Seed Counting Application for Android Device

CIS 690 Final Report

by

Zhiqiang Xiong

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

05/13/2017

Approved by:

Professor
Dr. Mitchell L. Neilsen

SEED COUNTING APPLICATION FOR ANDROID DEVICE

FINAL REPORT

INTRODUCTION

The application is mainly focused on tracking and counting seeds that mainly used in agricultural. The code consists of background subtraction, noise decreasing, seeds detecting, contours drawing, track and count. The following is some information regarding what each of these processes is about,

1. **Background Subtraction**: subtract background that is static and unchanging over consecutive frames of a video [1].
2. **Noise Decrease**: decrease shadowing, reflections, lighting conditions, and any other possible change in the environment.
3. **Seeds Detecting**: detect each substantial change and save it as a seed.
4. **Contours Drawing**: draw contour and center coordinate for seeds.
5. **Track and Count**: Track seeds by coordinates and count them when they pass by the line.

The major function of the application is to track the seeds and count the amount of them. If we have got the center coordinates of seeds, then the details of tracking and counting are as follows,

i. Compare coordinates between 2 continue frames, if they are close enough, recognize they are the same seed.
ii. Save all coordinates of each seed in an array.
iii. Compare coordinates of each seed, if the coordinate before is above than line we set and the coordinate after is below than the line, then counter will plus one.

The application has a varied user base. It not only can be used in agriculture to calculate yield, but also in traffic to detect traffic flow. What's more, it also works in supermarket to detect people flow. It will be a useful tool by roadways and transportation officials to improve road and traffic situations respectively. So if the situation needs motion detection and counting the objects, then the application will work well technologically.

The application based on Python and OpenCV.

SETTING UP OPENCV AND PYTHON

Introduction of OpenCV

OpenCV is an Image Processing computer vision library consisting of several programming functions to meet the requirements of real-time Computer Vision. In order to use OpenCV, we need to download and set it up in our project. There are several versions of OpenCV with numerous enhancements. The latest version is OpenCV 3.2, released on 23rd

December, 2016. However, the version 3.0 is used predominantly because it is closed to the latest version and more stable than it.

Introduction of Python

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java.[2][3] The language provides constructs intended to enable writing clear programs on both a small and large scale.[4] The version 2.7 is used predominantly in this program.

Step1. Set up environment [5]

After downloading and install Xcode, we need to install Homebrew, which is labeled as "The missing package manager for OSX" (and they really are not joking about that one). Think of Homebrew as an (almost) equivalent of apt-get for Ubuntu.

To install Homebrew, simply head to the Homebrew website and simply copy and paste the command underneath the "Install Homebrew" section into your terminal:

```
$ cd ~
$ $ ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebr
ew/install/master/install)"
delimiter //
```

Now that Homebrew is installed, you'll need to update it and grab the latest package (i.e. "formula") definitions. These formula are simply instructions on how to install a given package or library.

To update Homebrew, simply execute:

```
$ brew update
```

Step2. Install Python

Use Homebrew to install our user-specific version of Python 2.7:

```
$ brew install python
```

Step3. Setup Python

However, before we proceed, we need to update our PATH in our ~/.bash_profile file to indicate that we want to use Homebrew packages before any system libraries or packages. This is an absolutely critical step, so be sure not to skip it!

Open up your ~/.bash_profile file in your favorite editor (if it does not exist, create it), and append the following lines to the file:

```
$ export PATH=/usr/local/bin:$PATH
```

From there, reload your ~./bash_profile file to ensure the changes have been made:

```
$ source ~/.bash_profile
$ which python
/usr/local/bin/python
```

If your output of which python is /usr/local/bin/python , then you are indeed using the Homebrew version of Python. And if your output is /usr/bin/python , then you are still using the system version of Python — and you need to go back and ensure that your ~/.bash_profile file is updated and reloaded correctly.

Installing some Python packages. We need to install NumPy since the OpenCV Python bindings represent images as multi-dimensional NumPy arrays:

```
$ pip install numpy
```

Step4. Setup Opencv

```
$ brew install cmake pkg-config
$ brew install jpeg libpng libtiff
openexr
$ brew install eigen tbb
```

Step5. Install OpenCV

Change directory to our home directory, followed by pulling down OpenCV from GitHub, and checking out the 3.0.0 version:

```
$ cd ~
$ git clone
https://github.com/Itseez/opencv.git
$ cd opencv
$ git checkout 3.0.0
$ cd ~/opencv
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local \
        -D
PYTHON2_PACKAGES_PATH=~/.virtualenvs/cv/
lib/python2.7/site-packages \
        -D
PYTHON2_LIBRARY=/usr/local/Cellar/python
/2.7.10/Frameworks/Python.framework/Vers
ions/2.7/bin \
        -D
PYTHON2_INCLUDE_DIR=/usr/local/Framework
s/Python.framework/Headers \
        -D INSTALL_C_EXAMPLES=ON -D
INSTALL_PYTHON_EXAMPLES=ON \
        -D BUILD_EXAMPLES=ON \
        -D
OPENCV_EXTRA_MODULES_PATH=~/opencv_contr
ib/modules ..
```

There are some very important options we are supplying to CMake here, so let's break them down:

CMAKE_BUILD_TYPE : This option indicates that we are building a release binary of OpenCV.

CMAKE_INSTALL_PREFIX : The base directory where OpenCV will be installed.

PYTHON2_PACKAGES_PATH : The explicit path to where our site-packages directory lives in our cv virtual environment.

PYTHON2_LIBRARY : Path to our Hombrew installation of Python.

PYTHON2_INCLUDE_DIR : The path to our Python header files for compilation.

INSTALL_C_EXAMPLES : Indicate that we want to install the C/C++ examples after compilation.

INSTALL_PYTHON_EXAMPLES : Indicate that we want to install the Python examples after complication.

BUILD_EXAMPLES : A flag that determines whether or not the included OpenCV examples will be compiled or not.

OPENCV_EXTRA_MODULES_PATH : This option is extremely important — here we supply the path to the opencv_contrib repo that we pulled down earlier, indicating that OpenCV should compile the extra modules as well.

