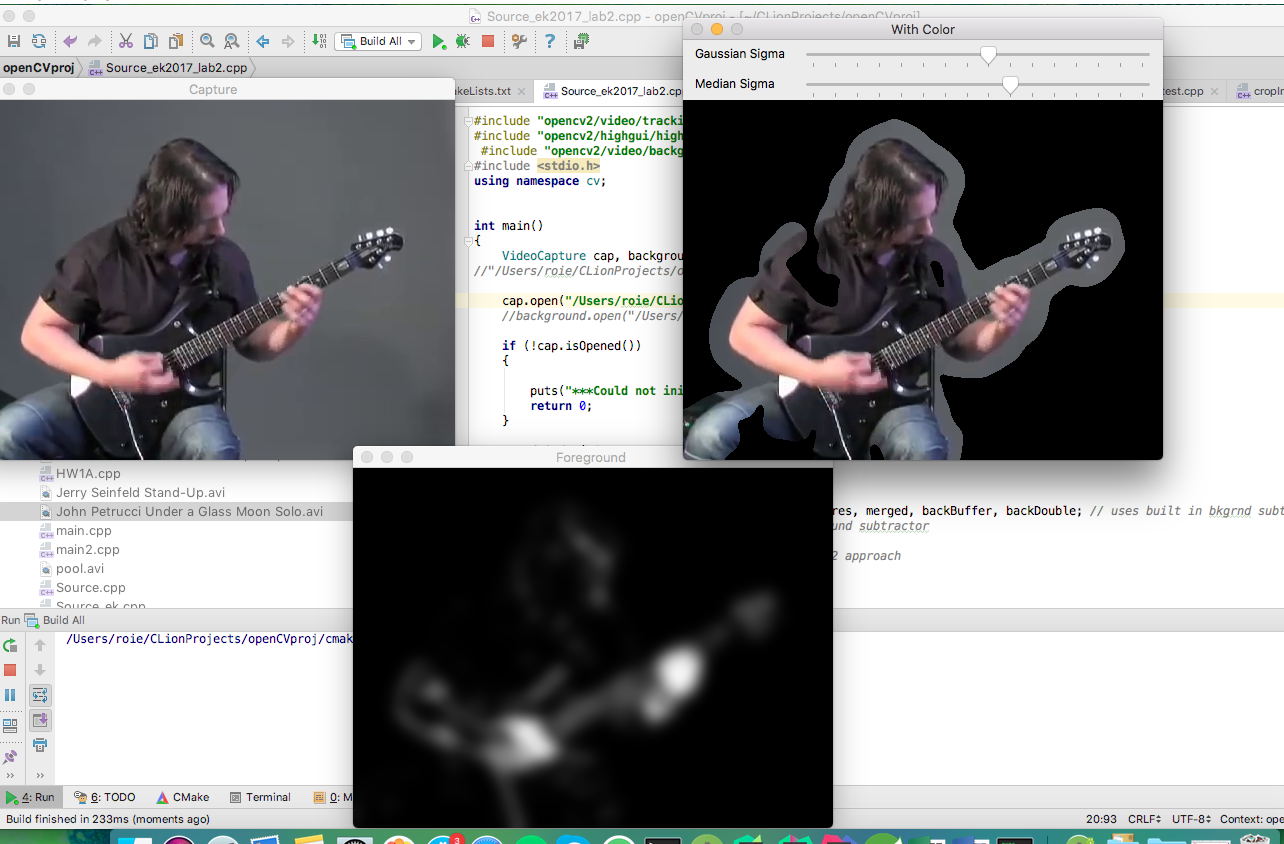
Introduction to Computer Vision - HW1

Moshe Amini – 204000103

Roie Danino – 316440296

17 November 2017

Part A: Cropping an Image/Video/Camera Using the Cursor:

