Niwot High School Independent Study: Security Camera

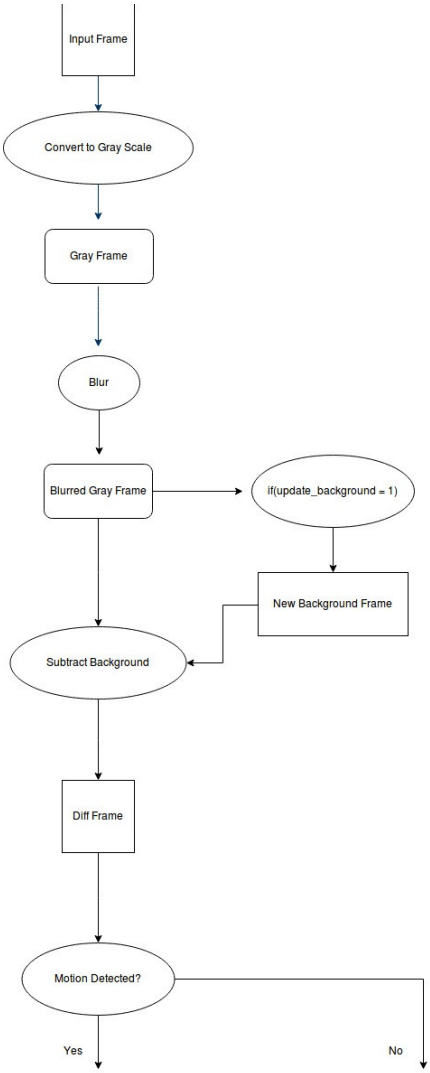
By: Aidan Coopman
Year: Junior
Date: Nov 13, 2018

Semester One Objective:
Make a security camera that has reduces false positives compared to other security 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, we 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 Status:

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

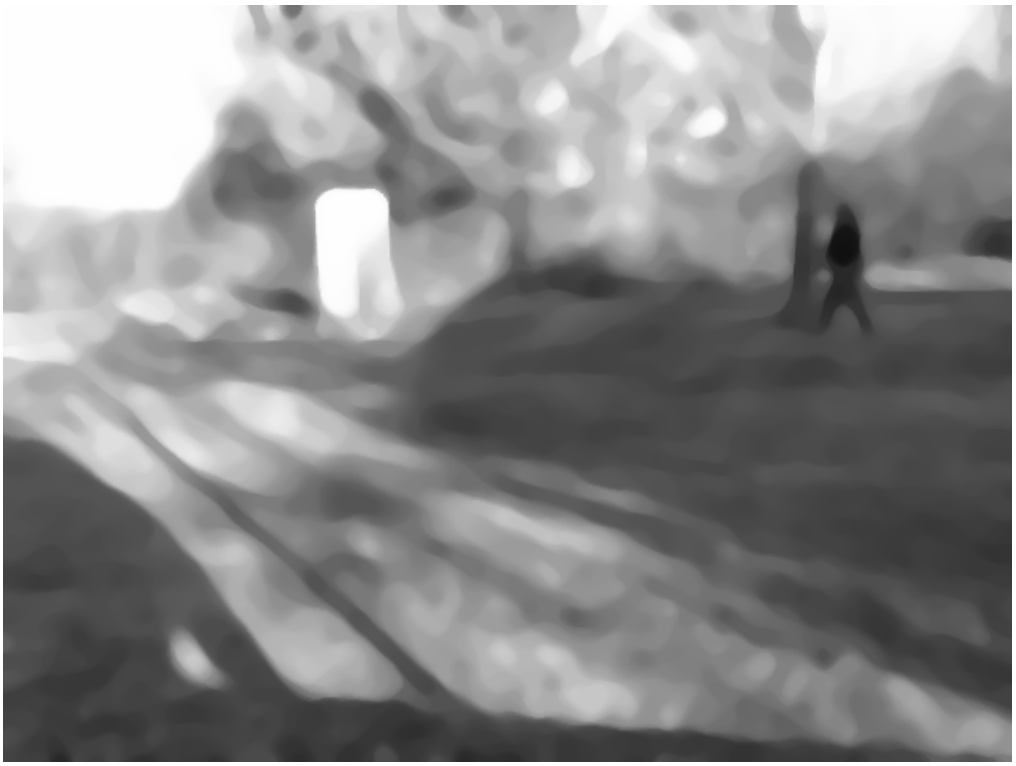
The frame comes in RGB but for processing we use grey scale. Grey scale is just black and white image, which makes images a lot more easy to control and work with. A pixel is stored as a byte which has values between 0-255 with 0 being black and 255 being white. For this conversion we use the openCV function from RGB 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. This is important because little things like leaves could move and could trigger the program to think that it is motion.

