Niwot High School Independent Study: Security Camera

By: Aidan Coopman
Year: Junior
Date: Dec 16, 2018

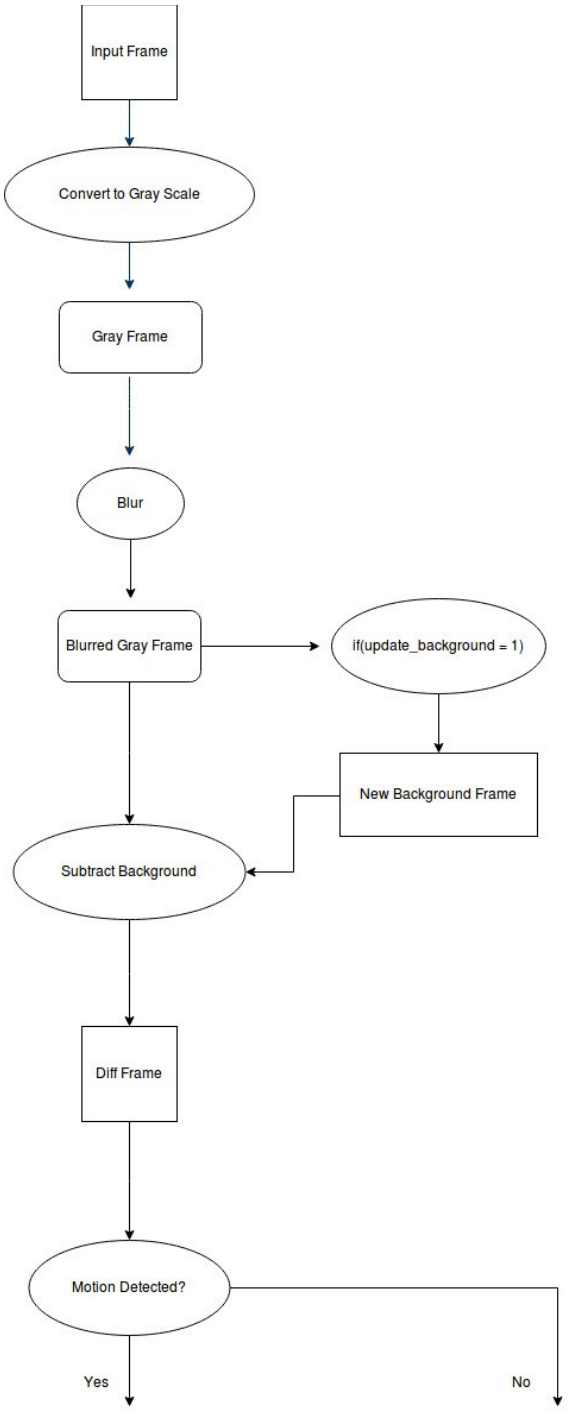
Semester One Objective:

The goal for semester one is to make a security camera that reduces false positives compared to other cameras such as Arlo. Arlo has false positives caused by the camera moving, and bugs landing on lens.

Semester Two Objective:

After an object is detected, one can use the detected image for classification. Classification is identifying the object in the video such as: dog, cat, person, car, etc. With this classification information we can farther reduce false positives such as not giving alerts if a squirrel walks by.

Semester One: Motion Detection Pipeline



1) Frame Capture

For frame capture I used the openCV library and used this tutorial:
https://www.learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/

```
VideoCapture cap(1); // open the default camera is 0, usb camera is 1
   if(!cap.isOpened())  // check if we succeeded
     return -1;

//main loop of the program
   while(1)
     {
            //use openCV's VideoCapture class to get an input frame from camera
            cap >> frame;

            //get the height and width parameters
            int height  = frame.rows;
            int width = frame.cols;
…}

   cap.release();
```

