Media Streaming with IBM Cloud Video Streaming
Phase 4 submission

Introduction:

Media streaming is an integral part of the modern digital landscape, revolutionizing the way we consume and share content. IBM Cloud Video Streaming, a powerful and versatile platform offered by IBM, has emerged as a frontrunner in this domain. As the demand for high-quality video content continues to grow, IBM Cloud Video Streaming provides a comprehensive solution for individuals and businesses alike, allowing them to deliver, manage, and optimize their video content efficiently and effectively.

IBM Cloud Video Streaming is designed to meet the diverse needs of its users. Whether you are a content creator, an enterprise looking to engage with your audience, or an educational institution seeking to reach a broader student base, this platform offers a tailored solution. It enables users to seamlessly stream live events, webinars, on-demand videos, and more. With IBM Cloud Video Streaming, you can engage your audience in real-time, create interactive live experiences, and archive content for future viewing.

One of the standout features of IBM Cloud Video Streaming is its robust and scalable infrastructure. It leverages IBM's extensive global network and data centers to ensure high-quality streaming, even in regions with limited connectivity. This ensures that your content is accessible to audiences worldwide, offering a seamless and buffer-free viewing experience. Additionally, the platform provides advanced analytics and monitoring tools, allowing users to gain valuable insights into their viewers' behavior, thus optimizing content delivery and engagement strategies.

Furthermore, security is a top priority in today's digital landscape. IBM Cloud Video Streaming prioritizes the protection of your content and data. It offers various security features, including password protection, encryption, and secure embed options to safeguard your video streams. In an era where data privacy is paramount, IBM's commitment to security ensures that your content remains protected from unauthorized access or breaches.

In conclusion, IBM Cloud Video Streaming empowers individuals, businesses, and organizations to harness the power of media streaming in the digital age. With its user-friendly interface, scalability, and commitment to security, it provides a comprehensive solution for all your video streaming needs. Whether you're a content creator looking to reach a global audience or an enterprise seeking to engage customers, IBM Cloud Video Streaming is a robust platform that offers the tools and capabilities to make your content streaming experience a success.

Pre-processing

In this notebook, we will pre-process the frames. For better visualization, we will just capture 2 frames and visualize all the steps. The steps are:

1. Capture 2 consecutive frames.
2. Find the difference between the frames to capture the motion.
3. Use Gaussian blur, thresholding, dilation, and erosion to pre-process the frames.
4. Image segmentation using contours. Extract the vehicles during this method.
5. Convert contours to hulls.

Development

In this phase, you will focus on integrating video streaming services, enabling on-demand playback, and allowing users to upload their movies and videos to your platform. You'll also integrate IBM Cloud Video Streaming services for high-quality video playback. Here's a step-by-step guide on how to proceed:

1. Integrate Video Streaming Services:
- Choose a video streaming service provider to integrate into your platform. Popular options include Vimeo, YouTube, or custom solutions using platforms like Amazon Web Services (AWS) or Microsoft Azure. The choice depends on your specific requirements and budget.

2. Implement On-Demand Playback:
- Ensure your platform has a user-friendly interface where users can browse and select videos to watch.
- Implement a video player component that allows for on-demand video playback. You can use HTML5 video players like Video.js, Plyr, or open-source media players like ExoPlayer for Android apps.

3. User Video Upload Functionality:
- Develop a user-friendly upload feature that allows users to submit their movies and videos to the platform. This feature may include the following steps:
  a. User authentication and account management: Allow users to create accounts and log in.
  b. Video upload form: Create a form where users can provide video titles, descriptions, and upload video files.
  c. Video file handling: Implement a server-side script to handle video uploads, validate file formats, and store videos on your server or cloud storage (e.g., AWS S3, Google Cloud Storage).
  d. Video processing: Transcode or process the uploaded videos into different quality formats suitable for streaming.

4. Integrate IBM Cloud Video Streaming Services:
- Sign up for IBM Cloud Video Streaming services and obtain the necessary API keys and access credentials.
- Implement the IBM Cloud Video SDK or API into your platform to enable smooth and high-quality video playback. Ensure you configure the settings for adaptive streaming, DRM protection, and analytics as needed.
- Leverage the IBM Cloud Video services for hosting and streaming videos on your platform. This can help you ensure a reliable and scalable infrastructure for video delivery.

5. User Engagement Features:
- Implement features like video categorization, tagging, and search functionality to enhance user engagement and discoverability.

- Allow users to like, comment, and share videos, creating a social aspect to your platform.