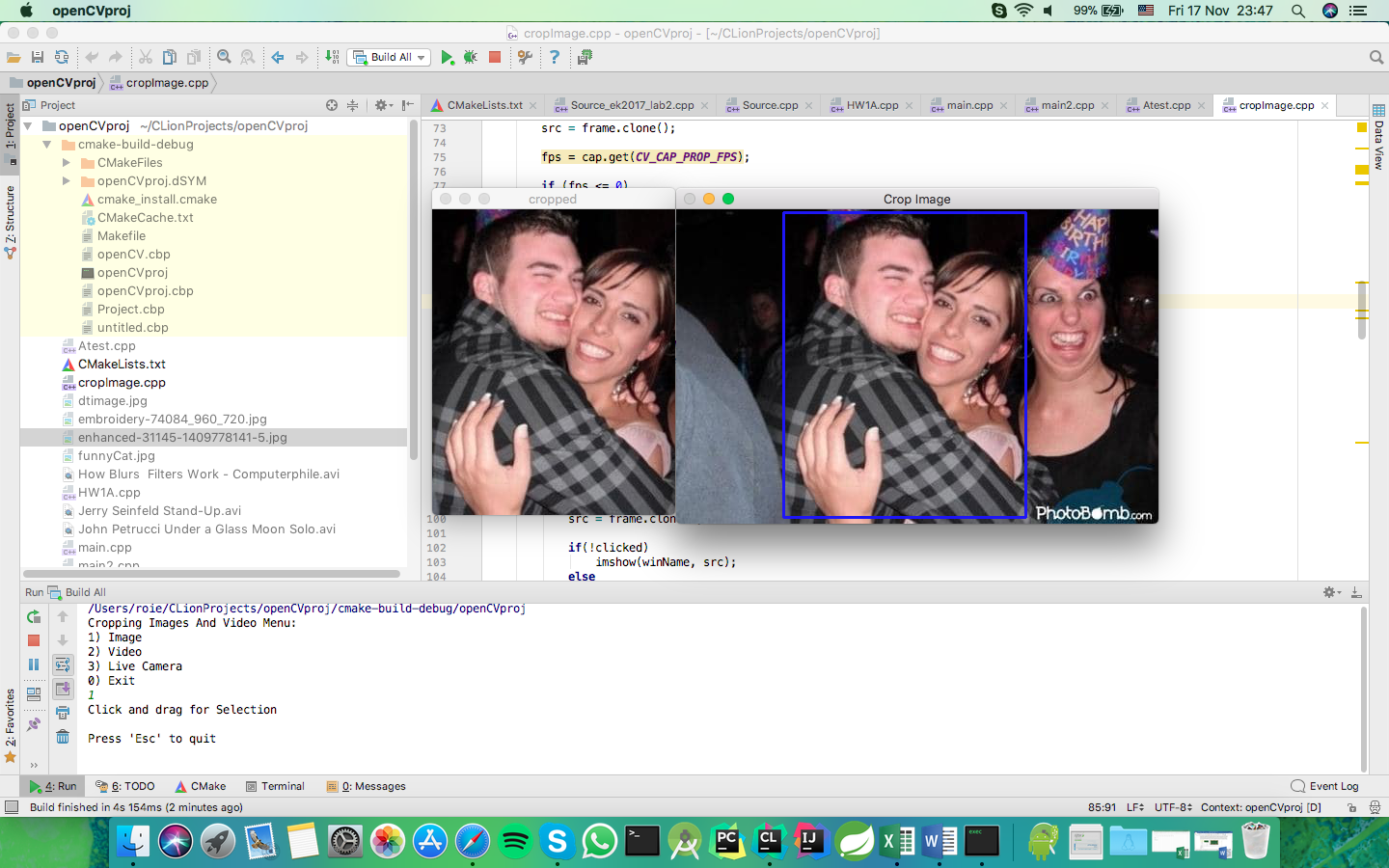
Part B: Video/Camera Background Removal:

**Abstract**

Part A:

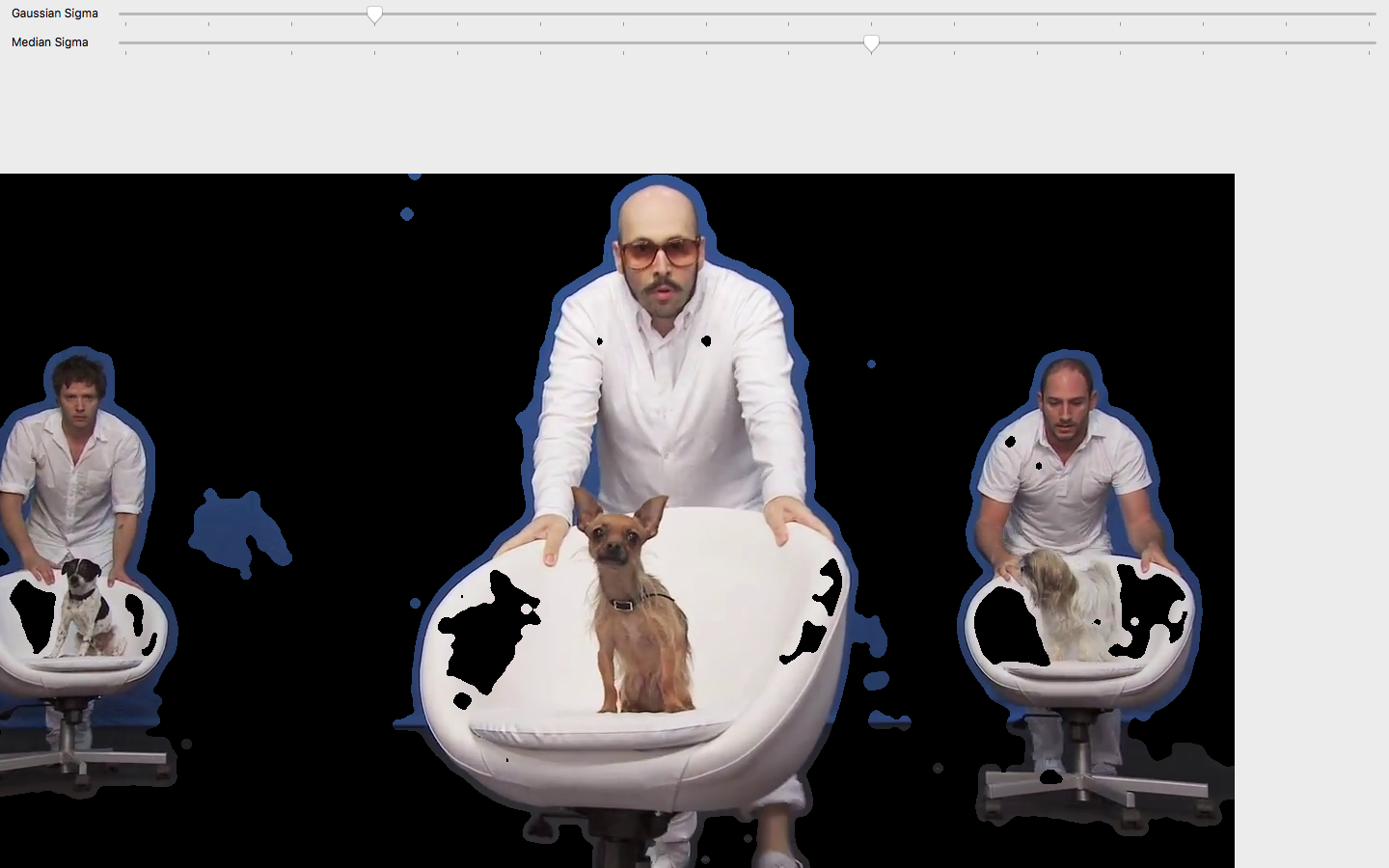
The mission was to create a simple software that showing an image to the user and enables him to draw a rectangle and to crop the selected sub-image from the original one, we also created more options such as cropping a picture from a video and also from a live camera,

