Applications of Computer Vision in Textiles

Dr. Samrat Mukhopadhyay
Robin Malhotra(2012TT10951), Rahul Sharma(2012TT10946) *Indian Institute of Technology, Delhi*

February 27,2015

With the advent of computers and automation practices, Computer Vision has found use in various applications- from looking for cracks in building foundations to observing plant growth in greenhouses. In this Mini Project (under the able guidance of Prof. Samrat Mukhopadhyay), we are going to explore the possibilities of using this technology for Quality asssurance in the textile industry.

1 Work done till now

In the past one month, we have successfully completed the hardware project. We have procured all parts required- a raspberry pi(credit card-sized single-board computer), associated CSI camera and the requisite software tools(python3.4 and openCV, to be precise) for the project. We have assembled a cardboard-and-glue prototype for capturing images and testing. We have also familiarised ourselves with the integration of the raspberry pi camera. We have also completed a literature review of a few research papers from sources like IEEE, Hong Kong University etc. In the following sections we shall discuss the mathematical logic and algorithms that we plan to incorporate in our software application as well as the challenges we shall face.

2 Current Industry State

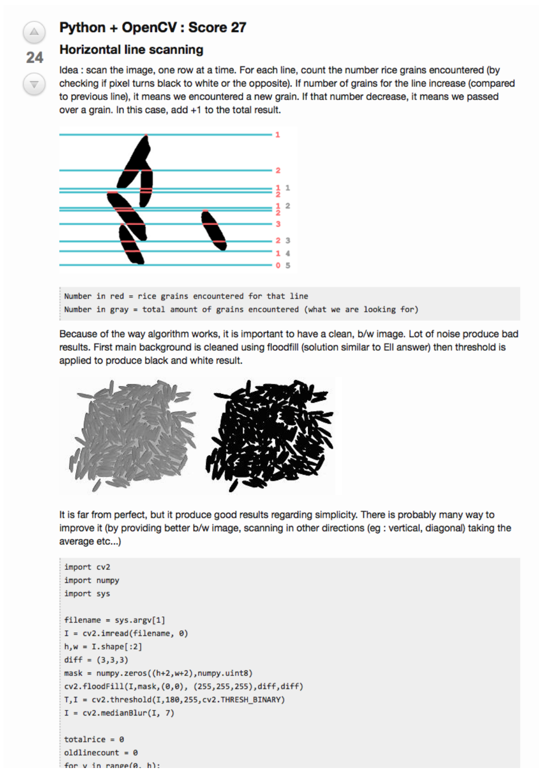
While the current industry has attempted to use Computer Vision and Image Processing for Quality Assurance, attempts have often been expensive and cumbersome. Take the example of AVT Inc, Israel that sells high quality instruments that use spectral imaging techniques to look for defects in fabrics. While these techniques may be useful and necessary in fabrics for aeronautical and structural purposes, this appears to be plain and simple overkill for garment fabrics. Also, research in this field has more or less remained stagnant after 2006, leaving the algorithms that run these processes fairly inefficient.

3 Algorithms and logic

We have reviewed several findings and research papers(our internet history from yesterday should be a testament to that). Here we will discuss the standard methods used by various researchers across the world.

- Our first challenge is to count the ends per inch(epi) and picks per inch(ppi). The algorithm behind this, ironically was not found in research labs, but on a code challange!

  1. Refer to the rice grain challenge on Stackexchange. Link
  2. Several methods were proposed by the participants, involving adaptive thresholds, deep learning, watershed algorithms and other mathematics that doesn't make sense to me.
  3. Some surprisingly simple algorithms worked deceptively well for this problem.
  4. Idea : scan the image, one row at a time. For each line, count the number rice grains encountered (by checking if pixel turns black to white or the opposite). If number of grains for the line increase (compared to previous line), it means we encountered a new grain. If that number decrease, it means we passed over a grain. In this case, add +1 to the total result.



