Multi Object Tracking using Low Rank Sparse Decomposition for CCTV footage

Project-II (EC47004) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Electronics and Electrical Communication Engineering

by

Sanku Yogesh

(17EC35020)

Under the supervision of

Dr. Saumik Bhattacharya

Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology Kharagpur

Spring Semester, 2020-21

May 2, 2021

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: May 2, 2021 (Sanku Yogesh)
Place: Kharagpur (17EC35020)

DEPARTMENT OF ELECTRONICS AND ELECTRICAL COMMUNICATION ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Multi Object Tracking using Low Rank Sparse Decomposition for CCTV footage** " submitted by **Sanku Yogesh** (Roll No. 17EC35020) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Electronics and Electrical Communication Engineering is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2020-21.

[illegible]

Dr. Saumik Bhattacharya
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Date: May 2, 2021

Place: Kharagpur

Abstract

Name of the student: **Sanku Yogesh** Roll No: **17EC35020**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Electronics and Electrical Communication Engineering**

Thesis title: **Multi Object Tracking using Low Rank Sparse Decomposition for CCTV footage**

Thesis supervisor: **Dr. Saumik Bhattacharya**

Month and year of thesis submission: **May 2, 2021**

This project aims to study low-rank sparse decomposition, segment objects using image subtraction, and explore how low-rank sparse decomposition can be used to track objects.

The main focus of this project is to implement a multiprocessing algorithm using Godec and image subtraction to live track objects recorded by a static camera.

Acknowledgements

I wish to express my sincere thanks to Prof. Mrityunjoy Chakraborty, Head of the Department and Prof. S.K Varshney, Faculty Advisor for providing me with all the necessary facilities for the research.

I place on record, my sincere thanks to Prof. Saumik Bhattacharya, supervisor for this project, for his suggestions and guidance.

Contents

Chapter 1

Project Description

1.1 Introduction

Object Tracking in computer vision is a famous problem with many applications like monitoring, computer-human interaction, object counting. The design of a robust tracking algorithm is challenging due to occlusion, background clutter, and changes in brightness and illumination source. Many researchers have proposed many tracking algorithms addressing these problems. Object tracking or segmentation of moving objects is based on temporal information of the frames. Using temporal information, we can separate background and foreground where the background has all the stationary things, and foreground has all the moving objects in that time. Many methods have been proposed to track objects in a recorded video by a static camera and a moving camera. We will try to segment or track objects captured by a fixed CCTV camera using fundamental image processing operations and the multiprogramming module. In the future, a high-level model can be imposed on it to get better results. We used Godec background modeling for separating foreground and background and applied it for tracking using image subtraction. This method requires some time, which is not possible in live tracking. For live tracking, we have used multiprogramming, where we simultaneously compute the background using previously captured frames and track objects in the current frame.

1.2 Related Work

Go Decomposition (GoDec) is a low-rank and sparse decomposition algorithm where it robustly and efficiently estimates the sparse composition, low-rank composition, and also the noise associated with them. There are many other algorithms like Fast principal component pursuit via alternating minimization by (6), which is the fastest LRSD algorithm. GoDec by (11), which we have implemented in our work, stood second among all the LRSD algorithms by speed and complexity. There is one more variation to GoDec known as Semi-soft GoDec, which is slower than the actual one. A Matrix completion algorithm using singular value thresholding by (3) has the highest complexity among all the LRSD algorithms. All the LRSD algorithms are nicely arranged in a library known as lrslibrary by (1). They divided them into eight different categories for Non-negative tensor factorization, Robust PCA, Subspace tracking, Low-rank Representation, Matrix completion, Non-negative Matrix Factorization, Three-Term Decomposition, and standard tensor decomposition.

Chapter 2

Godec

2.1 Low-rank and sparse decomposition

For understanding what is low-rank, let's visit SVD(singular value decomposition). Specifically, the singular value decomposition of a matrix M of size $m * n$ is a kind of factorization of the form $U\Sigma V^*$, where U is a unitary matrix of size $m * m$, Σ is a diagonal matrix of size $m * n$ with only non-negative real numbers on its diagonal, V is a unitary matrix of size $n * n$. V and U are real when M is real, and when they follow this condition, SVD can be written as $U\Sigma V^T$.
The entries on the diagonal of Σ are known as the singular values of M. We can find the rank of M by counting the number of non-singular values. The right singular vectors and left singular vectors of an M are the columns of V and U columns. The SVD can be written in many forms, and it is always possible to make the decomposition such that all the singular values are in ascending order. In that case, M can uniquely represent Σ (not necessarily V and U).
SVD can be related to eigenvalue decomposition. Any complex or

real matrix M of size $m * n$ can be decomposed using singular value decomposition, but eigenvalue decomposition is only applicable to the matrices which are diagonalizable. If we have the SVD of M, the below mentioned two relations hold:

$$M^*M = V\Sigma U^*U\Sigma V^* = V(\Sigma^*\Sigma)V^* \tag{2.1}$$

$$MM^* = U\Sigma V^*V\Sigma U^* = U(\Sigma^*\Sigma)U^* \tag{2.2}$$

The left-hand sides eigenvalue decompositions are given on the right-hand sides, and we can say:

- The columns of V (also known as right singular vectors) are equal to the eigenvectors of M^*M
- The columns of U (also known as left singular vectors) are equal to the eigenvectors of MM^*
- The non-zero elements of Σ (also known as non-zero singular values) are equal to the square roots of the non-zero eigenvalues of M^*M or MM^*

