# CS/ECE 8690 Computer Vision
# Homework 4 –Part B [60 pts]

Out: Tuesday April 4, 2023
Due: Thursday April 13, 2023

### Moving Object Detection using Simple Background Subtraction

The goal of this assignment is to implement a very simple background subtraction algorithm to detect moving objects in a scene imaged using a stationary camera. In background subtraction methods, moving regions are detected through difference between the current frame and a reference background image.

$$| frame_i - Background_i | > Th$$

These approaches provide the most complete feature data but are often sensitive to dynamic scene changes due to lighting and extraneous events.

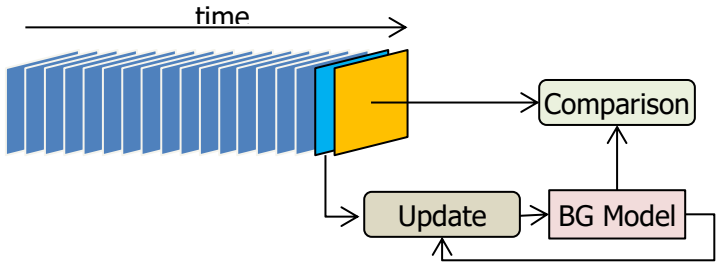
**Assignment:**

Implement a very simple background subtraction algorithm where previous values for each pixel is modeled using a single Gaussian distribution (μ,σ).

**Mean and Covariance (Single Gaussian)**

Update equations:

- $\mu(i,j,t+1) = \alpha\, frame(i,j,t) + (1-\alpha)\mu(i,j,t)$
- $\sigma^2(i,j,t+1) = \alpha(frame(i,j,t) - \mu(i,j,t))^2 + (1-\alpha)\sigma^2(i,j,t)$



## Program Structure:

Write a background subtraction class with the following three methods and call from a main program. You can make small changes in the input/output arguments according to your method.

```
class BGSubModel:
  """Background subtraction model.

  Please replace the ellipsis (...) with actual implementation code.
  """
  def __init__(self, first_frame, alpha, tm):
    """Initialize the background subtraction model.

    Args:
        first_frame (np.ndarray): The initial background image.
        alpha (float): Learning rate.
        tm (float): Matching threshdold.
    """
    self.mean = ...
    self.var = ...

    self.alpha = alpha
    self.tm = tm

  def classify(self, current_frame):
    """Classification method.
    Given the BG model and the current_frame, this function compares each pixel
    from the new frame to the BG model and identifies background and foreground
    regions. High difference is a sign of foreground.

    Args:
        current_frame (np.ndarray): The current frame we wish to classify.
    """
    ...

  def update(self, current_frame):
    """Update the BG model.
    Given the BG model and the new frame, updates the BG model. See the single
    Gaussian update equations above.

    Args:
```

```
        current_frame (np.ndarray): The current frame.
    """
    ...
```

## Reporting the Results:

1) Download the following test data and use it for the following questions. CAVIAR1 dataset from http://sbmi2015.na.icar.cnr.it/SBIdataset.html

2) Produce the results for the parameters given below (a-b). Show input frame, background model, and foreground mask for frames 5, 100, 400 and interpret the results (for four cases listed below).

    a. Test two different learning rates, alpha=0.01 and alpha=0.001

    b. Test two different matching thresholds Tm=2 and Tm=3. Note the actual thresholding is done with Tm x $\boldsymbol{\sigma}$.

Include source code, figures, and interpretations to your report. Submit report + programs to canvas.

## Important:

- Failure to follow the given programming structure will result in deduction of points.
- Failure to use the given test data will result in deduction of points.

### See the following source for further details on background subtraction:

- Slides Lec16_MotionAnalysis

Your main program should look somewhat like below.

```
# Parameters
ALPHA = 0.01
TM = 3

# Files & Folders
INPUT_PATH = './input'
OUTPUT_PATH = './output'

def main():
  if not os.path.isdir(OUTPUT_PATH):
    os.mkdir(OUTPUT_PATH)
  flist = [f for f in os.listdir(INPUT_PATH) if f.endswith('.jpg')]
  flist = sorted(flist)
  n = len(flist)

  # Read the first image and initialize the model
```

```
  im = cv2.imread(os.path.join(INPUT_PATH, flist[0]))
  bg_model = BGSubModel(im, ALPHA, TM)

  # Main loop
  for fr in range(n):
    # Read the image
    im = cv2.imread(os.path.join(INPUT_PATH, flist[fr]))

    # Classify the foreground using the model
    fg_mask = bg_model.classify(im)

    # Update the model with the new image
    bg_model.update(im)

    # Save the results
    fname = 'FGmask_' + flist[fr]
    fname_wpath = os.path.join(OUTPUT_PATH, fname)
    cv2.imwrite(fname_wpath, fg_mask)

    fname = 'BGmean_' + flist[fr]
    fname_wpath = os.path.join(OUTPUT_PATH, fname)
    cv2.imwrite(fname_wpath, bg_model.mean.astype('uint8'))


if __name__ == '__main__':
  main()
```