2) Converting to Gray Scale

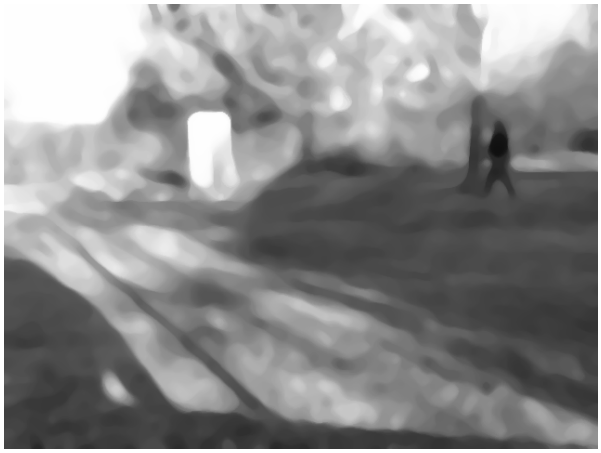
Image frames come in RGB format but for processing gray scale is used. Gray scale is just a “black and white” image which makes images a lot more easy to control and work with. A pixel is stored as a byte which has values between 0-255 with 0 being black and 255 being white. For this conversion the program uses the openCV function below:

```
//covert the input frame to a black and white frame
cvtColor(frame, gray_frame, CV_BGR2GRAY);
```

3) Image Blurring

Image blurring gets rid of edges in the image. Small movements such as leaves moving on a tree can cause a detection which is a false positive. By blurring the images we get rid of these false positives.

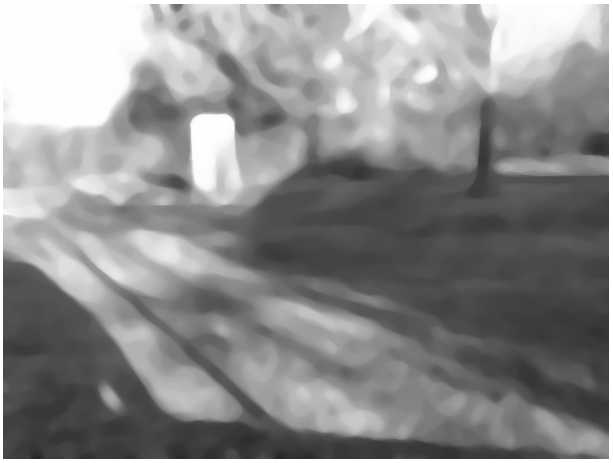
```
void image_blur(Mat & in, int N)
{
  for (int i  = 0; i < N; i++)
    medianBlur ( in, in, 9 );
}
```



4) Creating a Background Frame

The background frame is the current input frame and gets updated approximately every 30 frames, which is 1 second.

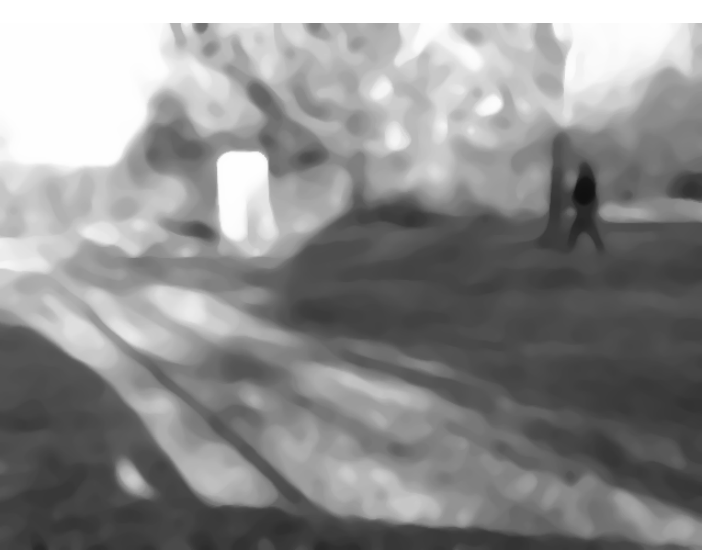
```
void update_background(Mat & input, Mat & background, motion_detect_params_t * params,
                                    int motion_flag, int frame_count)
{
  if(( (frame_count % params->update_frequency) == 0) && (!motion_flag))
    {
      background = input.clone();
      cout << "Background has changed\n";
    }
}
```

