# Background Subtraction using Local SVD Binary Pattern

Franci Suni Lopez
Universidad Católica San Pablo-Arequipa Peru
franci.suni@ucsp.edu.pe
Joel Oswaldo Gallegos Guillen
Universidad Católica San Pablo-Arequipa Peru
joel.gallegos@ucsp.edu.pe

Abstract—Background subtraction is a technique widely used to detect moving objects from static cameras. In the field of video surveillance, it is essential since it allows rapid discrimination when an event has arisen and studies of mobility of objects on the scene. In the last few years, many methods with different strengths and requirements have been proposed. For instance, background subtraction using local SVD binary pattern was a proposed method combining different techniques.

In this paper we study, replicate and implement the paper before mentioned.

Keywords-camera calibration,

#### I. INTRODUCTION

Background subtraction is a fundamental stage of a large number of applications in which it is necessary to detect movement, identify and / or follow objects in sequences of dynamic images. Among these applications, video surveillance can be named, such as the detection and capture of movement flow, pose estimation, mancomputer interaction or video coding based on content, among others. The process of subtraction is also called foreground extraction or foreground extraction, and consists of a series of methods that allow distinguishing between background or static zones (background), and dynamic zones that correspond to the foreground.

In order to extract the background, a representation of the scene is maintained through the digital modeling of the background. This representation must contain the necessary information for each moment to determine, for each new entry image, which areas of the same correspond effectively with the background. Since the background is not invariant in time, it is necessary to perform a dynamic modeling of the background to adapt it to the variations that may occur in it. The first time, any significant change in a region of the image with respect to the background model will be considered foreground. In general, each element of the model is updated for each certain time interval, as long as it has lost significance.

# II. PREREQUISITES TO RUN AVAILABLE SOURCE

The source code was implemented on *python*, therefore we need the python compiler installed on our computer. Initially, we installed the the prerequisites on Ubuntu 16.04, but we can not run the code because there was a problem of compatibility between the *numpy* and *maxflow* libraries. Then we install Anaconda2 version 4.4.0 on windows 7 and also install *maxflow* library manually. Finally we run the source code with the following script:

python local\_svd\_2016 -g highway\groundtruth -f highway\input -k 1700

Where -g is the Directory with ground-truth frames, -f is the directory with input frames and -k calculates answer for K-th frame and output.

### III. MATCH BETWEEN CODE AND PAPER

The algorithm proposed by the paper can be summarized by the algorithm 1 in the page 90 of the paper [1]. Then in the source code we find the function to extract the LSBP(Local SVD Binary Pattern) using the function SVD (Singular value decomposition) and the conditions to compute the background. Figure III shows the relationship between the algorithm of the paper with the parts of the source code.

#### IV. RESULTS

We use the CDnet 2012 dataset<sup>1</sup> to test the source code. From them we take the five datasets that are showed on the paper (*copyMachine*, *highway*, *overpass*, *PETS2006* and *sofa*). Then the results of our experiments is showed on Figure 2.