The required solution functionality contains GUI elements such as windows and mouse events and also the ability to cut a part of the image and showing it in a separate window.



Part B:

The other mission was to preform something called Background Subtraction which means removing the background of the video using the fact that the background doesn’t move as much as the main objects / people, the required solution functionality was the ability to show (colorfully) the moving parts in a given video, the highlight of the algorithm was the use of MOG (Mixture of Gaussians) and different kinds of blurring (median and Gaussian).



**Introduction and Background**

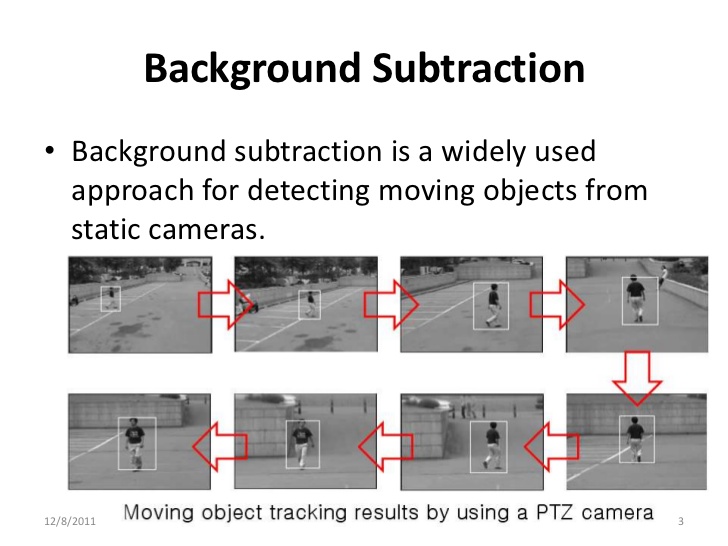
Part A:

**Cropping an image** is a commonly used feature that can be found in almost any image editing / processing software including MS Paint and Photoshop by Adobe,

Cropping an image means cutting a selected squared-slice out of it, it can be used for several reasons like cutting out parts you didn’t like in the picture or making it fit into a specific size.

Part B:

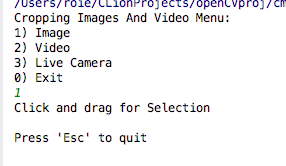
**Video background subtraction** mainly used for detecting moving objects from static camera videos, surveillance tracking and human poses estimation,the algorithmpreforms at its best with computer graphics videos.



**The Algorithm**

**Part A:**

**Cropping an image** from an image isn’t that hard, the first step is to load the photo you want to crop using “imread” function, creating a named-Window so you can refer to it on other lines of code and add mouse listener to it so we would able to draw a rectangle from the first press of the mouse until the user let the cursor button up, to do that we declare 2 Points, P1 for the first corner of the rectangle and P2 to the other, while drawing the rectangle we want to show the user the cropped image in the same time so he can see more clearly what the result is going to look like, for that purpose we created another name-Window that the cropped image will be presented on, and rewrite this photo with every move of the cursor so the window will always be updated.

****In our project, we gave the user the option to choose if he wants to crop the image from

1) an **image** 2) a **video** or 3) a **live camera** with a simple menu before the software starts:

The difference between cropping from a video and cropping from a live camera is minor, but they both very different from cropping an image, while cropping an image means the cropping of a static single image, cropping from a video means that you need to sample a frame every iteration of the loop and to re-write the result window as well.