Finally, the low-rank approximation of a matrix M given by SVD is written as $M^{'} = U\Sigma^{'}V^*$ Where matrix M is approximated to another matrix, $M^{'}$ can also be said as truncated. The rank of $M^{'}$ is r which is less than the rank of the M. If you are trying to minimize the Frobenius norm of the difference between $M^{'}$ and M under the condition that the rank of $M^{'}$ is r, then SVD gives the best solution. This technique is called as Eckart-Young theorem and proved by (9). We have understood that SVD gives a good low-rank approximation from

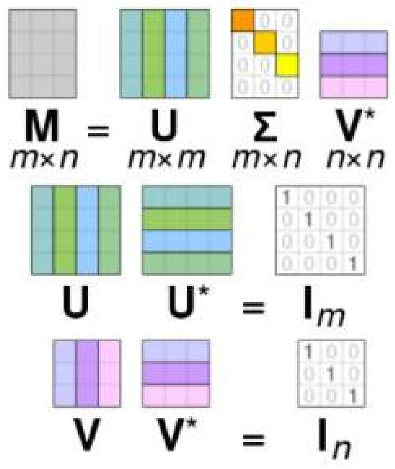


FIGURE 2.1: Visualisation of the matrix multiplications in singular value decomposition *source: https://en.wikipedia.org/wiki/Singular value decomposition*

the above discussion, but it has a high computational cost and the definition of low rank approximation with rank r is written as

$$\min_{p'} ||p - p'|| \text{ s.t. } rank(S(p')) \leq r.$$

2.2 Go decomposition

Given a matrix X, its low-rank approximation L and sparse approximated S are determined by minimizing the below objective function.

$$\min_{L,S} ||X - L - S||_F^2 \tag{2.3}$$

$$\text{s.t. } rank(L) \leq r, card(S) \leq k.$$

In GoDec rank of L and cardinality of S are predefined, which is quite different from RPCA. We will first discuss the naive GoDec algorithm where the above objective function is minimized by alternatively updating the sparse part using previous low-rank approximation and the low-rank feature using previous sparse approximation, and this can be formulated as

$$L_t = \underset{rank(L) \leq r}{argmin} ||X - S_{t-1} - L||_F^2; \tag{2.4}$$

$$S_t = \underset{card(S) \leq k}{argmin} ||X - S - L_t||_F^2; \tag{2.5}$$

The above subproblems can be solved by updating the low-rank part(L_t) by singular value thresholding (2) of $X - S_t$ and updating sparse part(S_t) via entry-wise hard thresholding of $X - L_t$, i.e..,

$$L_t = \sum_{i=1}^{r} \lambda_i U_i V_i^T, svd(X - S_{t-1}) = U \Lambda V^T;$$

$$S_t = P_\Omega(X - L_t), \Omega : |(X - L_t)_{i,j\epsilon\Omega}| \neq 0$$

$$\text{and } |(X - L_t)_{i,j\epsilon\Omega}| \geq |(X - L_t)_{i,j\epsilon\overline{\Omega}}|, |\Omega| \leq k.$$

Naive godec has high computational cost due to computation of SVD of $X - S_t$ for updating L_t. To decrease the computation cost, we use bilateral random projection, which is an entirely different method from SVD and has very low complexity and also, the error of BRP based approximation is close to the error of SVD based approximation (10). Before going into bilateral random projections, let us visit random projections once. The central idea behind random projections is given by (9); where the author describes that we can project points in a sufficiently high dimensional to a much lower dimension by preserving the relative distance among them. Using a random matrix R of size $k * d$ whose columns are unit lengths in random projection, we can project the original data of dimension d onto a subspace of dimension k where k is very much less than d. Let us say we have data X, which has n observations, and each observation is of dimension d, then using random projection, we can map this d dimensional data on to a k dimensional, and the representation of random projection is

$$X^{RP}_{k*N} = R_{k*d}X_{d*N} \tag{2.6}$$

Here R is a random matrix of size $k * d$, and X^{RP} is the random projection of X using R. The complexity of this operation is significantly less. Just make a random matrix R and do the matrix multiplication; therefore, the complexity of random projection is $O(dkN)$. If X has only c non-zero entries in its columns or says it is sparse, then the algorithm complexity can be reduced to $O(ckN)$. If Random matrix R is generated using a Gaussian distribution, it is known as Gaussian random projection.
For a dense matrix X of size $m * n$, the fast rank-r approximation is

$$L = Y_1(A_2^T Y_1)^{-1}Y_2^T \tag{2.7}$$

Where $Y_1 = XA_1$ and $Y_2 = X^T A_2$, wherein $A_1 \epsilon \mathbf{R}^{n*r}$ and $A_2 \epsilon \mathbf{R}^{m*r}$ are random matrices. The intuition behind bilateral random projections is that a random matrix has a rank equal to its highest dimension, and a matrix and its transpose have the same rank. So we take random projections on column space and row space and combine them to get the fast rank approximation. The low-rank approximation discussed above has been proposed by (4) as an approximation of a matrix X with rank-r using Y_1 and Y_2, where A_1, A_2 are two independent Gaussian random matrices. Y_1 is known as the right random projection, Y_2 is the left random projection, A_2 is the right projection matrix and, A_1 is the left projection matrix. To enhance the precision of the above

