Counting Sarees in a Retail Shop

Mini Project Final Report
2017 - DS/NC/ESD 863 Machine Perception

Nigel Fernandez* IMT2013027

Rishabh Bhadauria† IMT2013034

Shivam Kumar‡ IMT2013045

Team 29
IIIT-Bangalore

April 19, 2017

Contents

*nigelsteven.fernandez@iiitb.org

†Rishabh.Bhadauria@iiitb.org

‡Shivam.Kumar@iiitb.org

List of Figures

1 Problem Statement and Solution Strategies

The aim of our mini project is to count the number of sarees present in an image of a retail outlet. To solve this larger problem we attempted two solution strategies. In the first solution strategy we detected shelves in the retail shop image. We ran the popular "Line Segment Detector" algorithm which detected shelves and sarees as lines of various lengths and slopes. We used this global information about the shelves (specifically the orientation or slopes of the shelves) to infer local information (orientation) of the saree. We then counted short lines (which represent sarees) of the same orientation as the shelf to obtain the count of the sarees. This solution strategy is an indirect method to count sarees by using shelf information as reference. Results of this method were 78% accurate when tested on 10 images.

The other solution strategy which we experimented with was a more direct approach. Since shelf detection requires manual parameter tuning for maximum accuracy we thought of experimenting with a more direct approach by directly detecting sarees in the image without using shelves as reference. To do so we used morphological image analysis to obtain the contours of the sarees. Due to considerable overlap between the sarees, the contours (outlines of the sarees) produced as output of the algorithm were either the contours of a single saree (most cases) or n sarees together. We applied a smart hack by finding the countour length of a single saree, finding the total contour length of all contours found in the image and performing a division of total length by single saree length to obtain the count of the sarees. Results of this method were 86% accurate when tested on 10 images.

2 Indirect Solution Strategy using Shelf Detection

In this solution strategy the information obtained of the shelves is used to obtain a count of the sarees in an indirect manner as explained before. We experimented with two different solution strategies for shelf detection. These strategies were (1) edge detection and Hough line transform based and (2) line segment detection, merging and filtering. They are stated in the order of attempt with increasing accuracy of results.

2.1 Shelf Detection with Hough Line Transform

We converted the given image to gray scale and applied `Canny` edge detector to reduce our image to edges. These edges were used as input to the probabilistic `Hough` transform. The probabilistic `Hough` line transform outputs straight lines computed from the edges. We were able to detect shelves as straight lines but these detections were superimposed by considerable noise resulting from detection of saree borders as straight lines also. The output of our implementation is shown in figure 1

In the probabilistic `Hough` transform, lines are represented in the polar coordinate space with parameter (r, θ). An equation of a line then becomes:

$$r = xcos\theta + ysin\theta$$

Each pair (r, θ) in the polar space represents a line that passes by (x_0, y_0). For a given (x_0, y_0), the family of lines passing through this point will result in a sinusoidal plot in the polar space. We iteratively perform this operation for all points in the image resulting in sinusoids for each of them. If two different sinusoids intersect in the polar plane then that will imply that both points belong to the same line. The number of such intersections can be used to detect the presence of straight lines. The more number of intersections, the more number of points on the line.