6. Testing and Quality Assurance:
- Thoroughly test the video streaming, upload, and playback functionalities to ensure they work smoothly on various devices and browsers.
- Pay special attention to video quality, load times, and responsiveness.

7. User Documentation and Support:
- Create user documentation to guide users on how to upload videos and use the platform effectively.
- Provide user support, such as a contact form or support email, for any issues users may encounter.

8. Legal Considerations:
- Ensure that you have the necessary permissions for hosting and streaming user-uploaded content. This includes addressing copyright concerns and implementing content moderation if necessary.

9. Scalability and Performance:
- Plan for scalability by using cloud-based resources, CDNs, and load balancing to handle increased traffic and ensure low latency.

10. Security:
- Implement robust security measures to protect user data and prevent unauthorized access or video content theft. This includes encryption, secure authentication, and regular security audits.

Once you've completed these steps, you'll have a platform that integrates video streaming services, allows on-demand video playback, and enables users to upload their videos. Make sure to continuously monitor and update your platform to provide a seamless and enjoyable video streaming experience for your users.

```
# Run these if OpenCV doesn't load

import sys

#
sys.path.append('/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/cv2/')
```

First, we import the necessary libraries

```
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt
%matplotlib inline
```

Defining the variables

Next, we define variables that will be used through the duration of the

code # Here, we define some colours

```
SCALAR_BLACK = (0.0,0.0,0.0)
SCALAR_WHITE = (255.0,255.0,255.0)
SCALAR_YELLOW = (0.0,255.0,255.0)
SCALAR_GREEN = (0.0,255.0,0.0)
SCALAR_RED = (0.0,0.0,255.0)
SCALAR_CYAN = (255.0,255.0,0.0)
```

Function to draw the image

```
# function to plot n images using subplots

def plot_image(images, captions=None, cmap=None ): f,
    axes = plt.subplots(1, len(images), sharey=True)
    f.set_figwidth(15)
    for ax,image,caption in zip(axes, images, captions):
        ax.imshow(image, cmap)
        ax.set_title(caption)
```

Capturing movement in video

Two consecutive frames are required to capture the movement. If there is movement in vehicle, there will be small change in pixel value in the current frame compared to the previous frame. The change implies movement. Let's capture the first 2 frames now.

```
SHOW_DEBUG_STEPS = True
```

 # Reading video

```
cap = cv2.VideoCapture('../input/video-analysis/AundhBridge.mp4')
# if video is not present, show error
if not(cap.isOpened()): print("Error
    reading file")
```

 # Check if you are able to capture the video

```
ret, fFrame = cap.read()

# Capturing 2 consecutive frames and making a copy of those frame. Perform all operations on the copy frame.
ret, fFrame1 = cap.read()
ret, fFrame2 = cap.read()
img1 = fFrame1.copy()
img2 = fFrame2.copy()
```

```
if(SHOW_DEBUG_STEPS):
    print ('img1 height' + str(img1.shape[0]))
    print ('img1 width' + str(img1.shape[1]))
    print ('img2 height' + str(img2.shape[0]))
    print ('img2 width' + str(img2.shape[1]))
```

 # Convert the colour images to greyscale in order to enable fast processing

```
img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

plot_image([img1, img2], cmap='gray', captions=["First frame", "Second frame"])
```

#plotting

```
img1 height720
img1
width1280
img2 height720
img2
width1280
```

Adding gaussion blur for smoothening

 # Add some Gaussian Blur

```
img1 = cv2.GaussianBlur(img1,(5,5),0)
img2 = cv2.GaussianBlur(img2,(5,5),0)
```

#plotting

```
plot_image([img1, img2], cmap='gray', captions=["GaussianBlur first frame", "GaussianBlur
second frame"])
```

Find the movement in the video

If the vehicle is moving, there will be a **slight change** in pixel value in the next frame compared to the previous frame. We then threshold the image. This will be useful further for preprocessing. Pixel value below 30 will be set as 0(black) and above as 255(white)
Thresholding: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

This imgDiff variable is the difference between consecutive frames, which is equivalent to detecting
Movement

```
imgDiff = cv2.absdiff(img1, img2)
```