Now that CMake has properly configured the build, we can compile OpenCV:

```
$ make -j4
$ sudo make install
```

SYSTEM IMPLEMENTATION

The system is implemented using a five-processes architecture consisting of the background subtraction, noise decreasing, seeds detecting, contours drawing, track and count.

1. **Background subtraction**:

Background subtraction is critical in many computer vision applications. We use it to count the number of cars passing through a toll booth. We use it to count the number of people walking in and out of a store. And we use it for motion detection.

There are many, many ways to perform motion detection, tracking, and analysis in OpenCV. Some are very simple. And others are very complicated. The two primary methods are forms of Gaussian Mixture Model-based foreground and background segmentation:

An improved adaptive background mixture model for real-time tracking with shadow detection by KaewTraKulPong et al., available through the cv2.BackgroundSubtractorMOG function.

Improved adaptive Gaussian mixture model for background subtraction by Zivkovic, and Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction, also by Zivkovic, available through the cv2.BackgroundSubtractorMOG2 function.

And in newer versions of OpenCV we have Bayesian (probability) based foreground and background segmentation, implemented from Godbehere et al.'s 2012 paper, Visual Tracking of Human Visitors under Variable-Lighting Conditions for a Responsive Audio Art Installation. We can find this implementation in the cv2.createBackgroundSubtractorGMG function (we'll be waiting for OpenCV 3 to fully play with this function though).

Well, in motion detection, we tend to make the following assumption:

The background of our video stream is largely static and unchanging over consecutive frames of a video. Therefore, if we can model the background, we monitor it for substantial changes. If there is a substantial change, we can detect it — this change normally corresponds to motion on our video.

Assumption: The first frame of our video file will contain no motion and just background — therefore, we can model the background of our video stream using only the first frame of the video.

Here's an example of the first frame of an example video:

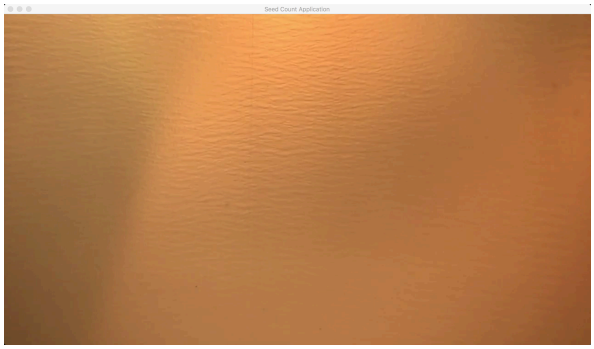

Figure 1-1 The first frame of video

a) We first change the frame into gray so that decrease the influence of change of the light condition.

```
    # Background substraction    #
    while(cap.isOpened()):
        ret, frame = cap.read()   #read a
    frame
        try:
```

```
gray = cv2.cvtColor(frame,
cv2.COLOR_BGR2GRAY)   #change to gray,
```

And the result is as below:

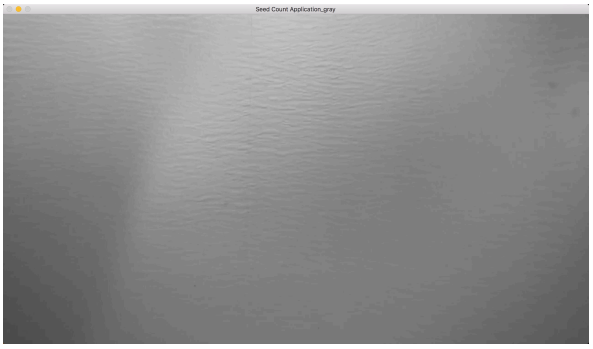

Figure 1-2 The first gray frame of video

Figure 1-3 Sample frame of video

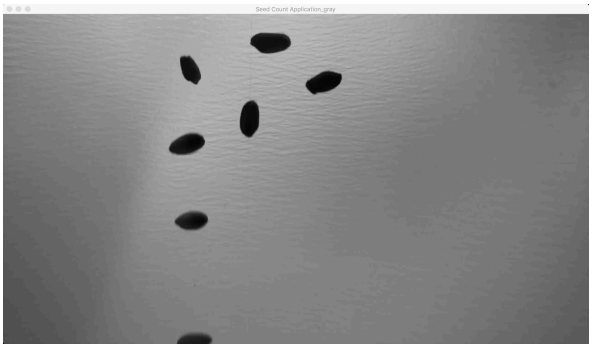

Figure 1-4 Sample gray frame of video

b) Computing the difference between two frames is a simple subtraction, where we take the absolute value of their corresponding pixel intensity differences.

```
# Background substraction    #
# compute the absolute difference
between the current frame and first
frame
frame2 = cv2.absdiff(firstFrame,gray)
```

And the result is as below:

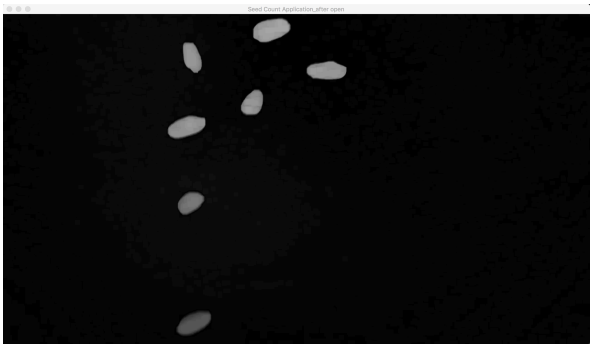

Figure 1-5 Sample background subtraction frame

c) Then we use the OpenCV function morphologyEx to apply Morphological Transformation such as: Opening and closing.

To do these operations you also need to specify a kernel or structuring element (strel). This is a matrix that is convoluted on the image of size n*n that defines the area to use when calculating the value of each pixel.

Opening

It is obtained by the erosion of an image followed by dilation.

$$dst = open(src, element) = dilate(erode(src, element))$$

Useful for removing small objects (it is assumed that the objects are bright on a dark foreground)

For instance, check out the example below. The image at the left is the original and the image at the right is the result after applying the opening transformation. We

can observe that the small spaces in the corners of the letter tend to disappear.

