------------------------------------------------------------------------------------------
     * Adapted from code provided as part of "A Practical Introduction to Computer Vision with OpenCV"
     * by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014.  All rights reserved.
     */

    median_background.UpdateBackground( current_frame );
    median_background_image = median_background.GetBackgroundImage();
    absdiff(median_background_image,current_frame,median_difference);
    cvtColor(median_difference, median_difference, CV_BGR2GRAY);
    threshold(median_difference,median_difference,thresholdValue,255,THRESH_BINARY); // DV: replaced literal 30 with thresholdValue
    median_foreground_image.setTo(Scalar(0,0,0));
    current_frame.copyTo(median_foreground_image, median_difference);

    /* ---------------------------------------------------------------------------------------------- */

    imshow(background_window_name, median_background_image);
    imshow(foreground_window_name, median_foreground_image);
}
```

Demos

The following code is taken from the backgroundModelGMM project in the lectures directory of the ACV repository

See:

```
backgroundGMMStatic.h
backgroundModelGMMImplementation.cpp
backgroundModelGMMApplication.cpp
```

```
/*
  Example use of openCV to extract the background and foreground images of a dynamic scene: Gaussian Mixture Model (GMM)
  ---------------------------------------------------------------------------------------------------------------------
  Implementation file

  David Vernon
  27 October 2017
*/

#include "backgroundModelGMM.h"

void backgroundModelGMM() {

   extern Mat current_frame;
   extern char* input_window_name;
   extern char* background_window_name;
   extern char* foreground_window_name;
   extern char* foreground_mask_window_name;

   Mat thresholded_image;
   static BackgroundSubtractorMOG2 gmm;

   /* ---------------------------------------------------------------------------------------------
    * Adapted from code provided as part of "A Practical Introduction to Computer Vision with OpenCV"
    * by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014.  All rights reserved.
    */

   Mat foreground_mask;
   Mat foreground_image = Mat::zeros(current_frame.size(), CV_8UC3);

   // Update the Gaussian Mixture Model
   gmm(current_frame, foreground_mask);
```

```
    threshold(foreground_mask,thresholded_image,150,255,THRESH_BINARY);
    Mat moving_incl_shadows, shadow_points;
    threshold(foreground_mask,moving_incl_shadows,50,255,THRESH_BINARY);
    absdiff( thresholded_image, moving_incl_shadows, shadow_points);
    foreground_image.setTo(Scalar(0,0,0));
    current_frame.copyTo(foreground_image, thresholded_image);

    // Create an average background image (just for information)
    Mat mean_background_image;
    gmm.getBackgroundImage(mean_background_image);

    /* ------------------------------------------------------------------------------------------------ */

    imshow(background_window_name, mean_background_image);
    imshow(foreground_window_name, foreground_image);
    imshow(foreground_mask_window_name, foreground_mask);
}
```

Reading

R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010.

Section 12.6.4 Whole body modeling and tracking