low-rank approximation, we can use Y_1 to get better A_2 and a much better A_1 using computed Y_2. Simply, after computing $Y_1 = XA_1$, we will update $A_2 = Y_1$ and then update $Y_2 = X^T A_2$ (the left random projection). After which we update $A_1 = Y_2$ and then update $Y_1 = XA_1$ (the right random projection). the more we repeat, the better low-rank approximation we get.
When singular values of X are decaying slowly, the discussed BRP based approximation may perform poorly. We use power scheme modification (7) in this situation where we compute the BRP for a different matrix $(XX^T)^q X$ having singular values decrease much faster than X. That is because the singular values of a matrix X are eigenvalues of XX^* and X^*X here singular values of $\widetilde{X}$ are eigenvalues of $(XX^*)^{2q+1}$ and $(X^*X)^{2q+1}$ so if singular values of X is λ, then singular values of $\widetilde{X}$ is λ^{2q+1}, and they share same singular vectors. The below equation shows how singular values change.

$$\widetilde{X}\widetilde{X}^* = (XX^T)^q X((XX^T)^q X)^* = (XX^T)^{2q+1} \tag{2.8}$$

and if eigenvalue of A is λ then eigenvalue of A^2 would be λ^2 so singular values of $\widetilde{X}$ is λ^{2q+1}.The bilateral random projection of $\widetilde{X}$ is

$$Y_1 = \widetilde{X}A_1, Y_2 = \widetilde{X}A_2. \tag{2.9}$$

and low-rank approximation of $\widetilde{X}$ is

$$\widetilde{L} = Y_1(A_2^T Y_1)^{-1} Y_2^T \tag{2.10}$$

We can obtain the low-rank approximation of X with rank r by calculating the QR decomposition of Y_1 and Y_2. Using QR decomposition any real matrix can be decomposition into $Q * R$ where R is an upper triangular matrix and Q is an orthogonal matrix

$$Y_1 = Q_1 R_1, Y_2 = Q_2 R_2. \tag{2.11}$$

The final low-rank approximation of X obtained using power scheme modification is

$$L = \widetilde{L}^{\frac{1}{2q+1}} = Q_1 [R_1 (A_2^T Y_1)^{-1} R_2^T]^{\frac{1}{2q+1}} Q_2^T. \tag{2.12}$$

2.2.1 Pseudo code for Fast Godec using bilateral random projections, power scheme modification.

When X is dense, there is no need for power scheme modification so, initialize q as zero in the algorithm. For a dense matrix X, algorithm one requires $r^2(m + 3n + 4r) + (4q + 4)mnr$ flops, so, compared to SVD, Godec has low computational complexity, and approximation is close enough to the SVD approximation.

Algorithm 1 Godec

```
  **Input:** $k, r, q, \epsilon, X$
  **Output:** $S, L$
  **Initialize:** $t := 0, S_0 := 0, L_0 := X$
  **while** $\epsilon < ||X - L_t - S_t||_F^2 / ||X||_F^2$ **do**
     $t \leftarrow t + 1$;
     $\widetilde{L} \leftarrow [(X - S_{t-1})(X - S_{t-1})^T]^q (X - S_{t-1})$;
     $Y_1 \leftarrow \widetilde{L} A_1, A_2 \leftarrow Y_1$;
     $Y_2 \leftarrow \widetilde{L}^T Y_1 \leftarrow Q_2 R_2, Y_1 \leftarrow \widetilde{L} Y_2 \leftarrow Q_1 R_1$;
     **if** $r > rank(A_2^T Y_1)$ **then**
        $r \leftarrow rank(A_2^T Y_1)$
        repeat from step one
     **end if**
     $L_t \leftarrow Q_1 [R_1 (A_2^T Y_1)^{-1} R_2^T]^{\frac{1}{(2q+1)}} Q_2^T$;
     $S_t \leftarrow P_\Omega (X - L_t), \Omega$ is the nonzero subset of the first k largest entries of $|X - L_t|$;
  **end while**
```

Algorithm 2 Godec

```
**Input:** $k, r, q, \epsilon, X$
**Output:** $S, L$
**Initialize:** $t := 0, S_0 := 0, L_0 := X$
while $true$ do
  for q times do
    $A_2 \leftarrow Y_1$ then $Y_1 \leftarrow XA_1$
    $A_1 \leftarrow Y_2$ then $Y_2 \leftarrow X^T A_2$
  end for
  $QR = qr(Y_2)$
  $L_{new} = (L * Q) * Q^T$
  $T = L - L_{new} + S$ (adjusting the left over L to S)
  $L = L_{new}$ ( updating L with new L)
  S= top k values in T
  Make top k values in T to zero
  if T energy is less than error then
    break
  else
    $L = L + T$ (adding left over part of S to L)
  end if
end while
```

2.2.2 Applications of Godec

Matrix Completion: They have sampled a few entries of a matrix X in each test, completed the whole matrix using Godec Algorithm1, and compared them with the original matrix. The results (11) say that it has performed well compared to the popular matrix completion method by (5).

Background modeling: We can write a video in low-rank and sparse form because all the backgrounds in video frames are related, whereas moving objects are independent. For a 200 frames video, with a resolution of $256*320$, matrix X is of size $81920*200$, where each frame is reshaped into a vector and written into columns of X. Applying Godec will separate the background and sparse part from these frames using temporal information.