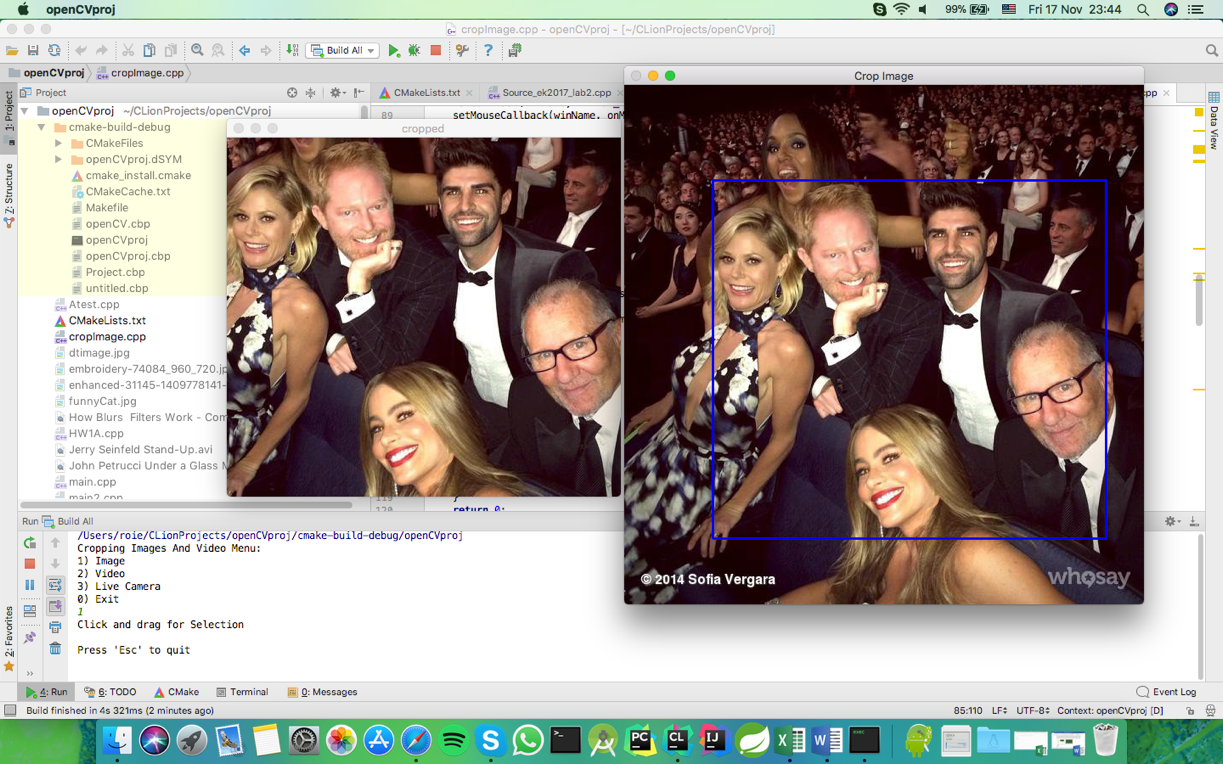
**Part B:**

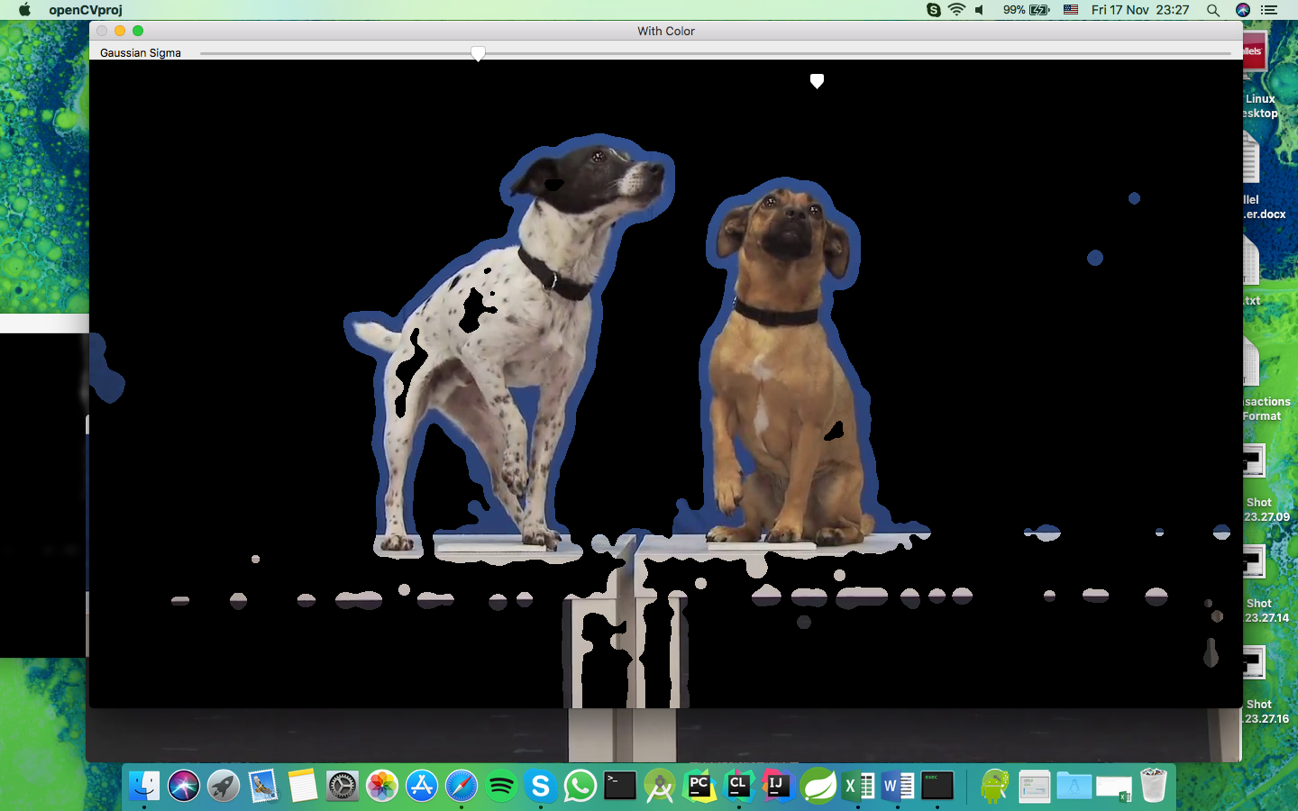
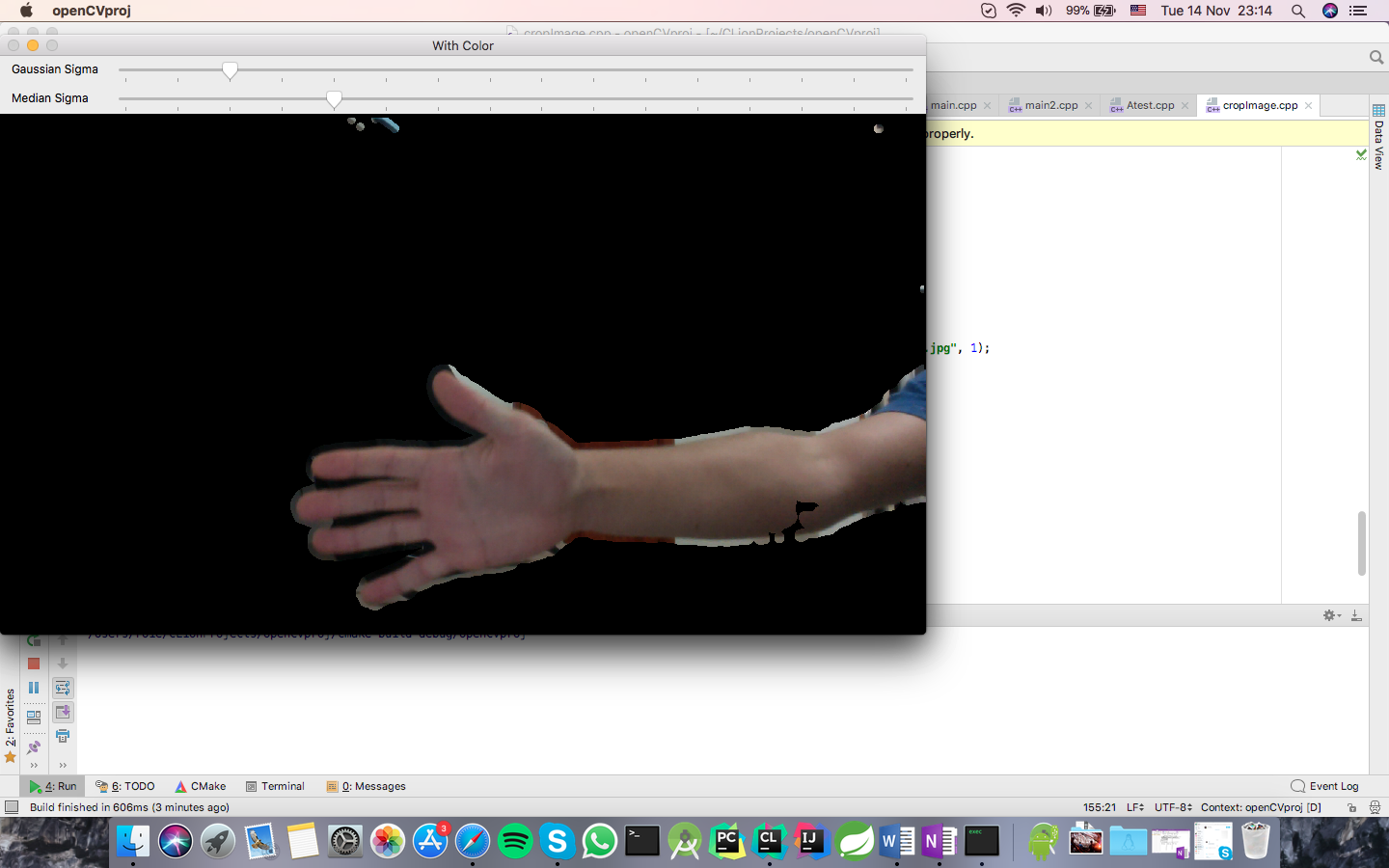
