**Software Team Task for AGV**

* Priyakant Gautam
* There are six tasks given a form which I have attempted task 1
* Which is further classified into two sub-tasks

## **Subtask 1**

* Optical flow (sparse optical flow)

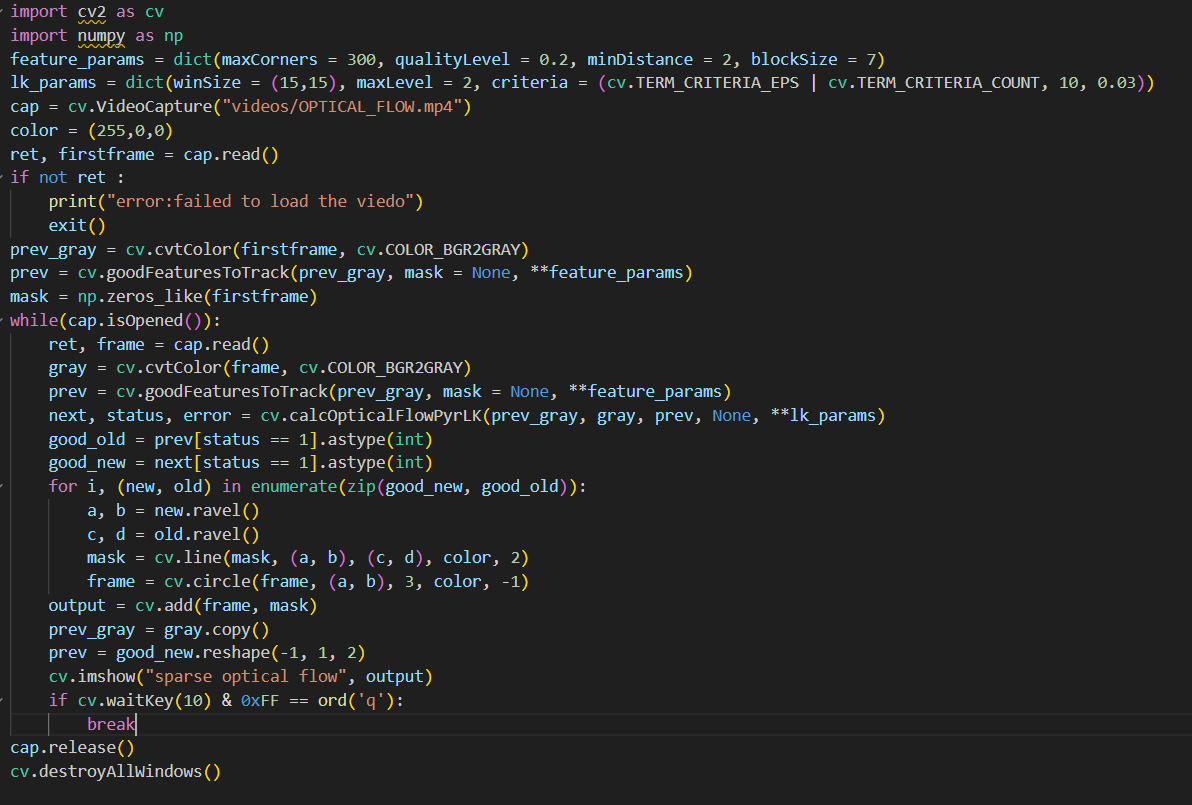
## The motto of this subtask-:

* In this task, we have to detect the optical flow

## Content of task

* We have to detect spare optical flow using the Lucas Kanade method the basic cv2 function and also the numpy module.
* Firstly, I have to go through the given research paper for which I have a basic idea of optical flow and all other methods that will be used in this task by this research paper I have to figure out the code to find spare optical flow this code is given the attached research paper.

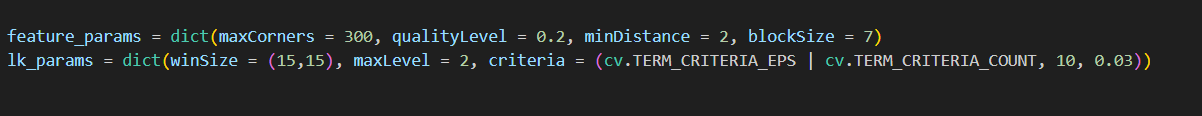
### The first demo code -:



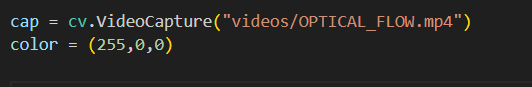
### Now the breakdown of this code-:

### 

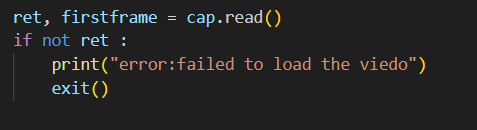
* First, we import cv2 as cv and numpy as np.