24

Python + OpenCV : Score 27

Horizontal line scanning

Idea : scan the image, one row at a time. For each line, count the number rice grains encountered (by checking if pixel turns black to white or the opposite). If number of grains for the line increase (compared to previous line), it means we encountered a new grain. If that number decrease, it means we passed over a grain. In this case, add +1 to the total result.

```
Number in red = rice grains encountered for that line
Number in gray = total amount of grains encountered (what we are looking for)
```

Because of the way algorithm works, it is important to have a clean, b/w image. Lot of noise produce bad results. First main background is cleaned using floodfill (solution similar to Ell answer) then threshold is applied to produce black and white result.

It is far from perfect, but it produce good results regarding simplicity. There is probably many way to improve it (by providing better b/w image, scanning in other directions (eg : vertical, diagonal) taking the average etc...)

```
import cv2
import numpy
import sys

filename = sys.argv[1]
I = cv2.imread(filename, 0)
h,w = I.shape[:2]
diff = (3,3,3)
mask = numpy.zeros((h+2,w+2),numpy.uint8)
cv2.floodFill(I,mask,(0,0), (255,255,255),diff,diff)
T,I = cv2.threshold(I,180,255,cv2.THRESH_BINARY)
I = cv2.medianBlur(I, 7)

totalrice = 0
oldlinecount = 0
for y in range(0, h):
```

- One of the most popular techniques is edge detection using Gabor filters. Its impulse response is defined by a sinusoidal wave (a plane wave for 2D Gabor filters) multiplied by a Gaussian function. A set of Gabor filters with different frequencies and orientations may be helpful for extracting useful features from an image. Gabor filters have been widely used in pattern analysis applications. In the research paper by HKU, this technique is used extensively for plain, twill and denim fabrics. **Filter selection and detection Algorithm:**

  1. $g_c(x, y) = e^{-\frac{1}{2}\left[\left(\frac{x'}{\sigma_x}\right)^2+\left(\frac{y'}{\lambda\sigma_x}\right)^2\right]} \cos\left[2\pi\mu_0 x'\right]$ where x' and y' are the rotated transformed values of the x and y coordinates.μ_0 is the frequency of a sinusoidal wave along the x axis and λ is the ratio between the variances in the y and x directions.
  2. $E = \sum_{x,y}\left[IM(x, y) - w^i g_e^i(x, y)\right]^2$ where g is the gabor function in x,y
  3. By minimising the above function, we create an optimal solution from a set of 7 real filters(by changing various parameters). Generally the size of defects

is greater than a yarn, the filtering results with finer resolutions than a yarn contribute very little for segmenting defects.

4. $O_f = \sum_{i=1}^{J} a^i f^i \big(f^i = IM * g_e^i; a^i = \alpha^i / \sum_{k=1}^{J} \alpha^k\big)$ Note that the image data here is the fourier convolution.
5. Finally,the final binary segmentation result of a fabric is obtained by thresholding output from the median filtering step.

- **Defect detection using bi-level thresholding:**Another method for defect detection is bilevel thresholding. This involves scanning across the width of the grayscale image.
  1. We will first define a threshold value for intensity that will be used to contrast the defect/yarns for counting(whatever may be the case).
  2. A histogram is plotted with greyscale intensity vs pixels to produce 2 sets of pixels: G_1 consisting of pixels with grey level $T \geq T_1$ and G_2 with $T \geq T_2$. We will then find the average value μ_1 and μ_2 for these 2 regions.
  3. Then the value of T can be computed as $T = \frac{1}{2}(\mu_1 + \mu_2)$
  4. We can repeat this algorithm until our difference in successive iterations is lower than a bound T_0

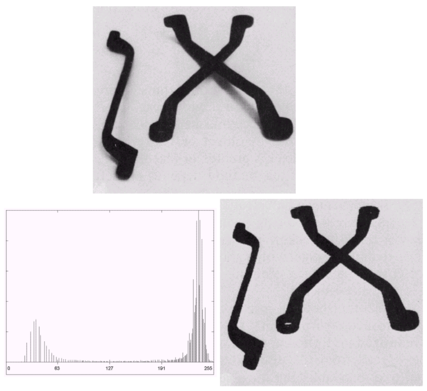


FIGURE 10.28 (a) Original image. (b) Image histogram. (c) Result of global thresholding with *T* midway between the maximum and minimum gray levels.