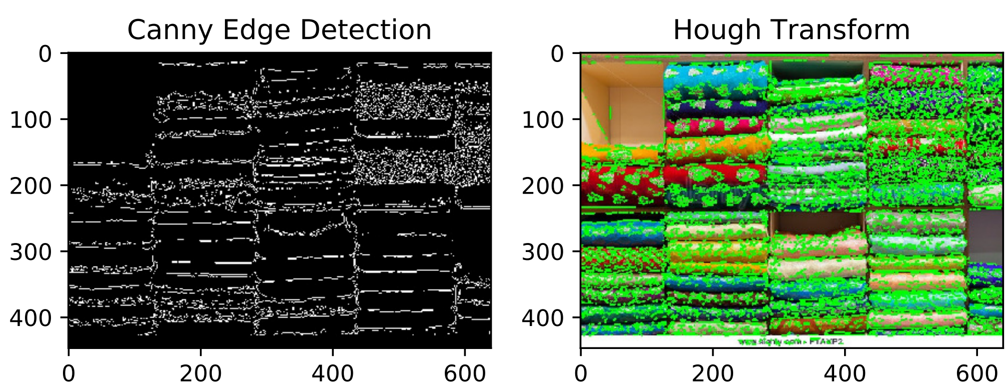

Figure 1: Edge Detection and Hough Transform

2.2 Shelf Detection using Line Segment Detection, Merging and Filtering

In our second solution strategy to detect shelves, we used the line segment detection algorithm from the paper "LSD: A Fast Line Segment Detector with a False Detection Control" [1]. This algorithm is implemented in `OpenCV`. LSD aims at detecting locally straight contours or line segments by using gradient changes in which the gray level is changing fast enough from dark to light or vice versa. To reduce the number of false alarms, LSD uses level line fields.

We used the set of line segments from the LSD algorithm as our input to our merging algorithm. To merge line segments we implemented the algorithm described in the paper "LSM: Perceptually Accurate Line Segment Merging" [2]. Since this algorithm is not available in `OpenCV`, we implmented the algorithm in Python and tuned the parameters for improved accuracy on our image data set.

The LSM algorithm attempts to solve the problem of breakage in line segments detected by LSD although the line segments are perceptually a single line. The perception of mergeability of two line segments depends on the following factors:

- Length of shorter line
- Angular difference between the two lines
- Relative spatial distance between the two lines
- Length of longer line segment

LSM uses adative thresholds for allowed spatial and angular difference computed based on lengths of the two lines. This solves the problem of one threshold not fitting all cases and provides greater flexibility. We will present the two major funtions in LSM which were implemented by us through pseudo code.

In algorithm `mergeLines` described and the algorithm `mergeTwoLines` the following notation is used:

- $\mathcal{L}$ is the set of merged lines
- $\mathcal{D}$ is the original set of line segments
- τ_s is the adaptive spatial proximity threshold

- ξ is the spatial proximity parameter
- τ_θ is the angular proximity parameter
- τ_s^* is the adaptive angular proximity threshold

Algorithm 1 Merge Line Segments

procedure MERGELINES($\mathcal{D}$)
    $\mathcal{L} \leftarrow \mathcal{D}$
    while no lines merged **do**
        $\mathcal{L} \leftarrow$ sort lines in descending order of length in $\mathcal{L}$
        for line $L1$ in $\mathcal{L}$ **do**
            $l1 \leftarrow length(L1)$
            $\tau_s \leftarrow \xi \cdot l1$
            $\mathcal{P} \leftarrow$ lines in $\mathcal{L}$ with angular difference from $L1$ within τ_θ
            $\mathcal{P} \leftarrow$ lines in $\mathcal{P}$ with spatial distance from $L1$ within τ_s
            $\mathcal{R} \leftarrow \phi$
            for line $L2$ in $\mathcal{P}$ **do**
                $M \leftarrow mergeTwoLines(L1, L2, \xi, \tau_\theta)$
                if M is not None **then**
                    $L1 \leftarrow M$
                    $\mathcal{L}(L1) \leftarrow M$
                    $\mathcal{R} \leftarrow \mathcal{R} \cup L2$
            $\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{R}$

After merging line segments to form perceptually accurate lines, we filtered the smaller length line segments as noise from the image and used the top n longest lines as the final line segments. This approach led to considerably accurate results as shown in figure 2.