Chapter 3

Tracking

Object Tracking has numerous applications, especially analyzing the traffic videos. Here we are going to track the objects in CCTV footage by using LRSD. Tracking can be done in a prerecorded video where we process the frames, draw the bounding boxes, and do the necessary analysis after the incident happened, which is not helpful in most applications. Where in live tracking, we can analyze the situation and can take action right on the situation. We discuss how we track objects in a prerecorded video and extend it to live tracking.

3.1 Object tracking in a prerecorded video :

Our basic algorithm has three steps:

- Background or foreground estimation and image subtraction
- Filtering noise and Thresholding
- Drawing Bounding boxes

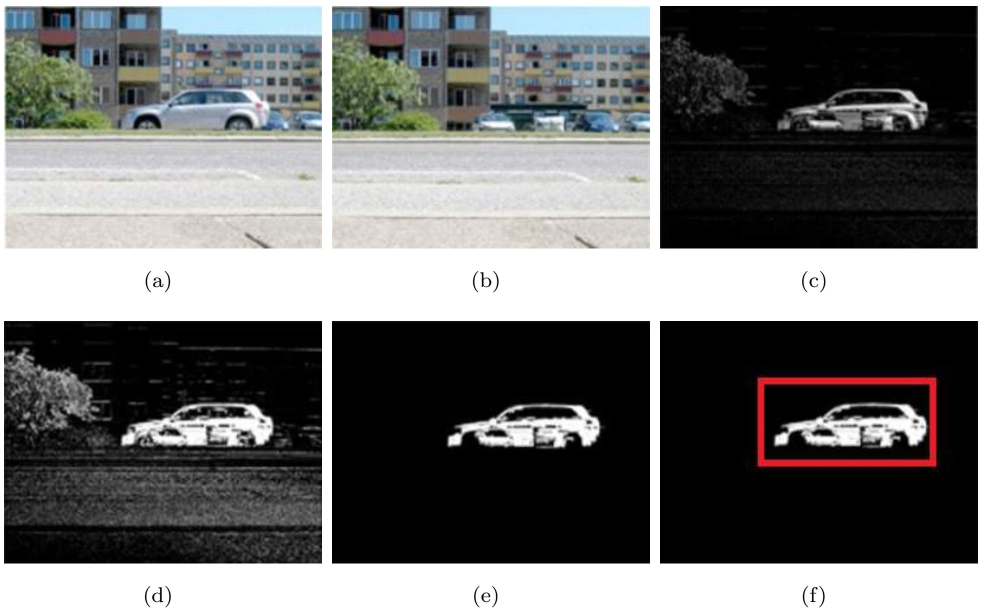


FIGURE 3.1: (a) original image; (b) background; (c) subtracted image; (d) thresholded image; (e) filtered image; (f) locating the object

Step1:

Using the background modeling, we can estimate the sparse and low-rank part of an image.Where sparse part of an image indicates the moving objects, and the low-rank part of an image is the background estimate at that frame.Let's say we have only the low-rank(background) part with us; we can estimate the sparse part(objects) of an image using image subtraction. Let us say $b(x, y)$ is the background image and $o(x, y)$ is the original image; then the resulting image $f(x, y)$ is given as

$$f(x, y) = o(x, y) - b(x, y). \tag{3.1}$$

In figure 3.1, we can observe that the car has been well separated from the background because the pixels in the car region are pretty different from pixels in the background image. However, it can also be possible that some of the pixels in the original image can be close enough to the pixels in the background image, then the pixel values in the resulting image would be close to zero. We can find this happening at the car's wheel region. So, a background that is very different from the foreground will lead to good results, for example, choosing a bright background to separate the dark foreground and vice-versa.

There is one more point to be noted while doing image subtraction. Sometimes negative values might appear in the resulting image during image subtraction. If objects moving in the foreground are darker than the background, then pixel values in the object region are less than the pixel values in the background image. So, during the subtraction $o(x, y) - b(x, y)$, where x, y are pixel coordinates in the object region leads to negative values. If these negative values are rounded off to zero, there is no way to track the object. So, the final modified formula is

$$f(x, y) = abs(o(x, y) - b(x, y)) \tag{3.2}$$

Step2:

Thresholding gives a binary image by comparing each pixel value in the resulting image to a threshold value.

$$\text{Binary image} = \begin{cases} 0 & if\ Abs(g(x, y)) < T \\ 255 & otherwise \end{cases} \tag{3.3}$$

The above thresholding produces some noise which will decrease the model performance. So, necessary filtering operations or morphological operations are needed to increase the model performance. There can be one issue using the global threshold value. In the fig 3.2, we can find two different histograms at two other places(1,2); this might be due to the randomness in the sunlight. The ideal threshold value at position2 would be 25 as it has a spread of 25, whereas at position1