Figure 1-6 Sample image for opening

For the sake of clarity, we have performed the opening operation (7x7 rectangular structuring element) on the same original image but inverted such as the object in white is now the letter.

Figure 1-7 Sample image for opening

```
# Background substraction    #
kernel = np.ones((9,9),np.uint8)
opening = cv2.morphologyEx(thresh1,
cv2.MORPH_OPEN, kernel)
```

The result we get after opening is shown as below:

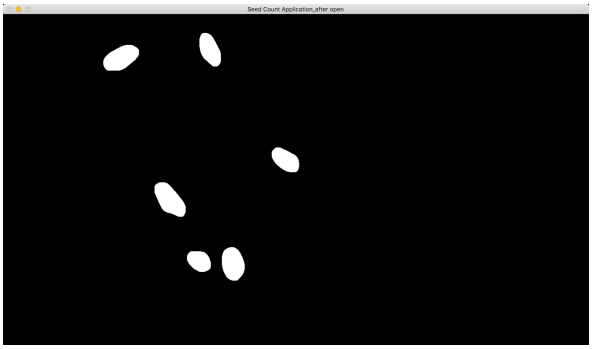

Figure 1-8 Sample frame after opening

Closing

It is obtained by the dilation of an image followed by erosion.

$$dst = close(src, element) = erode(dilate(src, element))$$

Useful to remove small holes (dark regions).

Figure 1-9 Sample image for closing

On the inverted image, we have performed the closing operation (7x7 rectangular structuring element): [6]

Figure 1-10 Sample image for closing

```
# Background substraction    #
kernel = np.ones((9,9),np.uint8)
closing = cv2.morphologyEx(opening,
cv2.MORPH_CLOSE, kernel)
```

The result we get after closing is shown as below:

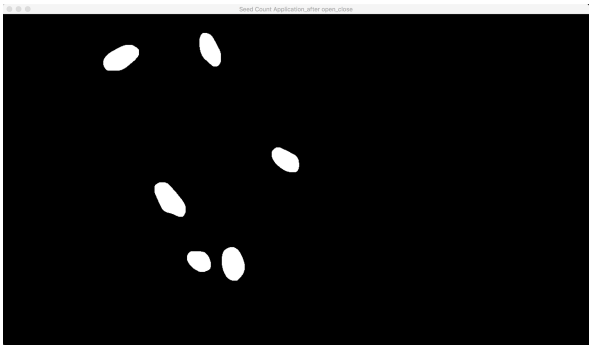

Figure 1-11 Sample frame after closing

2. **Noise decreasing**:

After background subtraction, we have eliminated most noise. Now we need to deal with the little noise as well.

a) We first change the gray frame into blur gray frame so that decrease the influence of change of the light condition much more.

```
# Noise decreasing    #
gray = cv2.cvtColor(frame,
cv2.COLOR_BGR2GRAY)   #change to gray,
gray = cv2.GaussianBlur(gray, (21, 21),
0)    #blur , decrease more noise
```

And the result is as below:

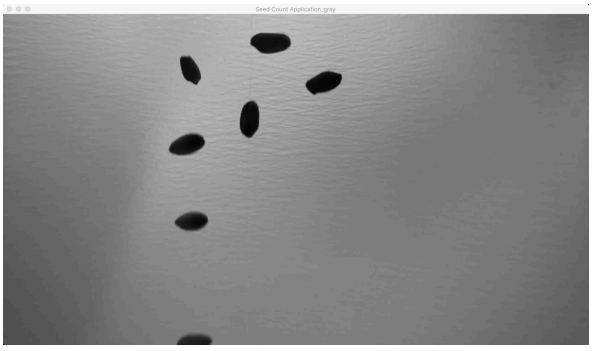

Figure 2-1 Sample gray frame before blur

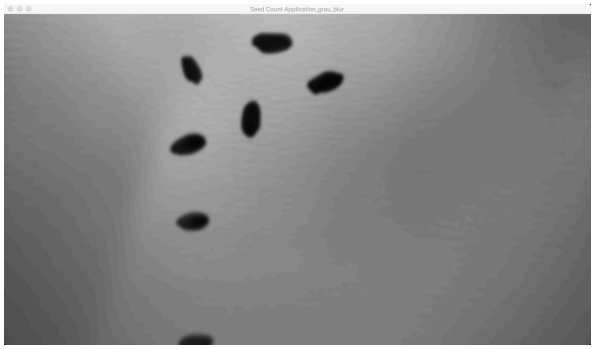

Figure 2-2 Sample gray frame after blur

b) We´re using the threshold method to binarize the image. Convert it from color to only black and white, two values (not the same as gray scale).

Simple Thresholding is straight forward. If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). The function used is cv2.threshold. First argument is the source image, which should be a grayscale image. Second argument is the threshold value which is used to classify the pixel values. Third argument is the maxVal which represents the value to be given if pixel value is more than (sometimes less than) the threshold value. OpenCV provides different styles of thresholding and it is decided by the fourth parameter of the function. Different types are:

cv2.THRESH_BINARY

cv2.THRESH_BINARY_INV

cv2.THRESH_TRUNC

cv2.THRESH_TOZERO

cv2.THRESH_TOZERO_INV

To plot multiple images, we have used plt.subplot() function. Please checkout Matplotlib docs for more details.[7]

Result is given below:

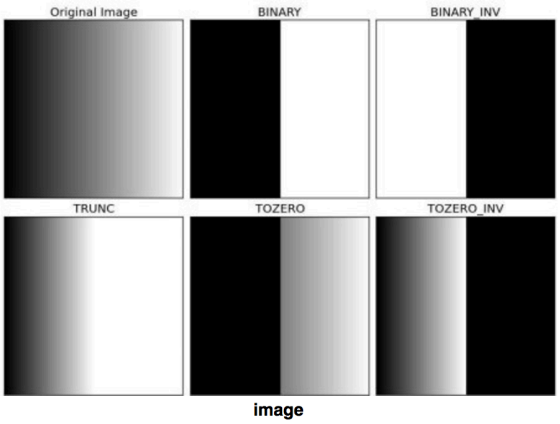

Figure 2-3 Sample image for threshold

3. **Seed detecting**:

It's time to detect contours for seeds on the frames. This is really simple with OpenCV's findContours function.

a) We first need to find the contours for seeds.