2.3 Using Shelf Information to Obtain Saree Count

After obtaining shelf lines in the image, we applied a smart hack to use this information to obtain the saree count. We applied some post processing to filter out lines of noisy slopes and length and keep only long lines which represent shelves. These lines will usually have the same slope assuming that shelves will be of the same orientation in the retail outlet. We then applied a majority vote to find the average shelf slope m along with a safety error

Algorithm 2 Merge Two Line Segments

```
procedure mergeTwoLines(L1, L2, ξ, τ_θ)
    l1 ← length(L1)
    l2 ← length(L2)
    Compute closest end points and distance between them (d)
    τ_s ← ξ · l1
    Compute adaptive angular threshold τ_s^*
    Compute angle between lines θ = θ_1 − θ_2
    if d ≤ τ_s and θ ≤ τ_s^* then
        M ← line with endpoints with farthest distance between them
        Compute angle θ_M of merged line
        if |θ_1 − θ_M| < 0.5 · τ_θ then
            return M
```

range ϵ_1. This range of shelf slopes gives us the orientation of shelves in the image. We then used the unfiltered line detection output after merging. On this output we found the length of the most frequently occurring line. We assume here that sarees are the frequently occurring items placed in shelves and when detected as lines are of the same length.

This most frequent line length l is thus the length of the line detected to be a saree. We then scanned through all lines having the same length l within an error range ϵ_2 and also having the same slope as the average shelf line m within an error range ϵ_1. The number of such lines were taken as the count of sarees in the image. Results of this method were 78% accurate when tested on 10 images. This solution strategy is an indirect method to count sarees by using shelf information as reference. In the next solution strategy we explain our experiments with a more direct approach.

3 Direct Solution Strategy using Morphological Image Analysis

We took our input image and converted it to gray scale. After thresholding, we made the darker regions of the image white and vice versa. We used a rectangular box of dimensions $(2, 8)$ to find the contours of the image. The dark region between two sarees placed on a shelf was used to find the separation between the two sarees. This approach has certain limitations due

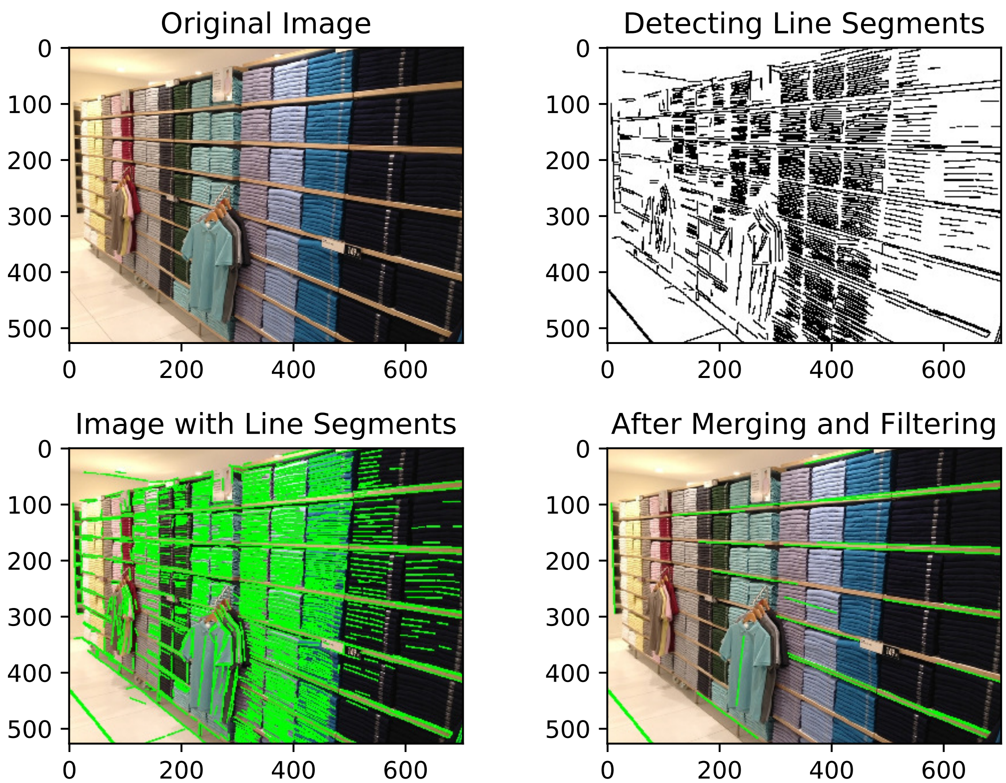

Figure 2: Detecting, Merging and Filtering Line Segments

to non existent clear marking of dark regions as separation regions between two sarees as one of the sarees itself maybe dark. Further the embroidery on the saree leads to edges and contours resulting in inaccurate detections. The results of this approach is depicted in the following figures 3 and 4.