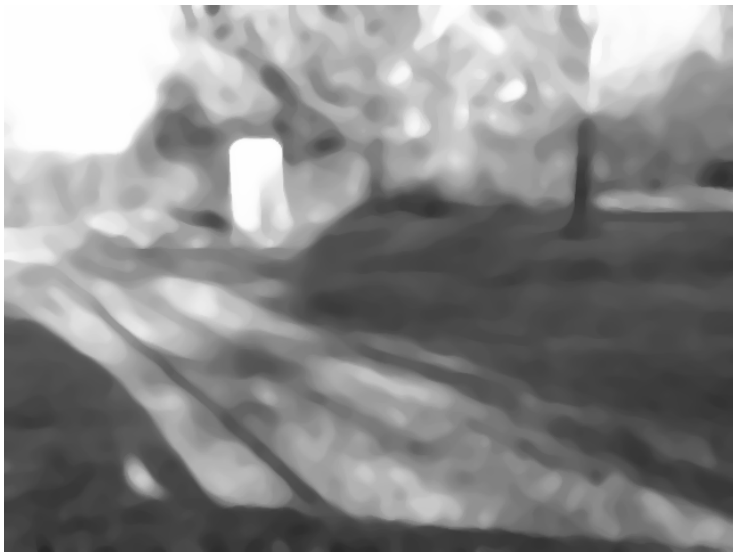


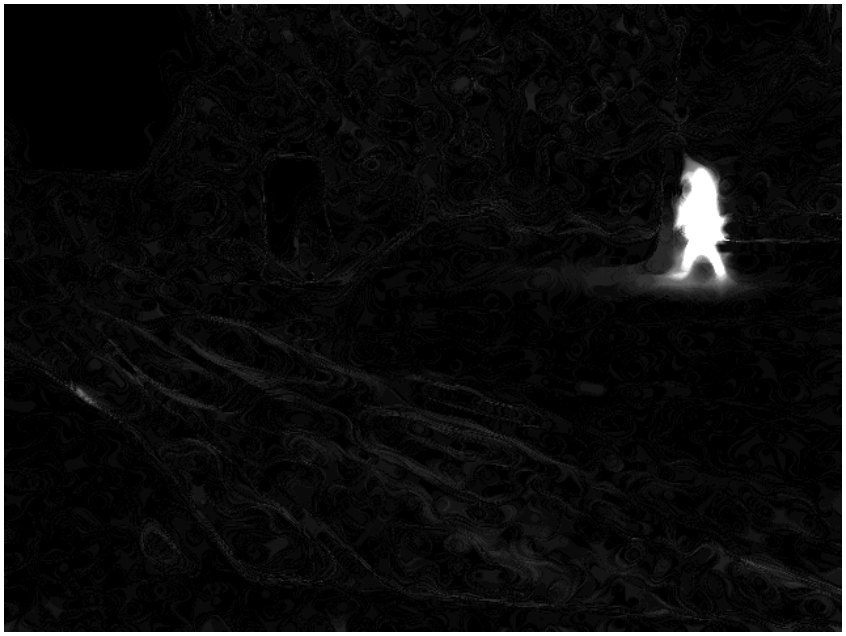
5) Subtracting the Background

After the background is set, a difference frame is made by subtracting the current frame by the background frame.

```
void subtract_background(Mat & diff_frame, Mat & input, Mat background_frame)
{
  diff_frame = abs(input - background_frame);
}
```







6) Detect Motion From Difference Frame

There are three features used to detect motion:

1. Number of pixels in the difference frame greater than a set threshold
2. The x coordinate standard deviation above the threshold
3. The y coordinate standard deviation above the threshold

Once a difference frame is computed, the number of pixels above a set threshold is counted. In the picture below the white pixels are counted. When the count reaches a certain threshold then motion is detected.

```
//if the pixels changed is greater than the set threshold, say there is motion
  if(pixels_change > param->threshold_pixel_change)
    {
      motion_flag = 1;
    }
  else
    {
      motion_flag = 0;
    }
```

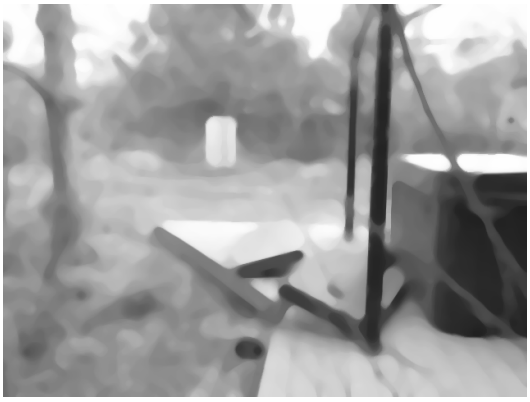
param->number_pixels_changed = 2694
center_x = 530 center_y = 169
std_x = 24.5564 std_y = 25.1769

This method of motion detection works fine but some false positives happen easily. A camera moving, or a bug landing on the lens would cause a false positive. To filter these out, standard deviation is used to determine the spread of the pixels that changed.