<sup>1</sup>http://jacarini.dinf.usherbrooke.ca/dataset2012/

```
Algorithm 1 Background Subtraction for FG/BG segmen-
tation using LSBP feature.
Initialization:

1: for each pixel of the first N frames do
2: Extract the LSBP descriptor for each pixels using E-
             quation (12)
           Push color intensities into BInt_{index}(x,y) and LS-BP features into BLSBP_{index}(x,y) as the background model
 4: Compute \overline{d}_{min}(x,y) for each pixe 5: end for
                                                                                                                                                                                   inge(W):
.clip(i, R, H+R-1)
.clip(j, R, H+R-1)
.clip(j, R, H-R-1)
.int[0,i,j] = frame[i,j]
lsbp[0,i,j] = lsbp[i,j]
xrange(i, R):
iR random.randrange
Mainloop:
6: for each pixel of newly appearing frame do
            Extract Int(x,y) and LSBP(x,y)
 9: matches \leftarrow 0
 10: index \leftarrow 0
11: for each pixel in current frame do
       while ((index \leq N) \&\& (matches < \sharp min)) do computer L1dist(Int(x,y), BInt_{index}(x,y))
13:
                 and H(LSBP(x,y),BLSBP_{index}(x,y))

if ((L1dist(x,y) < R(x,y))\&\&(H(x,y) \le
14:
                H_{LSBP})) then matches + = matches
                end if
                                                                                                                                                                                                ()
    dist[1]
    frame[i,j] = frame[i,j]
    sbp[k,i,j] = lsbp[i,j]
    dom() < Tacc[i,j]:
    n.randrange(0, S)
    condra
            index + = index
end while
 18:
            if (matches < \sharp min) then
                                                                                                                                                                                              dom.randrange(0, 5)
= np.clip(j=random.randrange(-1,2), 0, H-1), np.clip(j=random.randrange(-1,2), 0, H-1)
np.sum(np.abs(samples_int[:,i0,j0] - frame[i,j]), axis=1)
20:
                Foreground
            else
Background
22:
            end if
23:
24: end for
                                                                                                                                                                                                                         rocessing(im, unary):
= np.float32(unary)
= cv2.GaussianBlur(unary, (9, 9), 0)
                                                                                                                                                                                                                   maxflow.Graph[float]()
es = g.add_grid_nodes(unary.shape)
                                    .output is not knone:
i in xrange(f.shape[0]):
sec = time.time()
out = SVD_step(f[i])
mf = postprocessing(f[i], out)
print('Frame %d, %.3f sec. '% (i, time.time() - sec))
if i >= args.output:
    cv2.immrite('svd-frame.png', f[i] = 255)
    cv2.immrite('svd-mask.png', out = 255)
    cv2.immrite('svd-gt.png', gt[i] = 255)
    cv2.immrite('svd-mrf.png', mrf = 255)
    break
                                                                                                                                                                                                                            n xrange(im.shape[0]):
    j in xrange(im.shape[1]):
    v = nodes[i,j]
    g.add_tedge(v, -unary[i,j], -1.0*unary[i,j])
                                                                                                                                                                                                                    potts add_edge(i0, j0, i1, j1):
v0, v1 = nodes[i0,j0], nodes[i1,j1]
w = 0.1 * n,exp(-([in[[0,j0] - in[[i1,j1]])**2).sum() / 0.1)
g.add_edge(v0, v1, w, w)
                                    i in xrange(f.shape[0]):
    cv2.imshow('Frame', f[i])
    cv2.imshow('Ground-truth', gt[i])
    out = SVD_step(f[i])
    cv2.imshow('Output', out)
    cv2.imshow('Output' + MRF', postprocessing(f[i], out))
    if k = v2?.
                                                                                                                                                                                                           g.maxflow()
seg = np.float32(g.get_grid_segments(nodes))
return seg
```

Fig. 1. Match between source code and paper algorithm.

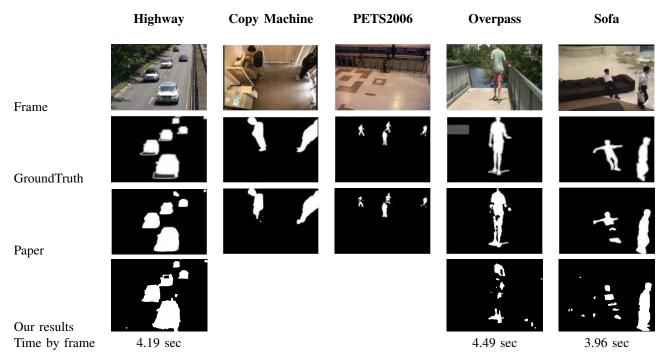


Fig. 2. Comparison between original frames, GroundTruth, results of paper and our results.

We can not compute the results of the *copyMachine* and *PETS2006* datasets because our RAM was insufficient (16 Gb). Additionally the resolution of each frame from the different datasets is different; *copyMachine* and *PETS2006* are of 720x480 and the others are of 320x240, therefore the compute task in the two firsts are more heavy because compute the *LSBP* is for each

pixel.

## REFERENCES

 L. Guo, D. Xu, and Z. Qiang. Background subtraction using local svd binary pattern. In 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 1159–1167, June 2016.