3.1 Obtaining Saree Count

After performing morphological image analysis and finding the contours of the sarees, we performed a smart hack to obtain the saree count. If our results were highly accurate the contours formed were connected only for individual sarees with no connection between conotours of different sarees we could directly use the count of contours as the count of sarees in the image.

Since the morphological analysis output was not highly accurate we did some post processing. After observing the output we observed that most of the sarees had connected contours with no overlap with other saree contours, i.e., these were isolated detections which is the ideal case. We found the average contour length of these isolated contours which represent the average length of a single saree contour or outline denoted by l.

Since our morphological analysis output is not accurate with overlap between contours, our smart hack was to work with aggregate information with individual information as a reference. We therefore found the total length of all contours, both connected with other contours due to overlap and also the isolated accurate contours. This length L is the total length of all the sarees contours. We divided L by the average length of a single saree contour l to obtain the saree count. Results of this method were 86% accurate when tested on 10 images. Inaccuracies were present due to embroidery and other noise producers which were also detected as contours.

4 Conclusion and Future Work

In this report we present two solution strategies for counting sarees in a retail shop image. In the first strategy we performed the count indirectly by using shelf information as a reference. In the second strategy we adopted a direct approach to reduce error propogation from shelf detection. The accuracies achieved by these methods were 78% and 86% respectively.

Future work in this project could possibly involve deep learning methods. We could form a finite set of representative images of sarees of all major types (different embroideries, different cloth materials, etc). We then use these images as templates to perform a template matching (sliding convolution operation) to detect sarees. In deep learning, we could also use these finite representative images to train a convolutional neural network (CNN) to detect sarees in the image.

Bibliography

[1] Rafael Grompone von Goi et. al. "LSD: a Line Segment Detector". In: *Image Processing On Line* (2012).