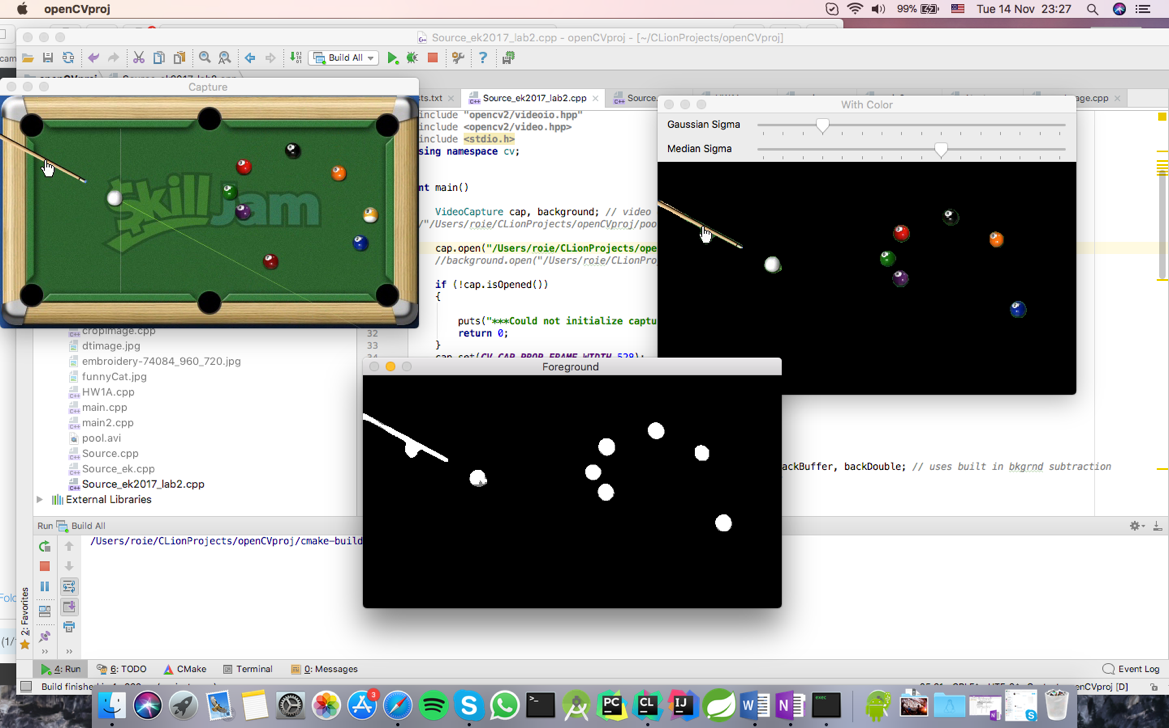
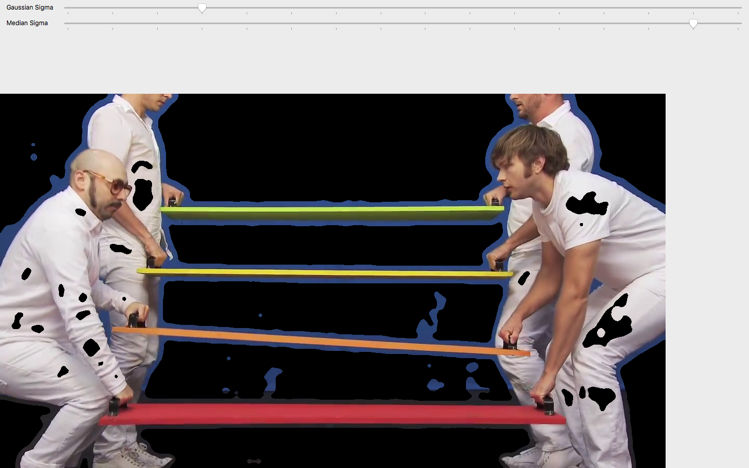
In the background subtraction algorithm, we created 3 windows that shows the user the original video, the foreground (the result of the MOG) and the foreground with full color,