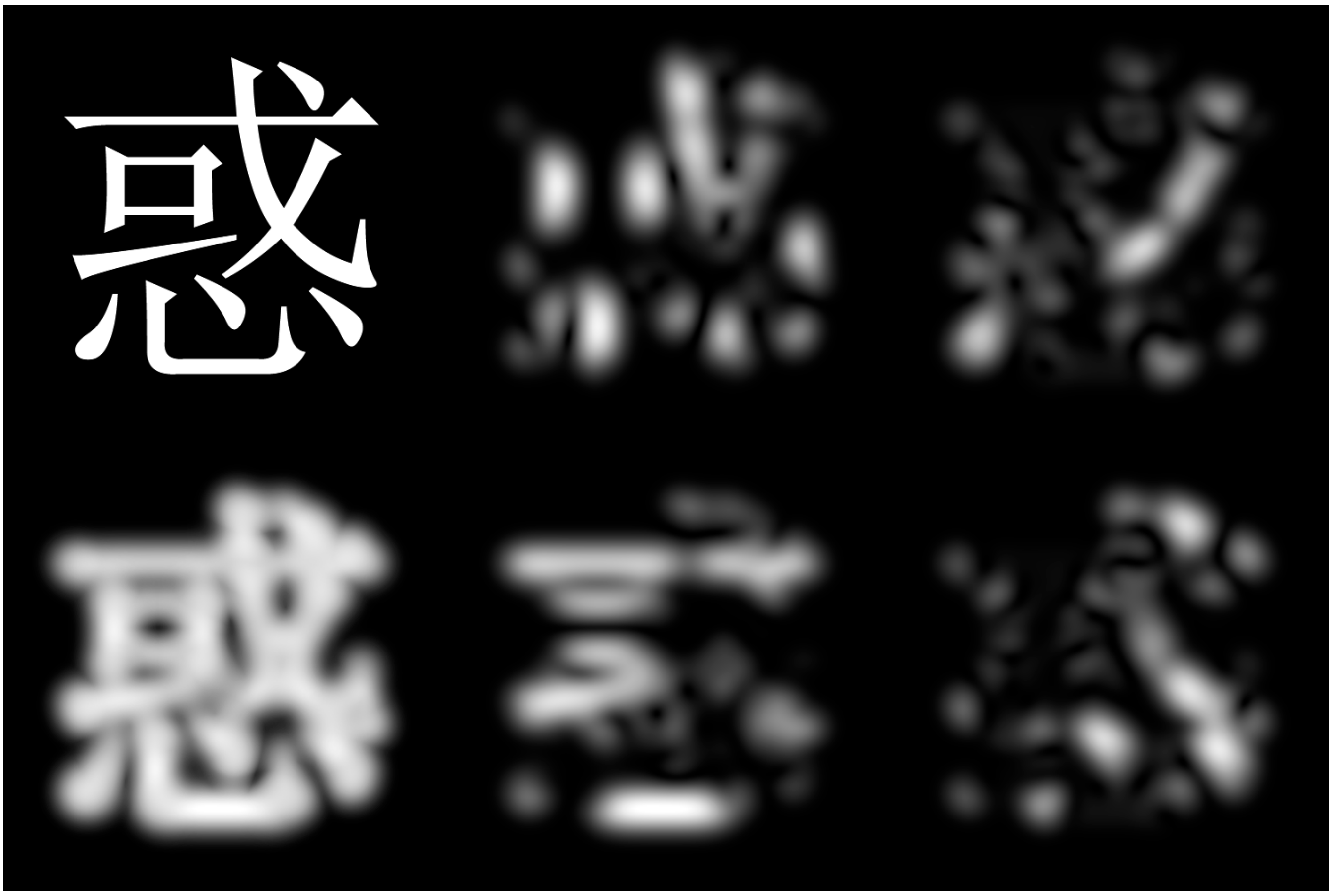
  5. Note that this was the probable idea used by our predecessors in the TTP200 course.
  6. A few factors working for us in this case is that we can accurately control the lighting and aperture etc of the camera, which should give us distinct peaks/valleys in the histogram.
  7. This method, however, will not work well with patterned fabrics.