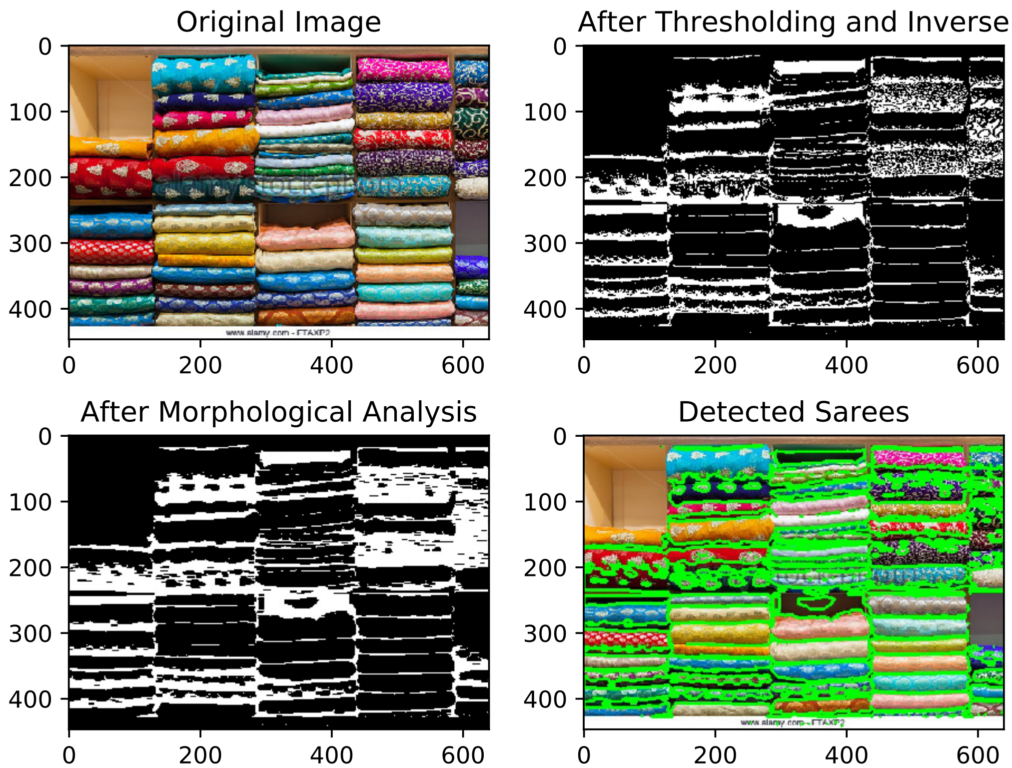

Figure 3: Morphological Analysis for Saree Detection on Image 1

[2] Naila Hamid Nazar Khan. "LSM: Perceptually Accurate Line Segment Merging". In: *Journal of Electronic Imaging* (2016).

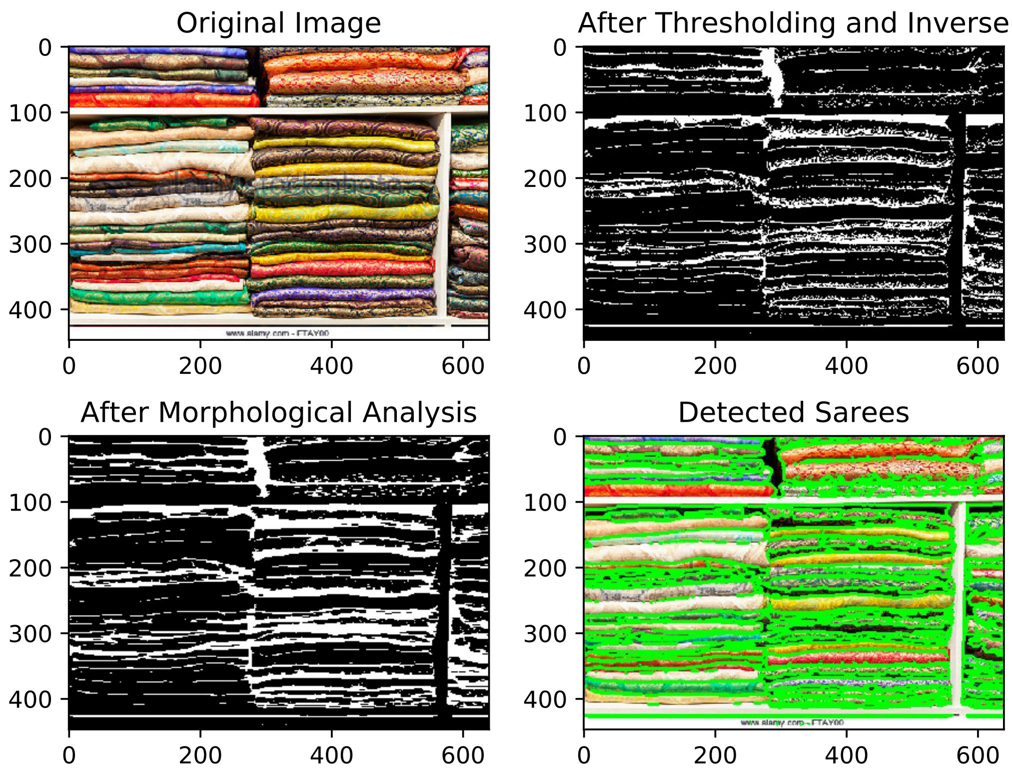

Figure 4: Morphological Analysis for Saree Detection on Image 2