So, again we need to load the input (From the camera or the video) frame by frame and each frame need to go through the MOG (Mixture of Gaussian) function which “remembers” the previous frames and based on them creating a black and white Mat (image) object in such a way that the more the pixel has changed, with respect to the few recent frames, the brighter it would paint it, and with that information and a little bit of help from the erosion and dilation filters and also some different kinds of blur we get a pretty nice mask that tells us where the foreground objects are, and all we need to do is to take the original frame and to use the “copyTo” method with the result of the MOG as a mask, of course the result cannot be perfect with non-computer-graphics videos but for that purpose we created 2 track-bars, one for the Gaussian blur sigma and the other for the median, in that way the user can change the sigma of the different blurs to get to the best result combination while the Median is better for computer graphics videos and the Gaussian is better for camera videos, we also noticed that the Gaussian making the moving object view clearer but also less accurate on the edges, unlike the Median which creating the opposite effect.

Original Foreground (MOG) Merged Result

**Results**

****

****

Live Camera Input

**Summary and Conclusion**

In conclusion, cropping an image is a commonly used and easy to use feature that enables the user to cut out unwanted parts, and the main part of the solution includes the use of mouse events and creating sub – images based on the rectangle that the user draws, in our opinion the ability to choose the input source (Image / Video / Live Camera) making our project unique and was the fun and challenging part of Part A,

The background subtraction is a very interesting feature to implement and also to watch the results, can come handy with surveillance camera because the camera is usually static and the background objects are not moving the most of the time, the solution contained few nice elements while the MOG and the Median & Gaussian blurs were the highlight of this part, in our opinion the ability to change and play with the sigma of the Gaussian blur and the Median blur separately is making this part of the project unique.

**References**

OK Go - White Knuckles

<https://www.youtube.com/watch?v=nHlJODYBLKs>

Awkward photobomb:

[https://img.buzzfeed.com/buzzfeed-static/static/2014-09/3/17/enhanced/webdr07/enhanced-31145-1409778141-5.jpg?downsize=715:\*&output-format=auto&output-quality=auto](https://img.buzzfeed.com/buzzfeed-static/static/2014-09/3/17/enhanced/webdr07/enhanced-31145-1409778141-5.jpg?downsize=715:*&output-format=auto&output-quality=auto)

Modern family photobomb:

<https://i.pinimg.com/originals/04/f8/c6/04f8c6dcefe0779c2d41088551626b17.jpg>

John Petrucci – Under a Glass Moon Solo:

<https://www.youtube.com/watch?v=bVmq2C5kLoM>

How Blurs & Filters Work – Computerphile:

<https://www.youtube.com/watch?v=C_zFhWdM4ic>

**Appendix**

**Part A:**

**int** main()  
{  
 **bool** CAMERA = **true**, VIDEO = **true**;  
 **int** fps = 10, choice;  
 string menu = **"Cropping Images And Video Menu:\n1) Image\n2) Video\n3) Live Camera\n0) Exit\n"**;  
  
 **do** {  
 cout << menu.c\_str();  
 cin >> choice;  
  
 **switch** (choice)  
 {  
 **case** 0:  
 **return** 0;  
 **case** 1:  
 CAMERA = **false**;  
 VIDEO = **false**;  
 **break**;  
 **case** 2:  
 CAMERA = **false**;  
 VIDEO = **true**;  
 **break**;  
 **case** 3:  
 CAMERA = **true**;  
 VIDEO = **true**;  
 **break**;  
 **default**:  
 cout << **"Invalid option try again"** << endl << endl;  
 }  
 } **while**(choice > 3 || choice < 0);  
  
 cout << **"Click and drag for Selection"** << endl << endl;  
 cout << **"Press 'Esc' to quit"** << endl << endl;  
  
  
 **if**(CAMERA)  
 cap.open(0);  
 **else** cap.open(**"/fakePath/pool.avi"**); *//change here to any video you would like to crop* **if** (VIDEO)  
 {  
 cap >> frame;  
 src = frame.clone();  
  
 fps = cap.get(CV\_CAP\_PROP\_FPS);  
  
 **if** (fps <= 0)  
 fps = 10;  
 **else** fps = 1000 / fps;  
 }  
  
 **else** {  
 src = imread(**"fakePath/fakeImage.jpg"**, 1);  
 }  
  
 namedWindow(winName, WINDOW\_NORMAL);  
 setMouseCallback(winName, onMouse, NULL);  
  
 imshow(winName, src);  
  
 **while** (1)  
 {  
  
 **if**(VIDEO)  
 {  
 cap >> frame;  
  
 src = frame.clone();  
  
 **if**(!clicked)  
 imshow(winName, src);  
 **else** imshow(winName,img);  
  
 }

**int** c = waitKey(fps);  
  
 **if**(clicked)  
 {  
 showImage();  
 }

**if** (c == 27)  
 **break**;  
 }  
 **return** 0;  
}

**void** checkBoundary()  
{  
 *//check cropping rectangle exceed image boundary* **if** (cropRect.width > img.cols - cropRect.x)  
 cropRect.width = img.cols - cropRect.x;  
  