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

For better accuracy, use binary images. So before finding contours, apply threshold or canny edge detection.

findContours function modifies the source image. So if you want source image even after finding contours, already store it to some other variables.

In OpenCV, finding contours is like finding white object from black background. So remember, object to be found should be white and background should be black.

```
#     Detect seeds    #
frame,   _, contours0, hierarchy =
cv2.findContours(closing,cv2.RETR_EXTERNA
L,cv2.CHAIN_APPROX_NONE)
# detect contours for seeds
```

There are three arguments in cv2.findContours() function, first one is source image, second is contour retrieval mode, third is contour approximation method. And it outputs the contours and hierarchy. contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.[8]

b) However, there some small noises that we don't consider, then we set a range area for legal seeds and ignore other objects.

```
#     Detect seeds    #
for cnt in contours0:
         # if the area is too small or
too large, skip this
        if area< int(areaL) or area>
int(areaH) :
            continue
       cv2.drawContours(frame, cnt, -1,
(0,255,0), 3, 8) # draw contours of
every seeds
```

And the result is as below:

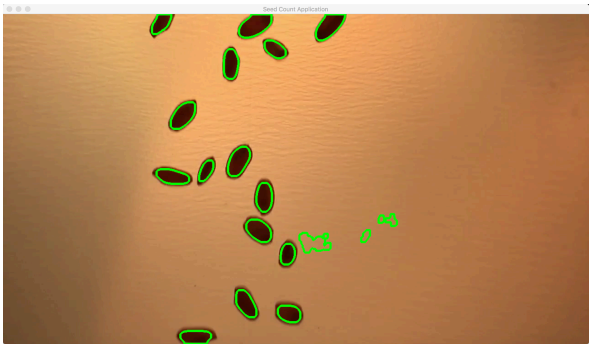

Figure 3-1 Sample contour frame before threshold

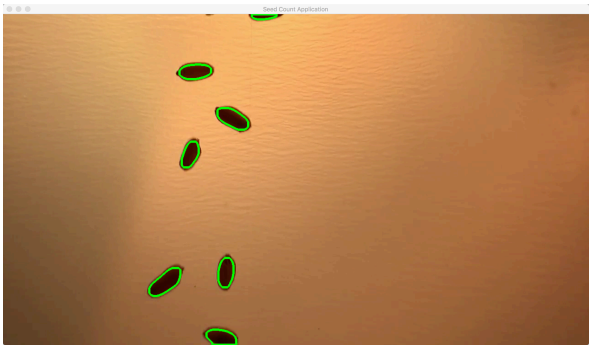

Figure 3-2 Sample contour frame after threshold

4. **Draw contours**:

After we fine the contours for seeds, then need to draw them on frames. Now we will draw contours for seeds on the frames and get center coordinate for each seed.

a) Find the center coordinate for each seed. Image moments help you to calculate some features like center of mass of the object, area of the object etc.

The function cv2.moments() gives a dictionary of all moment values calculated.

From this moments, you can extract useful data like area, centroid etc. Centroid is given by the relations, $C_x = \frac{M_{10}}{M_{00}}$ and $C_y = \frac{M_{01}}{M_{00}}$. This can be done as follows:

```
#     Draw contours    #
#get the centroid of the contours
cx = int(M['m10']/M['m00'])
cy = int(M['m01']/M['m00'])
cv2.circle(frame,(cx,cy), 5, (0,0,255), -
1)# get and draw the center of the
contours
#get the rectangle of contours
x,y,w,h = cv2.boundingRect(cnt)
frame =
cv2.rectangle(frame,(x,y),(x+w,y+h),(0,25
5,0),2) # get and draw the rectangle of
contours
```

And the result is as below:

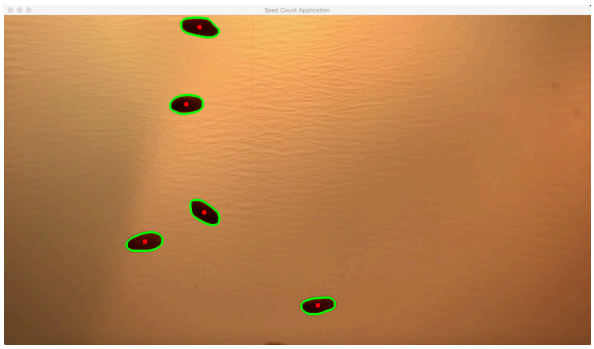

Figure 4-1 Sample contour frame after drawing center coordinate

b) Then we will draw bounding rectangles for each seed. There are two types of bounding rectangles.

b.1. Straight Bounding Rectangle

It is a straight rectangle, it doesn't consider the rotation of the object. So area of the bounding rectangle won't be

minimum. It is found by the function cv2.boundingRect().

Let (x,y) be the top-left coordinate of the rectangle and (w,h) be its width and height.

```
#     Straight bounding rectangle    #
x,y,w,h = cv2.boundingRect(cnt)
cv2.rectangle(img,(x,y),(x+w,y+h),(0,255
,0),2)
```

b.2. Rotated Rectangle

Here, bounding rectangle is drawn with minimum area, so it considers the rotation also. The function used is cv2.minAreaRect(). It returns a Box2D structure which contains following detals - (center (x,y), (width, height), angle of rotation). But to draw this rectangle, we need 4 corners of the rectangle. It is obtained by the function cv2.boxPoints()

```
#     Rotated rectangle    #
rect = cv2.minAreaRect(cnt)
box = cv2.boxPoints(rect)
box = np.int0(box)
cv2.drawContours(img,[box],0,(0,0,255),2
)
```

Both the rectangles are shown in a single image. Green rectangle shows the normal bounding rect. Red rectangle is the rotated rect. [9]