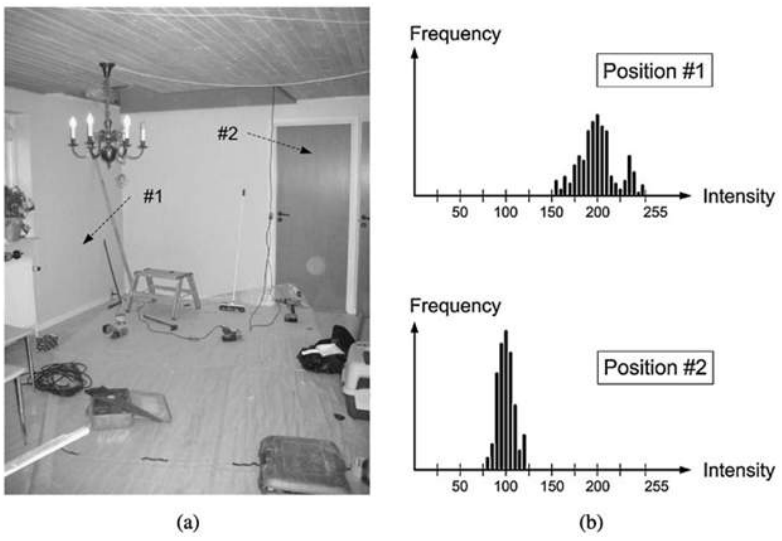


FIGURE 3.2: (a) Reference picture; (b) histograms at two different regions

would be 50, So defining the threshold value to a unique value might lead to incorrect results. It would often occur as it is very natural that different parts of the image will expose to different illuminations and lead to histograms with different variations. So, to avoid this threshold value has to be calculated differently at other places, and it is found by calculating the standard deviation at the locations.

$$\text{Binary image} = \begin{cases} 0 & if\ Abs(g(x,y)) < \beta * \sigma(x,y) \\ 255 & otherwise \end{cases} \tag{3.4}$$

β is the scaling factor, and $\sigma(x,y)$ is the standard deviation of pixel values in a patch of some arbitrary size. There is a chance for one more artifact known as the ghost artifact, where the model labels some non-object pixels in the image as object pixels that might appear like an object, so it is known as the ghost artifact.

Step3:

After binarization of the image, we need to perform connected components to identify the different objects. There can be many noise pixels in the binary image. To avoid them in the final result, we need to check the area of the connected components, and the components which have an area below some arbitrary area need to be discarded.

3.2 Live Tracking:

We have discussed how to track objects in a prerecorded video, but most of the applications need live tracking; in this section, we discuss how low-rank sparse decomposition is used to live track the objects.The discussed tracking objects in a prerecorded video algorithm requires quite some time to estimate the background. Let's say we have a prerecorded video of 30 fps and passed 200 frames to estimate the background for these frames, which might take 1.45 minutes. To obtain the tracking result of these 200 frames, we have to wait for 1.45 minutes.

The total number of frames to be processed in 1.45 minutes is nearly 3000, but we can process only 200 frames. To manage this, we can skip 14 frames after every captured frame so, totally skipped or captured frames in a second would be 30, which is the fps of the video, and captured frames in a second would be 2. After estimating the background of these captured frames, the background of the skipped frames can be estimated as the background of the nearest captured frame. This might cause some error in the model, but most of the time, this holds close to the factual background. We are calculating the background of 3000 frames in 1.45 minutes, but still, we have to wait for 1.45 minutes to get the result.

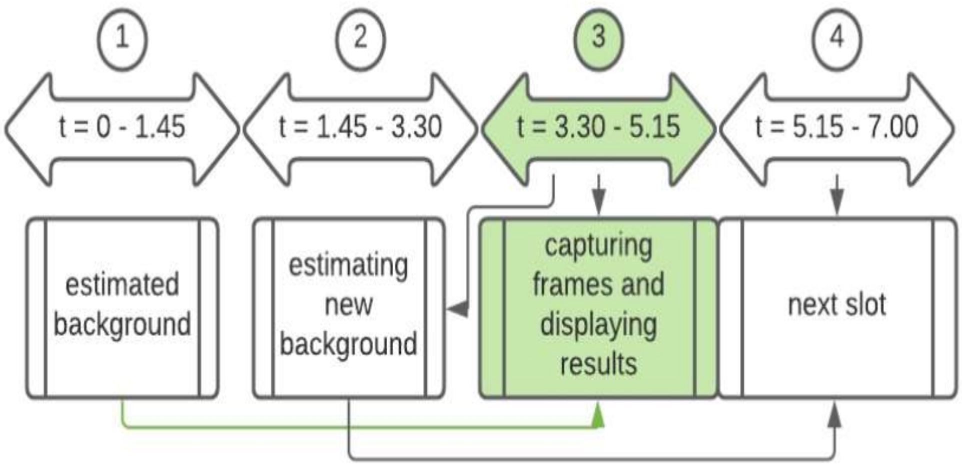


FIGURE 3.3: The shown situation is at t = 3.30 - 5.15 (3rd time slot)

A new background estimation strategy has been implemented to make this algorithm work for live tracking. Here we assume that there is no significant change in the background for 1.45*2 minutes. The previously estimated background can be taken as the background in the current 1.45 minutes. Similarly, background estimated in the recent 1.45 minutes can be taken as the background in the next 1.45 minutes. So, there have to be at least two processes running simultaneously where one process estimates the background and the other to show results.In fig 3.3, we can observe two processes processing simultaneously, where one process is estimating the new background from frames captured in the 2nd time slot. The other is capturing frames and displaying the tracking results using the background estimated from frames captured in the 1st time slot. So, at any instant, we have the estimated background, and using this estimated background, we detect the objects by image subtraction, as we discussed before. Initially, it will take 1.45 minutes to start tracking, but we would do live tracking after that.