 **if** (cropRect.height > img.rows - cropRect.y)  
 cropRect.height = img.rows - cropRect.y;  
  
 **if** (cropRect.x < 0)  
 cropRect.x = 0;  
  
 **if** (cropRect.y < 0)  
 cropRect.height = 0;  
}

**void** showImage()  
{  
 img = src.clone();  
 checkBoundary();  
  
 **if** (cropRect.width > 0 && cropRect.height > 0)  
 {  
 ROI = src(cropRect);  
 imshow(**"cropped"**, ROI);  
 }  
  
 rectangle(img, cropRect, Scalar(255, 0, 0), 2, 8, 0);  
 imshow(winName, img);  
  
}

**void** onMouse(**int** event, **int** x, **int** y, **int** f, **void** \*)  
{  
  
 **switch** (event)  
 {  
 **case** CV\_EVENT\_MOUSEMOVE :  
 **if** (clicked)  
 {  
 P2.x = x;  
 P2.y = y;  
 }  
 **break**;  
  
 **case** CV\_EVENT\_LBUTTONDOWN :  
 clicked = **true**;  
  
 P1.x = x;  
 P1.y = y;  
 P2.x = x;  
 P2.y = y;  
  
 **break**;  
  
 **case** CV\_EVENT\_LBUTTONUP :  
 P2.x = x;  
 P2.y = y;  
 clicked = **false**;  
  
 **break**;  
  
 **default**:  
 **break**;  
 }  
  
  
 **if** (clicked)  
 {  
 **if** (P1.x > P2.x)  
 {  
 cropRect.x = P2.x;  
 cropRect.width = P1.x - P2.x;  
  
 }  
 **else** {  
 cropRect.x = P1.x;  
 cropRect.width = P2.x - P1.x;  
 }  
  
 **if** (P1.y > P2.y)  
 {  
 cropRect.y = P2.y;  
 cropRect.height = P1.y - P2.y;  
 }  
 **else** {  
 cropRect.y = P1.y;  
 cropRect.height = P2.y - P1.y;  
 }  
 showImage();  
 }  
  
}

**Part B:**

**int** main()  
{  
 **bool** CAMERA;  
 **int** choice, sigma = 3,ksize, median = 9;  
 **double** fps;  
 VideoCapture cap; *// video capture source* string menu = **"Background Subtraction from a Video and Live Camera Menu:\n1)Video\n2)Live Camera\n0) Exit\n"**;  
  
 **do** {  
 cout << menu.c\_str();  
 cin >> choice;  
  
 **switch** (choice)  
 {  
 **case** 0:  
 **return** 0;  
 **case** 1:  
 CAMERA = **false**;  
 **break**;  
 **case** 2:  
 CAMERA = **true**;  
 **break**;  
  
 **default**:  
 cout << **"Invalid option try again"** << endl << endl;  
 }  
  
 } **while**(choice > 2 || choice < 0);  
  
 **if**(CAMERA)  
 cap.open(0);  
 **else** cap.open(**"/fakePath/fakeVideo.avi"**,***WINDOW\_GUI\_EXPANDED***);  
  
 **if** (!cap.isOpened())  
 {  
  
 puts(**"\*\*\*Could not initialize capturing...\*\*\*\n"**);  
 **return** 0;  
 }  
  
 namedWindow(**"Capture "**, ***CV\_WINDOW\_FREERATIO***);  
 namedWindow(**"Foreground "**, ***CV\_WINDOW\_FREERATIO***);  
 namedWindow(**"With Color"**,***CV\_WINDOW\_FREERATIO***);  
  
 Mat frame, foreground, invertForground, image, res, merged; Ptr<BackgroundSubtractor> pMOG2; *//MOG2 Background subtractor  
 //create Background Subtractor objects* pMOG2 = createBackgroundSubtractorMOG2(); *//MOG2 approach* fps = cap.get(***CV\_CAP\_PROP\_FPS***);  
   
 **if** (fps <= 0)  
 fps = 10;  
 **else** fps = 1000 / fps;  
  
 createTrackbar( **"Gaussian Sigma"**, **"With Color"**, &sigma, 15 );  
 createTrackbar( **"Median Sigma"**, **"With Color"**, &median,15);  
  
 **for** (;;)  
 {  
 cap >> frame; *// assign* **if** (frame.empty())  
 **break**;  
 image = frame.clone(); *//to avoid overwrite by the next frame* res = image.clone(); *//double buffered* pMOG2->apply(res, foreground, -1);  
  
 threshold(foreground, foreground, 20, 255, ***THRESH\_TOZERO***);  
  
 ksize = (sigma\*5)|1;  
  
 median |= 1;  
  
 GaussianBlur(foreground, foreground, Size(ksize, ksize), sigma, sigma);  
  
 medianBlur(foreground, foreground, median);  
  
 erode(foreground, foreground, Mat());  
 dilate(foreground, foreground, Mat());  
  
 threshold(merged, merged, 20, 255, ***THRESH\_MASK***);  
  
 res.copyTo(merged,foreground);  
  
 imshow(**"Capture "**, image);  
  
 imshow(**"Foreground "**, foreground);  
  
 imshow(**"With Color"**, merged);  
  
 **char** c = (**char**)waitKey((**int**)fps);  
 **if** (c == 27) *// ESC* **break**;  
  
 }  
}