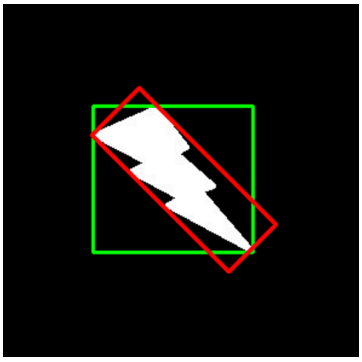

Figure 4-2 Sample image for rectangles

We choose the first one that is straight bounding rectangle.

```
#     Draw contours    #
x,y,w,h = cv2.boundingRect(cnt)   #get
the x,y,weight, height of the bounding
frame =
cv2.rectangle(frame,(x,y),(x+w,y+h),(0,2
55,0),2) # get and draw the rectangle of
contours
```

And the result is as below:

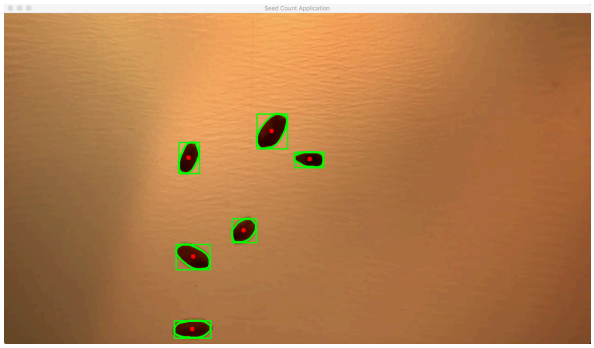

Figure 4-3 Sample contour frame after drawing center coordinate and rectangle contour

5. **Track and count**:

We already know where there's a seed in the image so far. So next is to track every seed and save all coordinates of each seed into an array. Then draw some lines and wait for the seed based on the array with coordinates, once it passes the line, the counter of lines will plus one.

a) In the first frame when we detect seed, we will give that seed an ID and store it's initial coordinate in the image.

Then, on the following frames, we want to keep track of that person, then we need to match the seed's contour in the following frames to the ID we set when it

first appeared, as well as keep storing that seed's coordinates.

To handle all of the IDs and storing of coordinates I created a class called Seed. The code is attached at the end of report.

```
#     Track seeds    #
for i in seeds:
   if len(i.getTracks()) >= 2:
        pts = np.array(i.getTracks(),
np.int32) # make tracks become ndarray
        pts = pts.reshape((-1,1,2))
   if (abs(cx - i.getX()) <= 18  and
(cy-i.getY()) <=int(speed)*h*0.83 ):
#The object is near one already detected
before
        new = False
        i.updateCoords(cx,cy)
```

We compare pairs coordinates in the array and if the range of abs(cx – i.getX()) and (cy – i.getY()) is in the range we set, which means that the coordinates between two continue frames are close enough, then consider them as one seed.

And the result is as below:

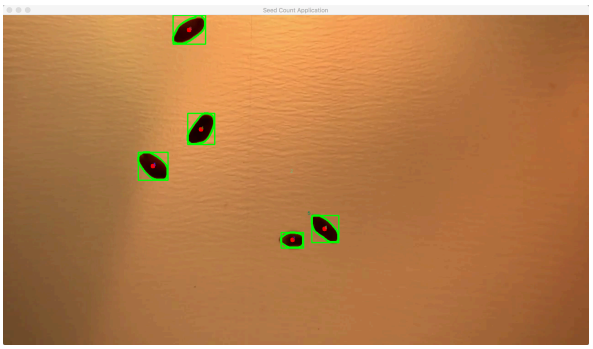

Figure 5-1 Sample contour frame after getting ID

b) Then we will draw some lines in every frame. And once the seed passes by the line, counter of the line will plus one.

b.1. Draw lines

Get the width and height of the frames. Then we divide the frame into 4 parts by drawing 3 lines in Y direction.

```
#     Count seeds    #
w_v = int(cap.get(3)) #get width of the
video
h_v = int(cap.get(4)) #get height of the
video
line_down1 = int(h_v/4)   # line1 to
detect seeds
line_down2 = int(2*h_v/4)   # line2 to
detect seeds
line_down3 = int(4*h_v/5)   # line3 to
detect seeds
frame = cv2.line(frame, (0,line_down1),
(w_v,line_down1), (0,255,0),
thickness=2) #draw a green line at 1/4
place to detect seeds
frame = cv2.line(frame, (0,line_down2),
(w_v,line_down2), (0,255,0),
thickness=2) #draw a green line at 2/4
place to detect seeds
frame = cv2.line(frame, (0,line_down3),
(w_v,line_down3), (0,255,0),
thickness=2) #draw a green line at 3/4
place to detect seeds
```

And the result is as below:

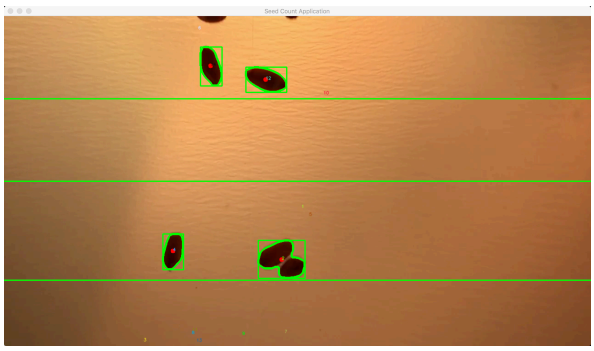

Figure 5-2 Sample contour frame after getting ID and lines

b.2. Count seed after passing by the line

We use three values as counters for each line. [10]

```
#     Count seeds    #
# give different lines and counters
num_seed1   = 0
num_seed2   = 0
num_seed3   = 0
str_down1 = 'DOWN1: '+ str(num_seed1)
str_down2 = 'DOWN2: '+ str(num_seed2)
str_down3 = 'DOWN3: '+ str(num_seed3)
cv2.putText(frame,
str_down1 ,(10,line_down1-
20),font,0.5,(255,0,0),1,cv2.LINE_AA)
cv2.putText(frame,
str_down2 ,(10,line_down2-
20),font,0.5,(255,0,0),1,cv2.LINE_AA)
cv2.putText(frame,
str_down3 ,(10,line_down3-
20),font,0.5,(255,0,0),1,cv2.LINE_AA)
```