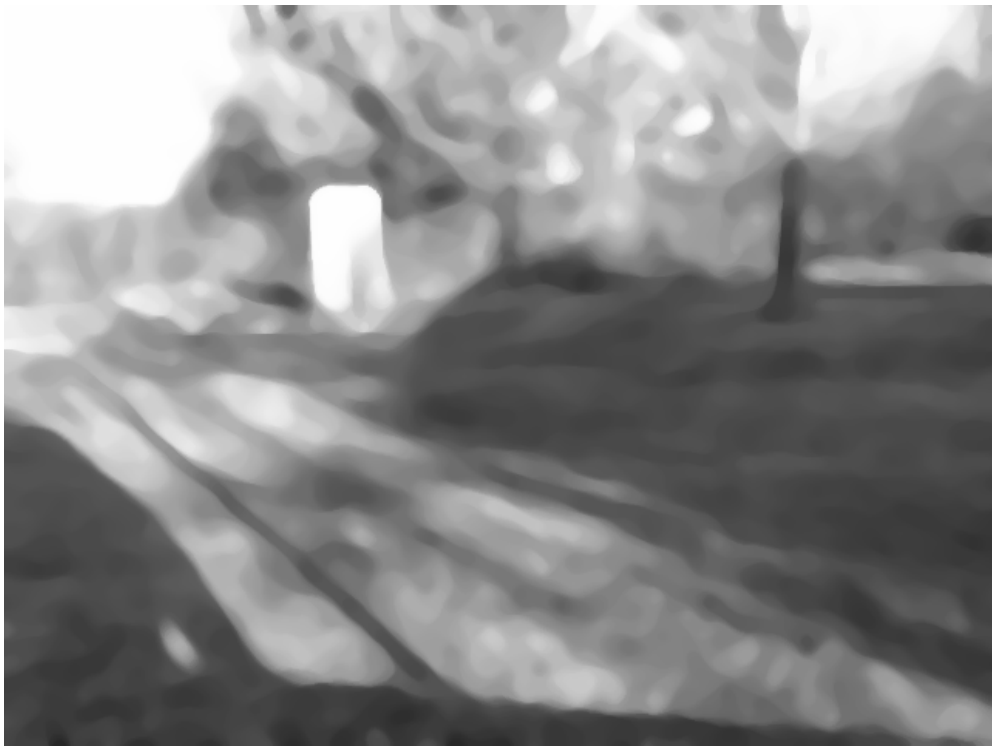
```
void image_blur(Mat & in, int N)
{
  for (int i  = 0; i < N; i++)
    medianBlur ( in, in, 9 );
}
```