3.2.1 Pseudo Codes and Implementation

We have implemented this with python3 on an 8Gb ram, core i5, 8th gen intel processor and implemented five different processes using the multiprogramming library, and these five processes run simultaneously. Interprocess communication is done by using queues shared among the processes. Let's name Shared queue1 as frames,used to share captured frames between processes two, three, and five. Shared queue2 as processedframes,used to share processed frames among processes three and four. Shared queue3 as bg, used to share estimated background among processes four and five.
The five processes are:

- main: It calls the remaining four child processes, and the initial background estimation will be done here.
- capturingframes: It captures frames from the webcam and keeps them in a shared queue1.
- processedframes: It does all the preprocessing required to a frame acquired from the shared queue1, like converting to grayscale, resizing, and stores them in a shared queue2.
- estimatebg: It estimates the background using the frames acquired from the shared queue2 by running the Godec algorithm and stores it in a shared queue3.
- displayframes: It calculates the sparse part using the estimated background acquired from the shared queue3 and captured frames from shared queue1 and displays the result by applying connected components.

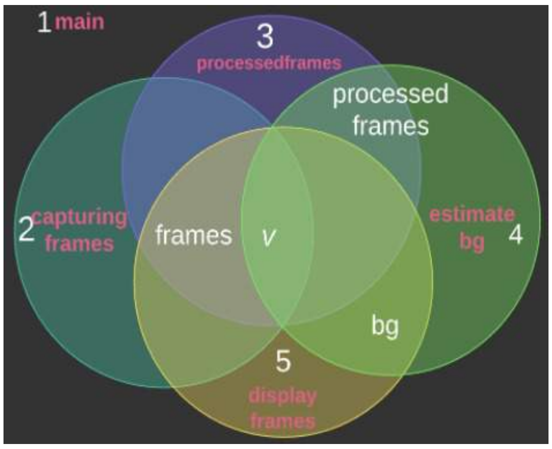


FIGURE 3.4: Processes and shared queues

Process1: main

```
Input : path to the video or webcam
Output : tracking results
initialise v, frames, processedframes, bg and their locks
for 200 frames do
  list.append(frames captured by webcam)
end for
processedframes, bg, frames ← Godec(list)
start all the four child processes and pass necessary parameters
while $v > 0$ do
  sleep for some time
end while
terminate all the four child processes
```

Process 2: capturingframes

```
while $v > 0$ do
  Frame ← captured frame
  if frameslock is open then
```

```
      if frames is full then
        remove one element from frames.
      end if
      frames.put(frame)
    end if
  end while
```

Process 3: processframes

```
  while $v > 0$ do
    if frameslock is open then
      keep all the frames in a list(frameslist)
    end if
    for each frame in frameslist do
      convert to grayscale, resize and keep in a list processframesllist
    end for
    for each frame in processframeslist do
      if processframeslock is open then
        if processframes is full then
          remove one element
        end if
        processframes.put(frame)
      end if
    end for
  end while
```

Process 4: estimatebg

```
  while $v > 0$ do
    if processedframelock is open then
      store value in a 2d array X
      $L, S \leftarrow Godec(X)$
    **end if**
    **if** bglock is open **then**
      **if** bg is not empty **then**
        remove bg
      **end if**
    **end if**
    bg.put(L)
  **end while**
```

Process 5 display

```
**while** $v > 0$ **do**
  **if** frameslock is open **then**
    **if** frames is not empty **then**
      frame ← frames.get()
    **end if**
  **end if**
  convert frame to grayscale and resize
  **if** bglock is open **then**
    bgimage ← get the frame from bg queue
  **end if**
  sparse ← frame - bgimage
  display image ← connected components(sparse)
  show the display image
  **if** exit key is pressed **then**
    v=0
  **end if**
**end while**
```

Chapter 4

Results and Discussion

4.1 Results

My supervisor Prof. Saumik Bhattacharya has collected some CCTV footage of IIT Kgp by submitting necessary documents to the institution, and we have used them in our work. Fig 4.1(a), 4.1(b) are two snapshots at the main gate outside the IIT Kharagpur. Fig 4.1(c) is a snapshot of a highway taken from (Sobral et al.).
In Fig 4.2 we have presented the results of Godec, and we can observe a person appearing in the low-rank part (fig 4.2(c)) because he is still for some time, the algorithm considered him in the background.