And the result is as below:

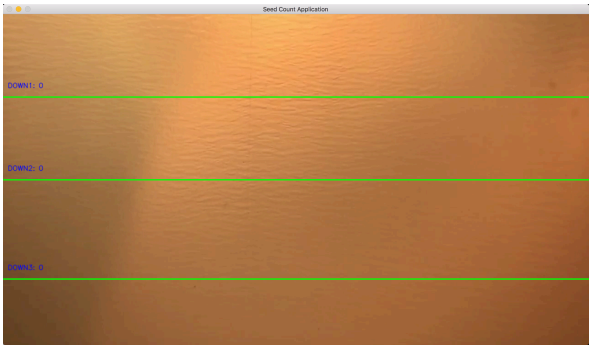

Figure 5-3 Sample frame after getting lines and counters

Then we use three if statements for detecting whether seeds have passed by the line. If it passes, then the counter will plus one.

```
#     Count seeds    #
if (abs(cx - i.getX()) <= 18  and (cy-
i.getY()) <=int(speed)*h*0.83 ):
   new = False
   i.updateCoords(cx,cy)
   if
(i.going_DOWN(line_down1,line_down1) ==
True):
      num_seed1 += 1
   if
(i.going_DOWN(line_down2,line_down2) ==
True):
      num_seed2 += 1
   if
(i.going_DOWN(line_down3,line_down3) ==
True):
      num_seed3 += 1
```

And the result is as below:

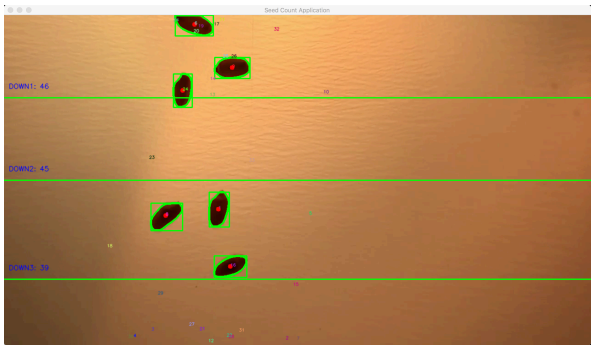

Figure 5-4 Sample frame for counting

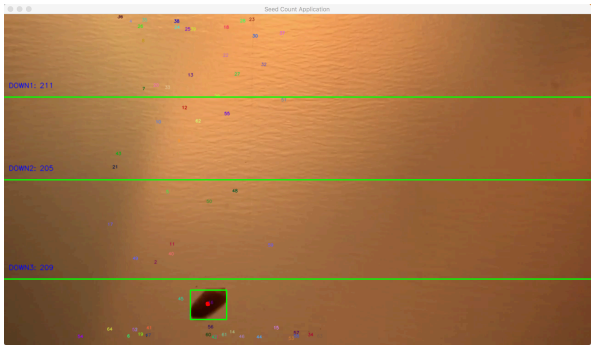

Figure 5-5 Sample frame for final counting

SYSTEM EVALUATION

We evaluated the system based on the performance of how close does the counting amount be compared to actual one. After testing many times and we find the best range for area, range for speed and range for decreasing noise. We have got 211, 205, 209 for three lines. This is reasonable because if there are some seeds are really close to each other, then the system will consider it as one seed. And we have noticed that it is more possible to be closer when seeds are in the middle part of frame. So we will choose the maximum of three counters we have as our final counting result. And the actual number of the test video we have is 217. So the percent of our counting number is near 97.2%, it is really close to the actual amount.

And the algorithm uses mainly the libraries of OpenCV and packages for image processing, so the delay for the algorithm is little. It is likely to apply this application into Android Device for real-time counting, but we still need to be careful because the CPU and system of Android Device is much cheaper and less stable than computer.

The strengths and limitations are shown as below:

Strengths:

1) Eliminate noise
2) Using background subtraction to adjust the light condition so that can used in many different situations
3) Users can set the range of area, speed for objects, so it is easy to transplant to other situation which need to detect motion
4) It is possible to detect and count seeds in real-time need
5) Use multiple counters to get maximum amount so that increase the accuracy

Limitations:

1) Can not deal with the situation when seeds are very close
2) Have to test many cases for getting the best parameter for range of speed and area of seeds
3) Cannot handle situation when the objects need to be detected are very fast and intensive.

SUMMARY

The Seed Counting Application provides user-friendly interface to detect and count seeds. The performance of data retrieval is exceptional and can be maintained, even if the situation is complicated and the light condition has not been controller. The packages what we used is easily to be imported and powerful. Thus making the application very dynamic with various options to expand the existing system. What's more, the performance of the application is really excellent, the counting result can be 97.2% close to actual amount.

I have learned to detect motion in a video, track objects and count them. The OpenCV and python are powerful tools for image and video processing.

One thing that I would update is that dealing with multiple seeds when they are pretty close. What if they are consider one seed before passing by the line, and depart from each other after the line. The first solution is adding an area condition before counters plus one. If the area of the seed is bigger than one seed, then the counter should add 2. And if the area of the seed is bigger than two seeds, then the counter should add 3 and so on. But it will work not so well if the ranges of objects we need to detect are large.

The other thing I would like to consider in the future is adding some testing code in it to get the best parameters, such as range of area of objects, speed and so on. Then we don't need to test a lot of test cases to get them.