4) Creating a Background Frame

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

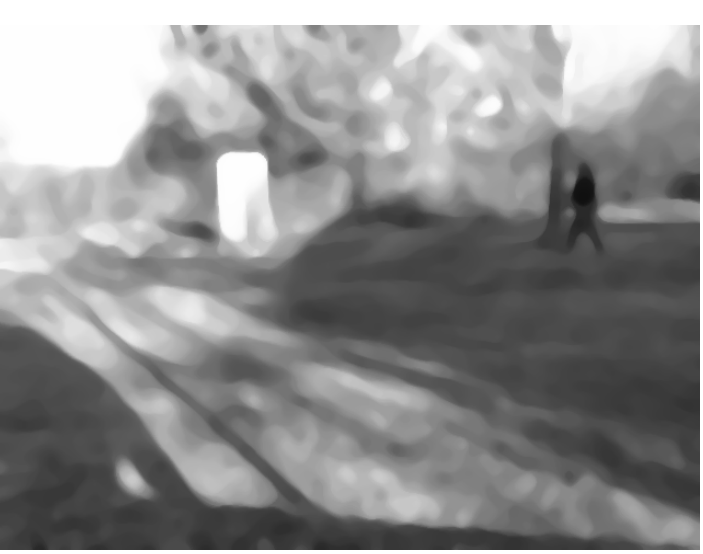
```
void update_background(Mat & input, Mat & background, motion_detect_params_t * params,
                                    int motion_flag, int frame_count)
{
  if(( (frame_count % params->update_frequency) == 0) && (!motion_flag))
    {
      background = input.clone();
      cout << "Background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

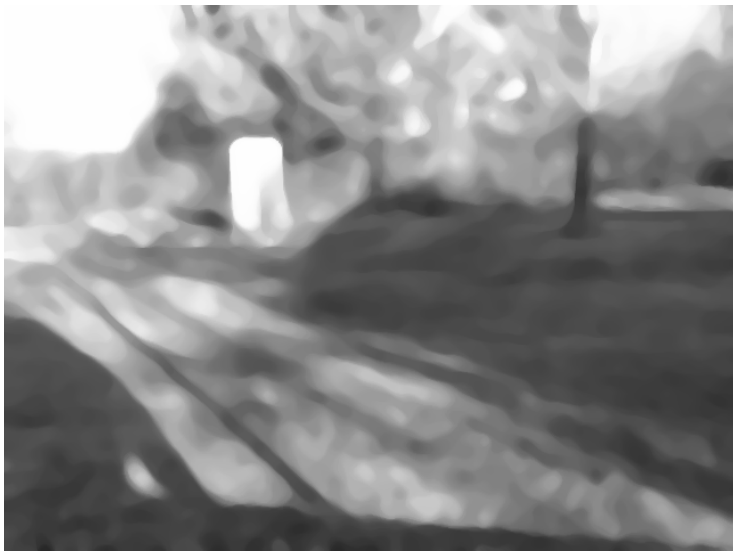


5) Subtracting the Background

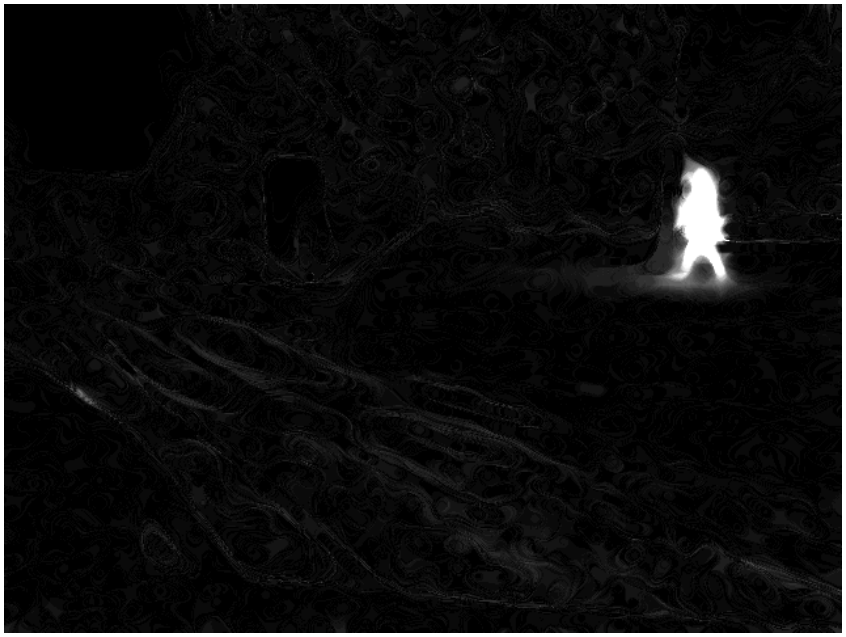
After the background is set, a difference frame is made by subtracting the current frame by the background frame.