- The last method is Fractal Dimensions. This basically tries to detect an irregular geometric object with an infinite nesting of structure at all scales. The localization accuracy of these detected defects is very poor and have high false alarm.
  1. An important (defining) property of a fractal is self-similarity, which refers to an infinite nesting of structure on all scales.
  2. Strict self- similarity refers to a characteristic of a form exhibited when a substructure resembles a superstructure in the same form.
  3. A great way to explain fractals is here
  4. The Japanese have extensively researched the possibility of using fractals to create intricate designs

  For more information, refer here

4 Plan for the future

Our plan for the rest of the semester involves implementing these algorithms in python till a usable product is achieved. We are also planning to improve our prototype with a macro lens system, as well as a mechanism for adjusting focus.



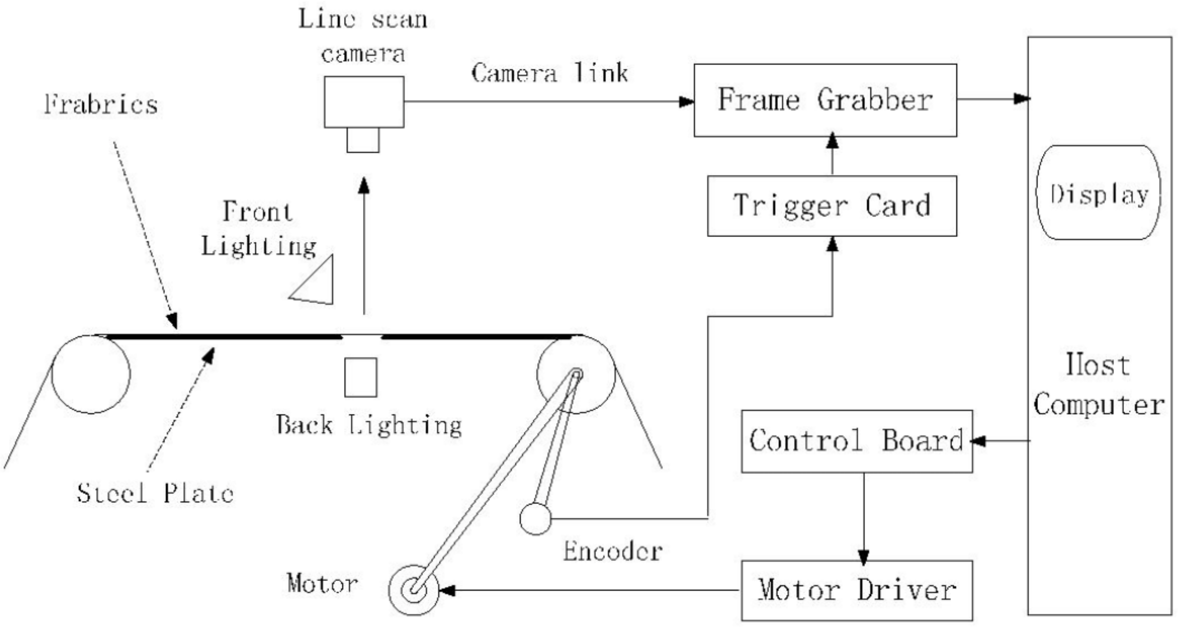
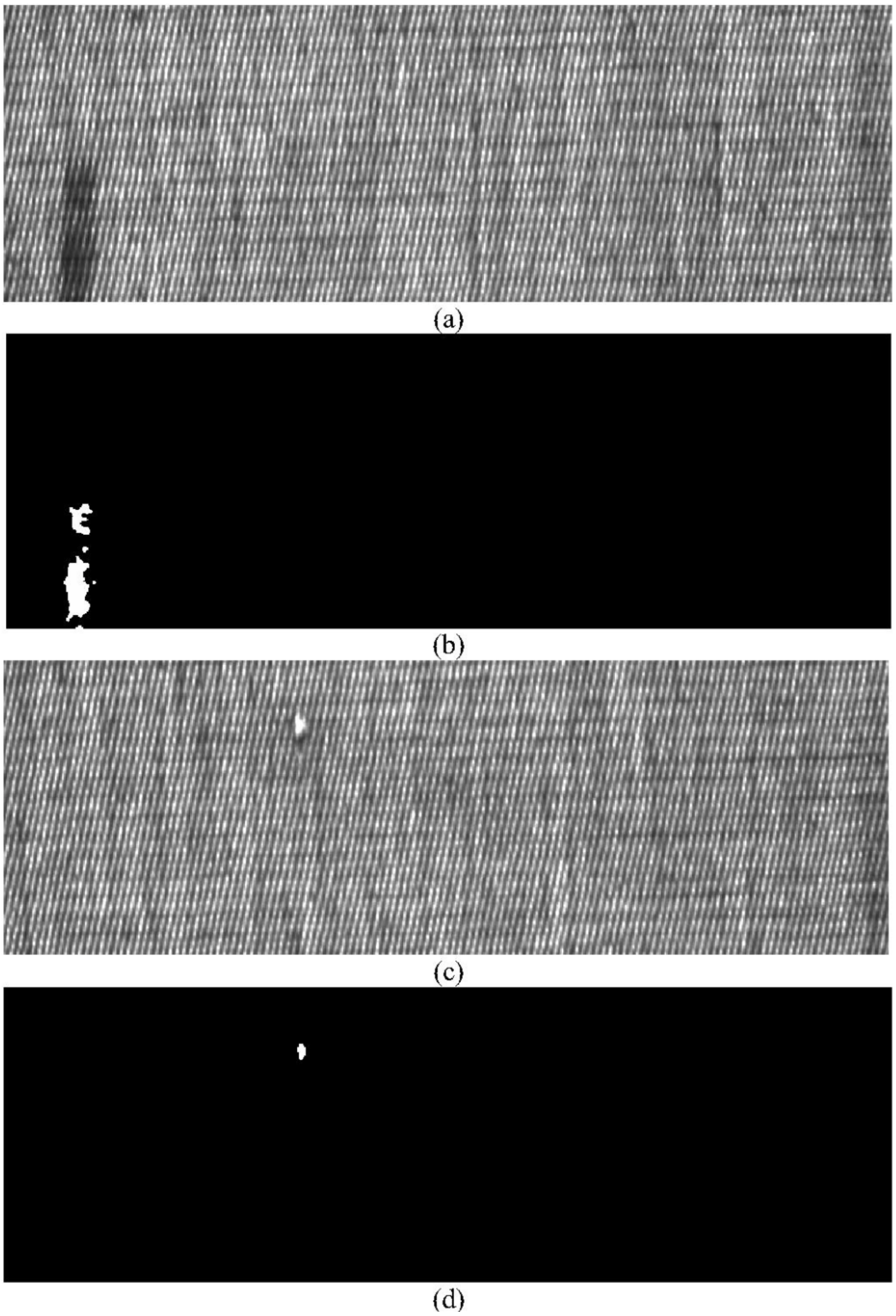


Fig. 1. Architecture of the vision inspection system.



System Performance

References

[Mak, K.L.; Peng, P.; Lau, H.Y.K] AA real-time computer vision system for detecting defects in textile fabrics *IEEE International Conference*

[www.csie.ntpu.edu.tw] $www.csie.ntpu.edu.tw/\ dalton/course/Computer_Vision/.../Chapter_4.ppt$