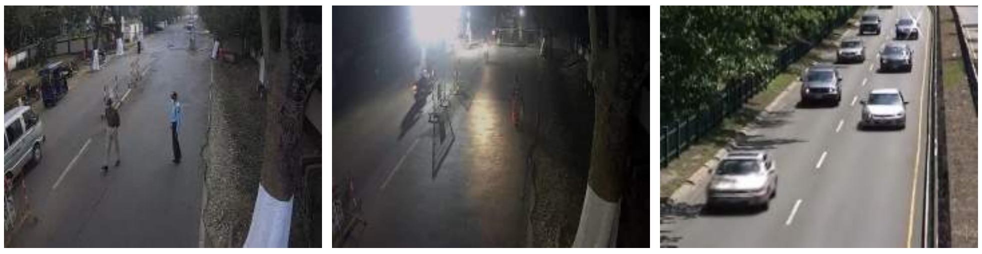
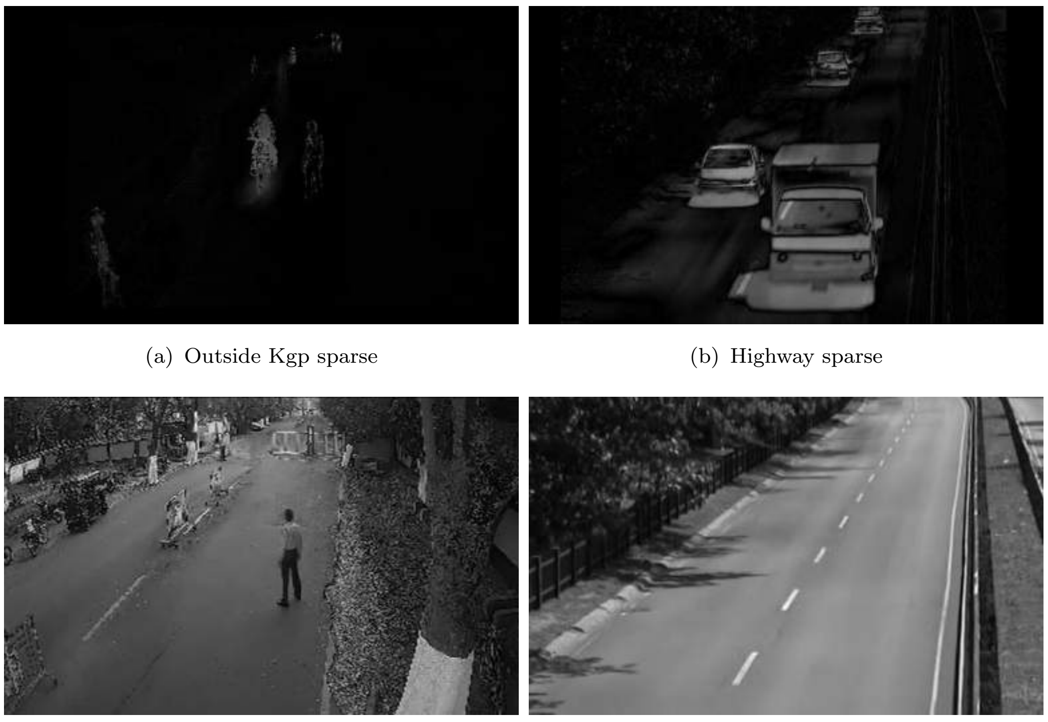

(a) Footage1 (b) Footage2 (c) Footage3

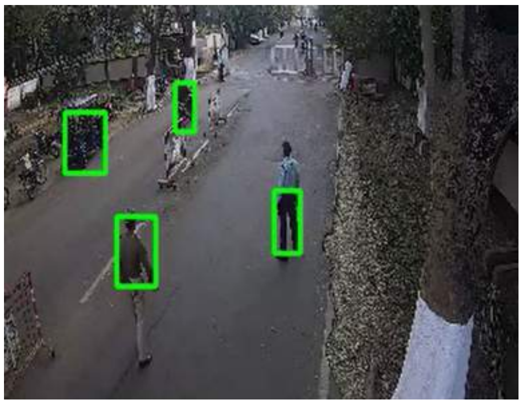
Figure 4.1: Data set


(a) Outside Kgp sparse (b) Highway sparse

(c) Outside Kgp low-rank (d) Highway low-rank

FIGURE 4.2: Low-rank and sparse parts using Godec

The tracking results are presented in fig 4.3. We can observe that only half of the person is being tracked by the algorithm because the intensity of his trouser and road are nearing the same so, the algorithm considered his trouser as noise.
Whenever an object occludes another object, they both were considered one object due to the overlap of their sparse part, which can be observed in figure 4.4(next page). We produced all the results we have observed by using only the closing operation in the morphological filtering before applying connected components. But using an opening operation after a closing operation led to the division of some objects, which we can observe in figure 4.5.Results might change depending upon the size of the structuring element we are using, and all the results are reported by 9*9 structuring element.


(a) Outside Kgp

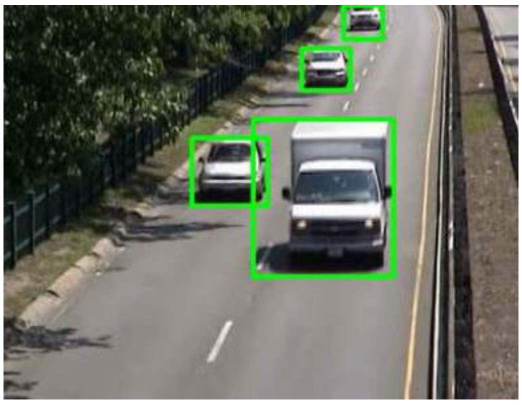
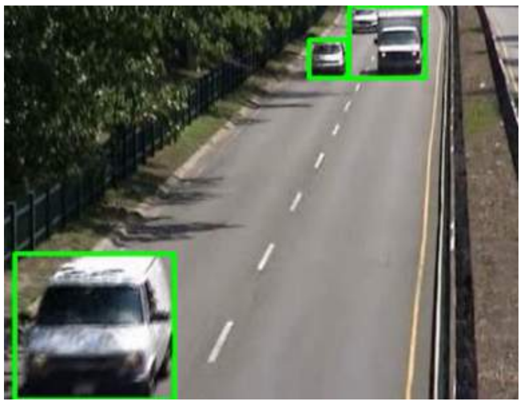