```
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 center of all the pixels above the threshold
3. The x coordinate standard deviation above the threshold
4. The y coordinate standard deviation above the threshold

Motion detected

center_x = 530 center_y = 169
std_x = 24.5564 std_y = 25.1769
int radius = ((detect_params.std_x+detect_params.std_y)/2);

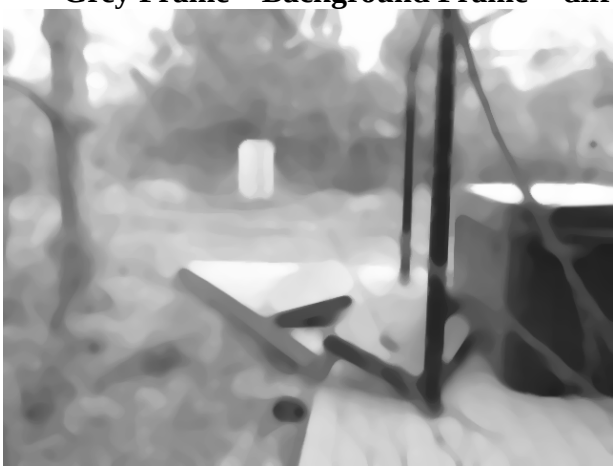
```
circle(frame, Point(detect_params.center_x, detect_params.center_y), 1,
                        Scalar(0,255,0), 5);

 circle(frame, Point(detect_params.center_x, detect_params.center_y), radius,
                        Scalar(0,0,255), 2);
```

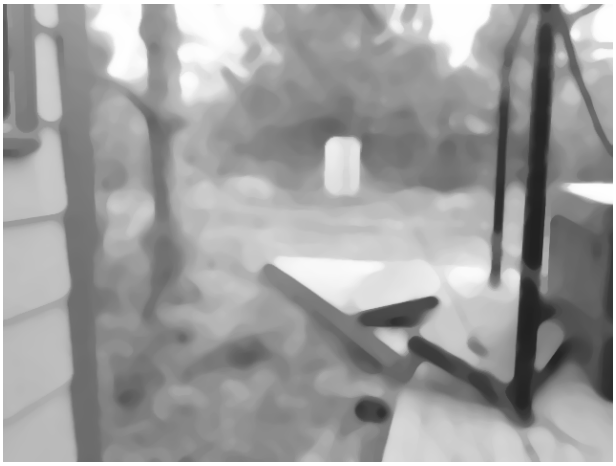
param->number_pixels_changed = 2694

Using Standard Deviation to get rid of False Positives

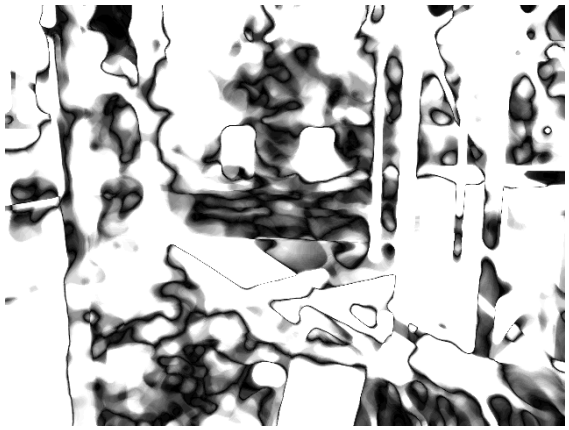
Grey Frame – Background Frame = diff Frame:



-

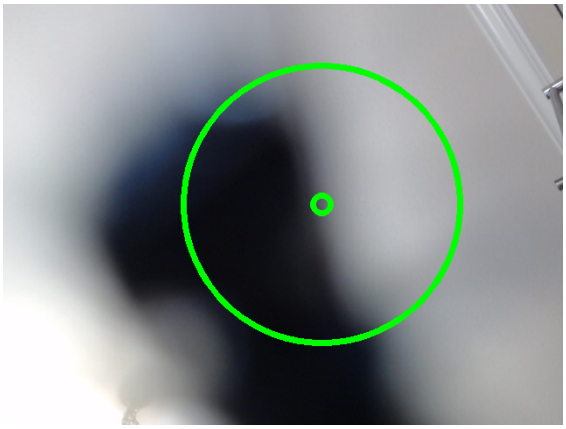


=

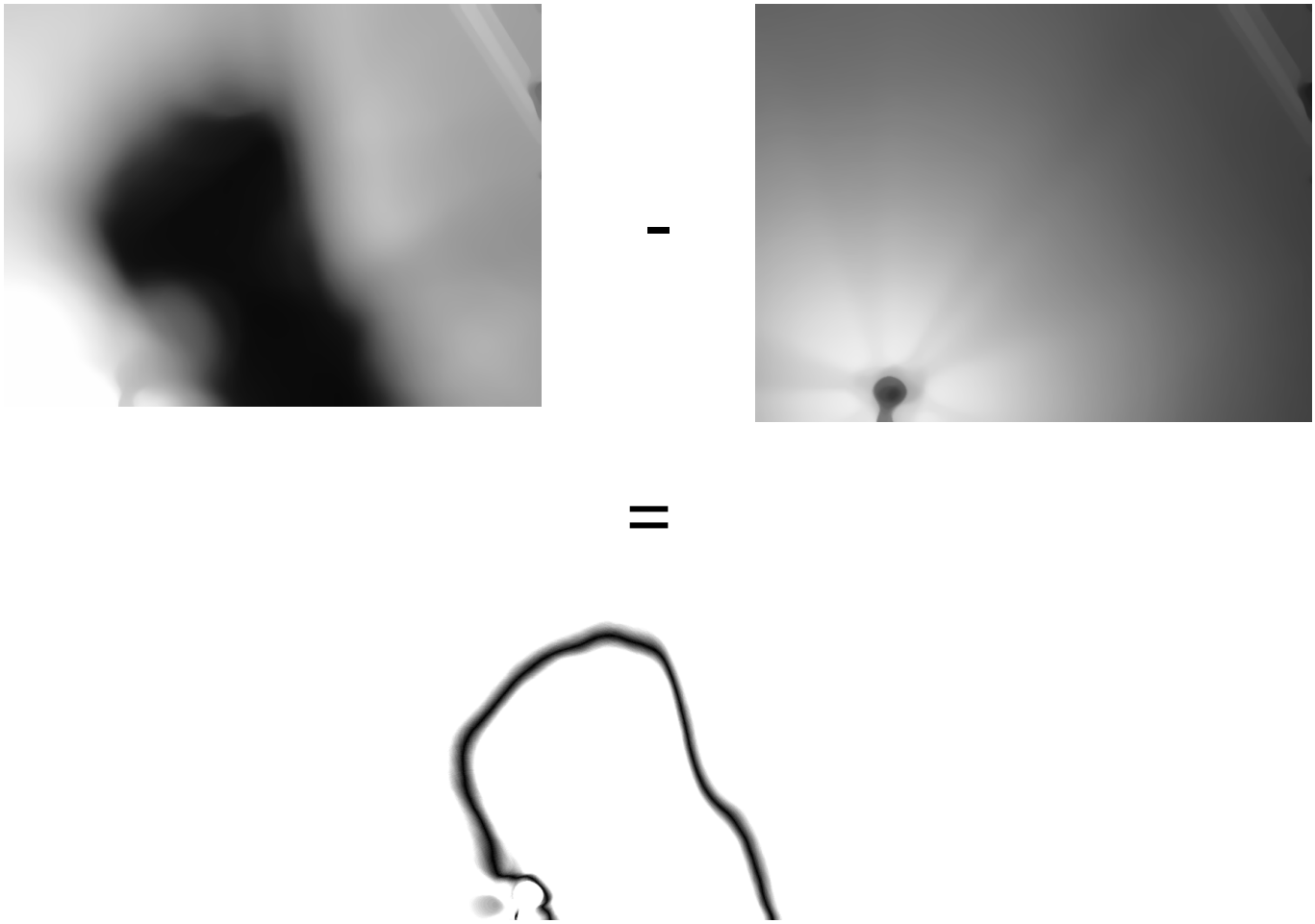


Motion detected
center_x = 401 center_y = 192
Pixel changes = 58719 threshold = 100
std_x = 162.855 std_y = 139.275

False Positive: Bug Land on Lens



Grey Frame – Background Frame = Difference Frame:



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

To get rid of the false positive, you can put a threshold on the standard deviation.

As you can see from the top two examples, when there is a big change in the frame, such as the camera moving or a bug landing on the lens the standard deviation becomes very large. In these examples the standard deviation is 140+, while when a person is walking the standard deviation in roughly 25.

```
 //check to make sure the std are widthin the threshold

  if(param->std_x > 100.0f)
    return 0;
  if(param->std_y > 100.0f)
    return 0;
```

This just is a simple filter to get rid of huge changes in the standard deviation (greater than 100).

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.