1. 

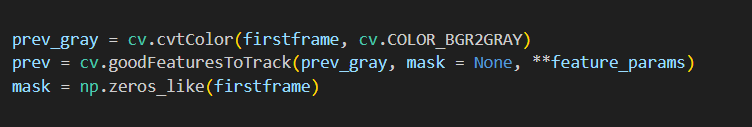
* We create some variables to use later in the code. These variables contain some features required by the function in which we will use them.

1. 

* In this, we capture the video frame by frame and the address of the video is("video/optical\_flow.mp4")
* The color is also a variable that we will use further and (255,0,0) this color code is red.

1. 

* By this line of code, we just want to clarify whether that video is successfully ret or not, and if not then it will exit the code and print the error: failed to load the video.

1. 

* Now in this-:
* Prev\_grey int this we make the video in gray color to reduce its complicity and easy to detect the required pixel
* Prev in this we track the corner of as mentioned condition by feature\_params it is based on the Shi-Tomashi corner detection method
* now the parameter which is used is:
* 1.prev\_gray-:the grayscale image to detect corner

2. Mask-: no specific region is selected

3. max corner-:maximum number of corners to detect

4. Qualiytylevel-:the minimum level of quality to detect the corner

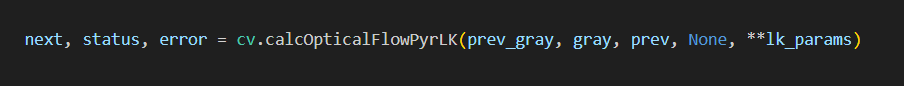
5. mindistance-: Minimum distance between detected features.

6. blockSize-:Size of the area for computing corners.

* Np.zeroes\_like makes an array of zeroes with the same shape and size as the first frame.
* This mask is later used to draw **optical flow paths** by overlaying lines on the original video frame.

1. 

* Cap.isOpened() check the video is successfully opened
* The while loop ensures that the frames are only considered while the video is open

1. 

* This function computes sparse optical flow btw two consecutive frames using the Lucas-Kanade method.
* It tracks the movement of detected feature points from the prev\_gray frame to the gray frame.
* Parameter-:

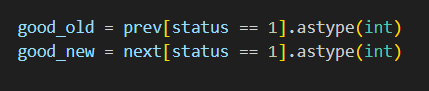
1.prev\_gray=previous grayscale image

2. Current grayscale image

3.prev=the set of points to detect

4.lk params

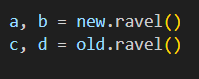
* It returns the next frame and throws an error.

1. 

* This ensures that only valid tracked point is used for visualization
* Astype(int) just converts the floating point num to the integer num
* Good\_old take valid prev tracked point
* Good\_new take valid next tracked point

1. 

* Zip(good\_new,good\_old)-: Pairs up the new and old positions of each successfully tracked feature.
* Enumerate-:add the index(i) which is helpful for the loop
* Zip()-:pair like[(new1,old1), (new2,old2) …….]

1. 

* .ravel()-:convert the multi-dimensional array into 1d array
* It converts the new and old arrays which are 2d arrays into separate x and y coordinates
* New.ravel() converts [x\_new,y\_new]into two separate value a=x\_new,b=y\_new
* Similarly for old.ravel()
* The (x,y) coordinates are extracted as individual values to be used in drawing the function.

1. 

* It is used to draw a line from(c,d) to (a,b) of the given color the thickness of the line is 2

1. 

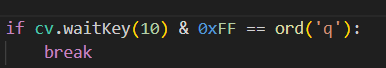
* For each tracked feature, a small red dot is drawn at its new position.
* This visually represents the tracked point on the video.

1. 

* Cv.add() combines two images by adding their pixel value
* Cv.add(frame, mask)merges them, so the motion lines appear on top of the frames in the video.

1. 

* Display the video in a window.
* The name of the video is Sparse Optical Flow.