References

1. http://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/
2. Summerfield, Mark. Rapid GUI Programming with Python and Qt. Python is a very expressive language, which means that we can usually write far fewer lines of Python code than would be required for an equivalent application written in, say, C++ or Java
3. McConnell, Steve (30 November 2009). Code Complete, p. 100. ISBN 9780735636972.
4. Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises"
5. http://www.pyimagesearch.com/2015/06/15/install-opencv-3-0-and-python-2-7-on-osx/
6. http://docs.opencv.org/2.4/doc/tutorials/imgproc/opening_closing_hats/opening_closing_hats.html
7. http://docs.opencv.org/trunk/d7/d4d/tutorial_py_thresholding.html
8. http://docs.opencv.org/3.1.0/d4/d73/tutorial_py_contours_begin.html
9. http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contour_features/py_contour_features.html
10. http://www.femb.com.mx/blog/

CODE

Main python file is shown as blew:

```
########################################
########################################
####
            # import the necessary
packages    #

import cv2
import numpy as np
import imutils
import time
import Seed

            # import the necessary
packages    #

########################################
########################################
####
            # reading the video file and
parameter    #

#open the video file
cap = cv2.VideoCapture("test1.m4v")

               # reading the video file
and parameter    #
########################################
########################################
####
            # variables and input from
users    #
# should give the user of input of areaL,
areaH(this is can be adjust in different
objects, but usually test different value
then get the optimal one)
# should give the user of input of speed
of the object
# give different lines and try to count
by different part
num_seed1   = 0
num_seed2   = 0
num_seed3   = 0

#cap.set(3,400) #set width
#cap.set(4,300) #set height

areaL = 800   # by testing many time,
then know the area of one seed

areaH = 8000  # when seeds are together
will has influence

#give the user to input
#areaL = input("Please input the
areaL(min area of your
object,default: 800):")
#if the areaH is too small ,  then has
influence on when the object become
together or very close
#areaH = input("Please input the
areaLH(max area of your
object, default: 8000):")
# should give the user of input of speed
of the object,has influence on tracking
#speed = input("Please input the speed of
your object(the faster, the number should
be larger,default: 4):")
speed = 3
#should input of noise decreasing, bigger
the number, less noise
#denoise = input("Please input threshold
to decrease noise (the bigger, noise will
decrease,default: 40):")
denoise = 40
deley_time = input("Input delay_time: ")
# initialize the first frame in the video
stream
firstFrame = None
```

```
w_v = int(cap.get(3)) #get width of the
video
h_v = int(cap.get(4)) #get height of the
video
print("the width of video:",w_v)
print("the height of video:",h_v)

line_down1 = int(h_v/4)   # line1 to
detect seeds
line_down2 = int(2*h_v/4)   # line2 to
detect seeds
line_down3 = int(4*h_v/5)   # line3 to
detect seeds

mx = int(w_v/2)
my = int(h_v/2)
font = cv2.FONT_HERSHEY_SIMPLEX
seeds = []
max_seeds_num = 500
sid = 1
                # variables and input
from users   #
########################################
########################################
####
            # Background substraction
#
num_frame = 0
while(cap.isOpened()):
    num_frame +=1
#    print("This is the ", num_frame, "
th frame")
    ret, frame = cap.read()   #read a
frame
    frame_ori = frame

    try:
        gray = cv2.cvtColor(frame,
cv2.COLOR_BGR2GRAY)   #change to gray,
decease some noise
        gray = cv2.GaussianBlur(gray,
(21, 21), 0)    #blur , decease more
noise

        #if the first frame is None,
initialize it
        if firstFrame is None:
                firstFrame = gray
                continue

        # compute the absolute difference
between the current frame and first frame
        frame2 =
cv2.absdiff(firstFrame,gray)
        ret,thresh1 =
cv2.threshold(frame2,int(denoise),255,cv2
.THRESH_BINARY)  #decrese the noise
        kernel = np.ones((9,9),np.uint8)
        opening =
cv2.morphologyEx(thresh1, cv2.MORPH_OPEN,
kernel)
        closing =
cv2.morphologyEx(opening,
cv2.MORPH_CLOSE, kernel)

                    # Background
substraction    #
########################################
########################################
####
            # if there are no more frames
to show   #

    except:
        #print("EOF")
        print('The seeds crossing line_1
are ',num_seed1)
        print('The seeds crossing line_2
are ',num_seed2)
        print('The seeds crossing line_3
are ',num_seed3)
        break
```

```
            # if there are no more frames
to show   #
########################################
########################################
####
            # draw countour for every
seed    #

    _, contours0, hierarchy =
cv2.findContours(closing,cv2.RETR_EXTERNA
L,cv2.CHAIN_APPROX_NONE) # detect
contours
    # for every contour in every frame
    num_cnt = 0
    for cnt in contours0:
        num_cnt+=1
 #       print("This is the ",num_cnt,
"cnt of frame")
        area = cv2.contourArea(cnt)    #
get the area of every contour, that is
the area of the seed
 #       print("The area of this contour
is", area)
        M = cv2.moments(cnt)   #From this
moments, you can extract useful data like
area, centroid etc
        # if the area is too small or too
large, skip this
        if area< int(areaL) or area>
int(areaH) :
            continue

 # if the object fit our requirement then
draw center of contour
#get the centroid of the contours
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
        cv2.circle(frame,(cx,cy), 5,
(0,0,255), -1)# get and draw the center
of the contours
 #       print("The coords of the seeds
is cx:",cx,"cy:",cy)
        #get the rectangle of contours
         x,y,w,h = cv2.boundingRect(cnt)
#get the x,y,weight, height of the
bounding
#        print("The w ",w, "and h of the
seed is",h)
        frame =
cv2.rectangle(frame,(x,y),(x+w,y+h),(0,25
5,0),2) # get and draw the rectangle of
contours,Let (x,y) be the top-left
coordinate of the rectangle and (w,h) be
its width and height.
        areaRec = w*h
        new = True   # means that this is
a new seed

                # draw countour for every
seed    #
########################################
########################################
####
            #   TRACKING    #
        num_seed = 0

        for i in seeds:
            num_seed += 1
 #           print("This is the
",num_seed, "th seed of frame")
 #           print("The tracks is ",
i.getTracks())
            if len(i.getTracks()) >= 2:
 #               print("The length of
i.getTrackes is", len(i.getTracks()) )
                pts =
np.array(i.getTracks(), np.int32) # make
tracks become ndarray
 #               print("The pts is ",
pts)
                pts = pts.reshape((-
1,1,2))
 #               print("The updated pts
is ", pts)
```

```
            if (abs(cx - i.getX()) <= 18
and (cy-i.getY()) <=int(speed)*h*0.83 ):
#The object is near one already detected
before
                new = False
                i.updateCoords(cx,cy)
                if
(i.going_DOWN(line_down1,line_down1) ==
True):
                    num_seed1 += 1
                """
                    if area > 5000:
                        num_seed1 += 2
                    else:
                        num_seed1 += 1
                """
                if
(i.going_DOWN(line_down2,line_down2) ==
True):
                    num_seed2 += 1
                '''
                    if area > 5000:
                        num_seed2 += 2
                    else:
                        num_seed2 += 1
                '''
                if
(i.going_DOWN(line_down3,line_down3) ==
True):
#                    print("The area is
",area)
                    #print("The areaRec
is ",areaRec)
                    num_seed3 += 1
                '''
                    if area > 5000:
                        num_seed3 += 2
                    else:
                        num_seed3 += 1
                '''
                break
            if i.getState() == '1':
                    if i.getDir() ==
'down' and i.getY() > down_limit:
                        i.setDone()
            if i.timeOut():
                index = seeds.index(i)
                seeds.pop(index)
                del i
        if new == True:
            p =
Seed.MySeed(sid,cx,cy,max_seeds_num)
            seeds.append(p)
            sid +=1

               #   TRACKING    #
########################################
########################################
####
            # put text and line on video
#

    for i in seeds:
        cv2.putText(frame,
str(i.getId()),(i.getX(),i.getY()),font,0
.3,i.getRGB(),1,cv2.LINE_AA)

    str_down1 = 'DOWN1: '+ str(num_seed1)
    str_down2 = 'DOWN2: '+ str(num_seed2)
    str_down3 = 'DOWN3: '+ str(num_seed3)
 #   cv2.putText(frame,
str_down1 ,(10,line_down1-
20),font,0.5,(255,255,255),2,cv2.LINE_AA)
    cv2.putText(frame,
str_down1 ,(10,line_down1-
20),font,0.5,(255,0,0),1,cv2.LINE_AA)
 #   cv2.putText(frame,
str_down2 ,(10,line_down2-
20),font,0.5,(255,255,255),2,cv2.LINE_AA)
    cv2.putText(frame,
str_down2 ,(10,line_down2-
20),font,0.5,(255,0,0),1,cv2.LINE_AA)
#    cv2.putText(frame,
str_down3 ,(10,line_down3-
```

```
20),font,0.5,(255,255,255),2,cv2.LINE_AA)
    cv2.putText(frame,
str_down3 ,(10,line_down3-
20),font,0.5,(255,0,0),1,cv2.LINE_AA)

    frame = cv2.line(frame,
(0,line_down1), (w_v,line_down1),
(0,255,0), thickness=2) #draw a green
line at 1/4 place to detect seeds
    frame = cv2.line(frame,
(0,line_down2), (w_v,line_down2),
(0,255,0), thickness=2) #draw a green
line at 2/4 place to detect seeds
    frame = cv2.line(frame,
(0,line_down3), (w_v,line_down3),
(0,255,0), thickness=2) #draw a green
line at 3/4 place to detect seeds

              # put text and line on
video    #
########################################
########################################
####
    cv2.imshow('Seed Count
Application',frame)
    time.sleep(float(deley_time))

    #Abort and exit with 'Q' or ESC
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
 #   cv2.imshow('Seed Count
Application_frame_original',frame_ori)
#    cv2.imshow('Seed Count
Application_gray',gray)
#    cv2.imshow('Seed Count
Application_after line',frame2)
#    cv2.imshow('Seed Count
Application_after open',opening)
#    cv2.imshow('Seed Count
Application_after open_close',closing)

cap.release() #release video file
cv2.destroyAllWindows() #close all openCV
windows
```