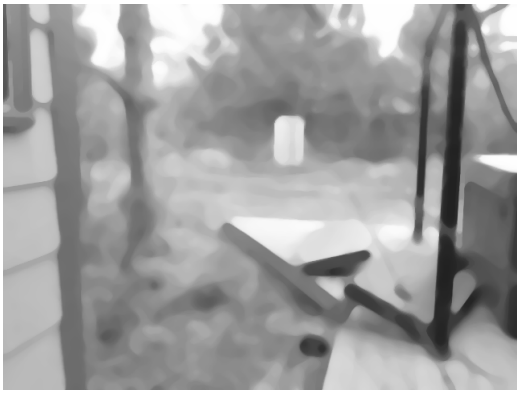
Using Standard Deviation to get rid of False Positives

Below is an example of when the whole camera moved causing a false positive because all the pixels changed. To filter out these false positives, standard deviation is used. If the standard deviation is above a threshold then motion detection is false.

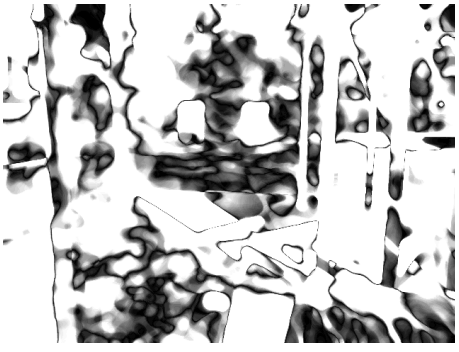
Grey Frame – Background Frame = diff Frame:



-

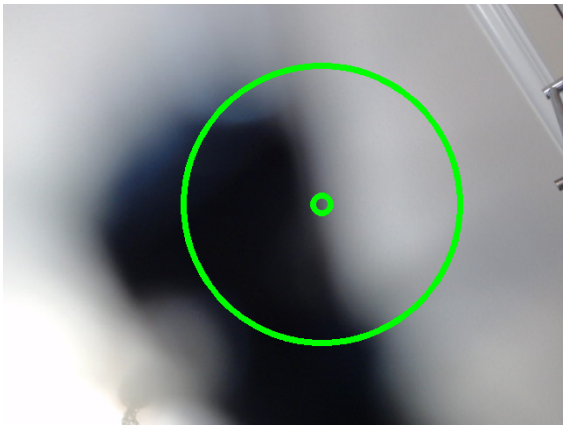


=



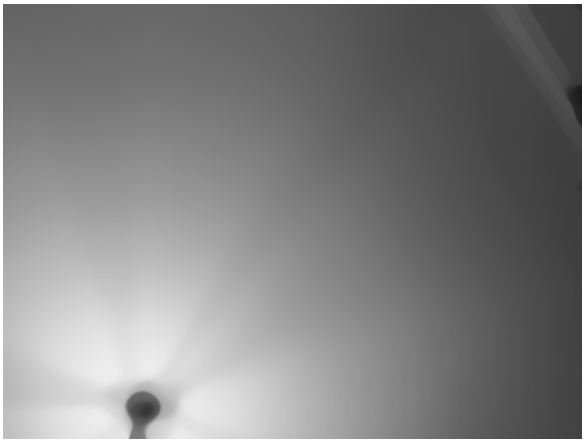
Motion detected
center_x = 401 center_y = 192
Pixel changes = 58719 threshold = 100
std_x = 162.855 std_y = 139.275

False Positive: Bug Land on Lens



Grey Frame – Background Frame = Difference Frame:

-



=

Motion detected
center_x = 364 center_y = 228
Pixel changes = 188176 threshold = 100
std_x = 178.82 std_y = 138.999

In the two above examples the standard deviation is over 140, while the example of the person walking in the backyard is roughly 25. We can use this information to get rid of false positives.

```
//check to make sure the std are within the threshold
if(param->std_x > 100.0f)
  return false;
if(param->std_y > 100.0f)
  return false;
```

Current Code Location:

https://github.com/acoopman/IndependentStudySecurityCamera

Future 2nd Semester Work Classification:

After an object has been detected, it needs to be classified. This is where the arlo camera falls apart. For example a bug that flies in front of the camera triggers a security alert.

References:

https://opencv.org/