(b) Highway

Figure 4.3: Tracking results


(a) Highway scene1

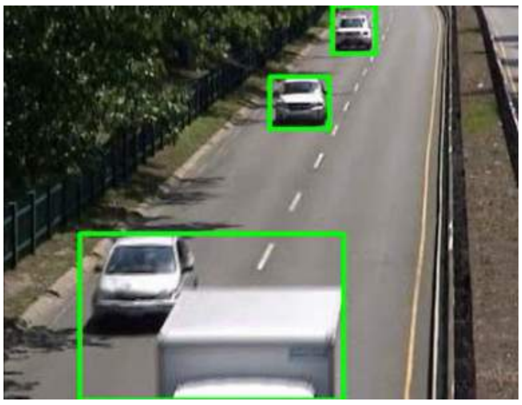

(b) Highway scene2

Figure 4.4: Segmentation of two objects into one

4.2 Future work

The operations we have performed uses basic image processing techniques like connected components and the python multiprogramming library. We have found different types of artifacts in the results section; in the future, we will try to fit our present model into the higher-level model using Deep learning or machine learning for better results. Finally, We will try to report the performance of our modified model compared with the state-of-the-art models.

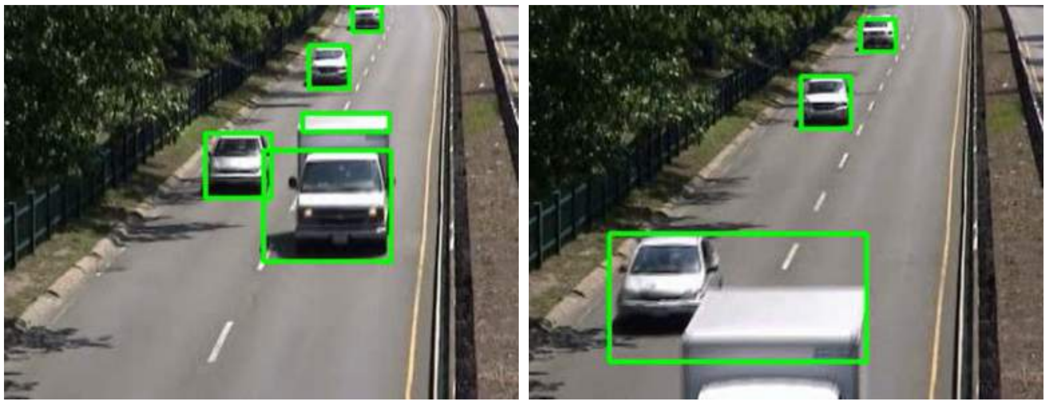

(a) Highway scene1 (b) Highway scene2

FIGURE 4.5: Single object dividing into two

4.3 Conclusion

We have discussed what does low-rank and sparse decomposition mean by visiting SVD. We then went through the Godec algorithm, where a matrix X is decomposed into L(low-rank part), S(sparse part), and G(noise) and can be represented in the form $X = L + S + G$. Naive Godec is highly computational due to the computation of SVD. Godec has been accelerated using bilateral random projects, and power scheme modification is used for dense matrices. Finally, we have proposed how to track an object in a prerecorded video using background modeling, which is one of the applications of Godec. Live tracking is difficult because a typical intel i5 processor Godec needs 1.45 minutes to estimate the backgrounds of 200 frames. Multiprocessing has been used to skip some frames and simultaneously estimate the background of captured frames and track objects in the current frame. We then discussed the pseudo-codes of all the processes. Finally, we have reported the results, types of artifacts, and reasons for them and future work.

Bibliography

[1] Bouwmans, T., Sobral, A., Javed, S., Jung, S. K., and Zahzah, E.-h. (2015). Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset.

[2] Bredies, K. and Lorenz, D. (2008). Iterated hard shrinkage for minimization problems with sparsity constraints. *SIAM J. Scientific Computing*, 30:657–683.

[3] Cai, J.-F., Candès, E., and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20:1956–1982.

[4] Fazel, M., Candes, E., Recht, B., and Parrilo, P. (2008). Compressed sensing and robust recovery of low rank matrices. In *2008 42nd Asilomar Conference on Signals, Systems and Computers*, pages 1043–1047.

[5] Keshavan, R. and Oh, S. (2009). A gradient descent algorithm on the grassman manifold for matrix completion. 910.

[6] Rodríguez, P. and Wohlberg, B. (2013). Fast principal component pursuit via alternating minimization. In *2013 IEEE International Conference on Image Processing*, pages 69–73.

[7] Roweis, S. (1998). Em algorithms for pca and spca. In *in Advances in Neural Information Processing Systems*, pages 626–632. MIT Press.

[Sobral et al.] Sobral, A., Bouwmans, T., and Zahzah, E.-h. Lrslibrary: Low-rank and sparse tools for background modeling and subtraction in videos. In *Robust Low-Rank and Sparse Matrix Decomposition: Applications in Image and Video Processing*. CRC Press, Taylor and Francis Group.

[9] Stewart, G. W. (1992). On the early history of the singular value decomposition.

[10] Zhou, T. and Tao, D. (2011a). Bilateral random projections.

[11] Zhou, T. and Tao, D. (2011b). Godec: Randomized lowrank sparse matrix decomposition in noisy case. pages 33–40.