The Seed file is shown as below:

```
from random import randint
import time

#class MySeed:
class MySeed:
    tracks = []
    def __init__(self, i, xi, yi,
max_age):
        self.i = i
        self.x = xi
        self.y = yi
        self.tracks = []
        self.R = randint(0,255)
        self.G = randint(0,255)
        self.B = randint(0,255)
        self.done = False
        self.state = '0'
        self.age = 0
        self.max_age = max_age
        self.dir = None
    def getRGB(self):
        return (self.R,self.G,self.B)
        #return (0,255,0) #use green to
show the notation of the seeds
    def getTracks(self):
        return self.tracks
    def getId(self):
        return self.i
    def getState(self):
        return self.state
    def getDir(self):
        return self.dir
    def getX(self):
```

```
            return self.x
        def getY(self):
            return self.y
        def updateCoords(self, xn, yn):
            self.age = 0

    self.tracks.append([self.x,self.y])
     #       print("The coords of the seeds
    is xn:",xn," yn: ",yn)
     #       print("The coords of the seeds
    is self.x:",self.x," self.y: ",self.y)
            self.x = xn
            self.y = yn
        def setDone(self):
            self.done = True
        def timeOut(self):
            return self.done
        def
    going_UP(self,mid_start,mid_end):
            if len(self.tracks) >= 2:
                if self.state == '0':
                    if self.tracks[-1][1] <
    mid_end and self.tracks[-2][1] >=
    mid_end: #pass by the line
                        state = '1'
                        self.dir = 'up'
                        return True
                else:
                    return False
            else:
                return False
        def
    going_DOWN(self,mid_start,mid_end):
            if len(self.tracks) >= 2:
               # print("The length of
    self.tracks is ",len(self.tracks))
                if self.state == '0':
                    if self.tracks[-1][1] >
    mid_start and self.tracks[-2][1] <=
    mid_start: #pass by the line
     #                   print("The
    self.tracks[-1][1] is ",self.tracks[-
    1][1])
     #                   print("The
    self.tracks[-2][1] is ",self.tracks[-
    2][1])
     #                   print("The
    self.tracks is ",self.tracks)
     #                   print("The
    mid_start is ",mid_start)
                        state = '1'
                        self.dir = 'down'
     #                   self.done = True
                        return True
                else:
                    return False
            else:
                return False
        def age_one(self):
            self.age += 1
            if self.age > self.max_age:
                self.done = True
            return True
```