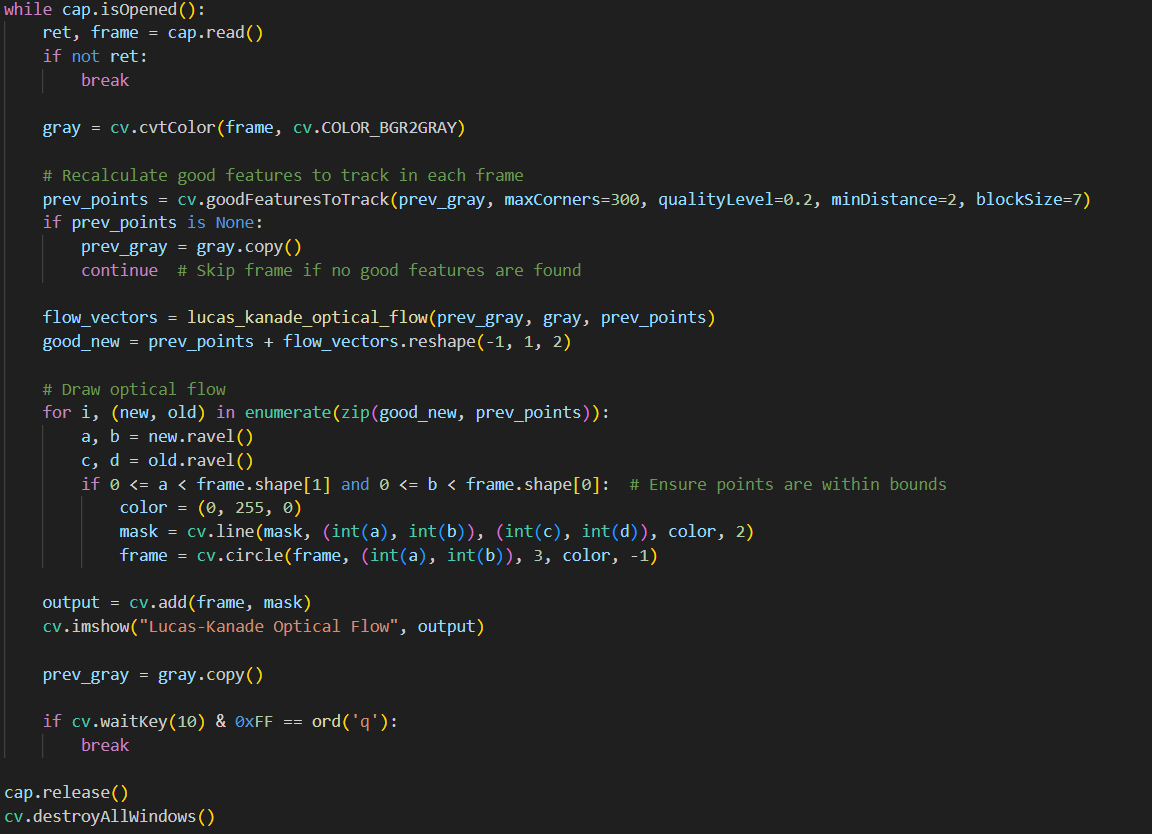
1. 

* It just commands when to break the loop and by which key we can break and finish the task.

## But in the above code, we have used the cv.CalcOpticalFlowPyrLK() but according to the task we have to implement the Lucas Kanade method manually so

### The final code -: first half

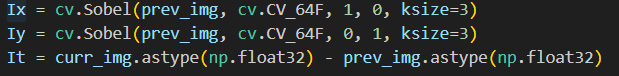
The other half is on another page-:



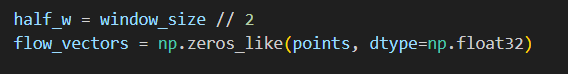
### Key point-:

* Now use Lucas Kanade optical flow manually
* We make a function to measure Lucas Kanade optical flow which is doing the same work as cv2.CalcOpticalFlowPyLk() function.

## Breakdown of code-:

1. 

* In this line code, we compute gradients
* Ix=compute the gradient of prev\_img in the x direction.
* Iy=compute the gradient in the y direction.
* It=compute the difference btw prev image and the current image.

1. 

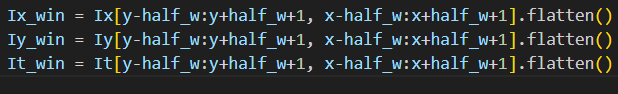
* Half\_w=This calculates half the window size.
* Flow\_vector=creates an array to store the optical flow of each point.

1. 

* The function iterates over each feature point detected in the previous frame.

1. 

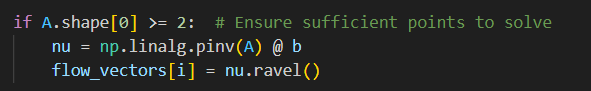
* This condition checks if the window around the point extend outside the image boundaries if yes it skip the point processing.

1. 

* These lines extract a small window.
* Then flattened into a 1D array for easier matrix operation.
* Ix\_win=gradient in the x-direction
* Iy\_win=gradient in the y-direction
* It\_win=change in intensity over time
* This provides the required matrix.

1. 

* This step constructs the Ax=b system.
* A is a matrix where a row represents [Ix, Iy].
* B is a column vector representing a negative temporal gradient.
* This equation helps us to determine how the feature points are moving.

1. 

* A.shape[0] is the number of rows we need at least two unknown point
* Np.linalg.pinv(A) solve using the pseudoinverse method
* Flow vector stores the value in the array

## The remaining code is the same as the first the only change is we call the function now.

# Subtask 2

* I am facing a problem in loading the iam2thor file in docker and am not able to complete it.