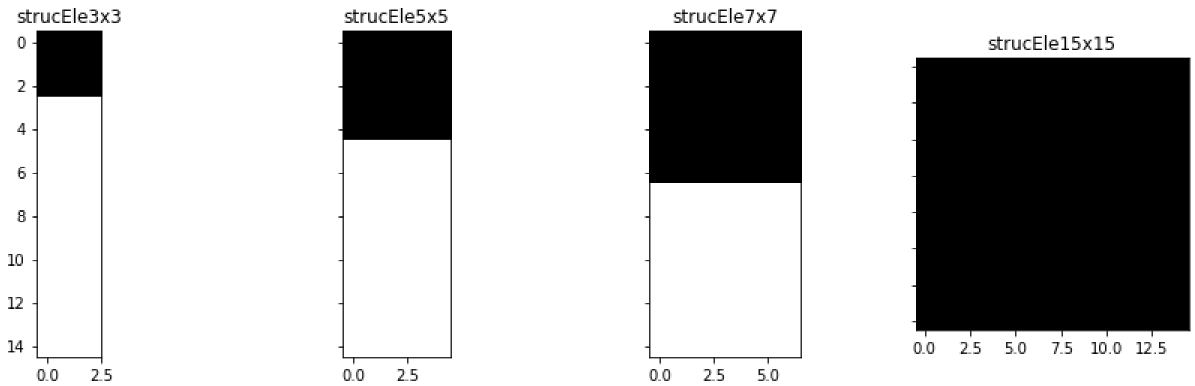
Thresholding the image that is obtained after taking difference. Pixel value below 30 will be set as 0(black) and above as 255(white*)*

```
ret,imgThresh = cv2.threshold(imgDiff,30.0,255.0,cv2.THRESH_BINARY) ht
= np.size(imgThresh,0)
wd = np.size(imgThresh,1)
plot_image([imgDiff, imgThresh], cmap='gray', captions = ["Difference between 2 frames", "Difference between 2 frames after threshold"])
```

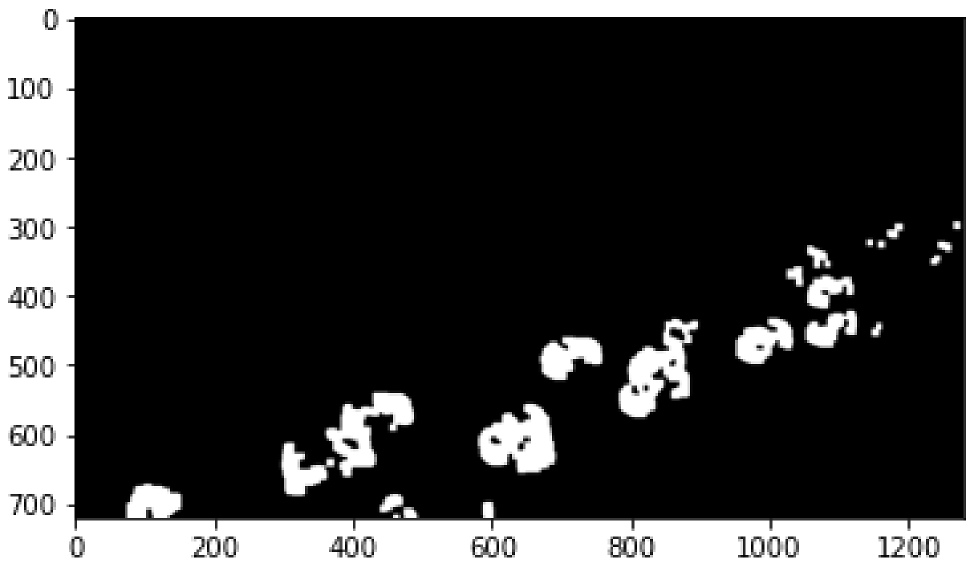
Now, we define structuring elements

```
strucEle3x3 = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
strucEle5x5 = cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))
strucEle7x7 = cv2.getStructuringElement(cv2.MORPH_RECT,(7,7))
strucEle15x15 = cv2.getStructuringElement(cv2.MORPH_RECT,(15,15))
```

```
plot_image([strucEle3x3, strucEle5x5, strucEle7x7, strucEle15x15], cmap='gray', captions =
["strucEle3x3", "strucEle5x5", "strucEle7x7", "strucEle15x15"])
```




```
for i in range(2):
    imgThresh = cv2.dilate(imgThresh,strucEle5x5,iterations = 2)
    imgThresh = cv2.erode(imgThresh,strucEle5x5,iterations = 1)

    imgThreshCopy = imgThresh.copy()
    if(SHOW_DEBUG_STEPS):
        print ('imgThreshCopy height' + str(imgThreshCopy.shape[0]))
        print ('imgThreshCopy width' + str(imgThreshCopy.shape[1]))

plt.imshow(imgThresh, cmap = 'gray')
plt.show()
imgThreshCopy  height720
imgThreshCopy width1280
imgThreshCopy  height720
imgThreshCopy width1280
```



Extracting contours

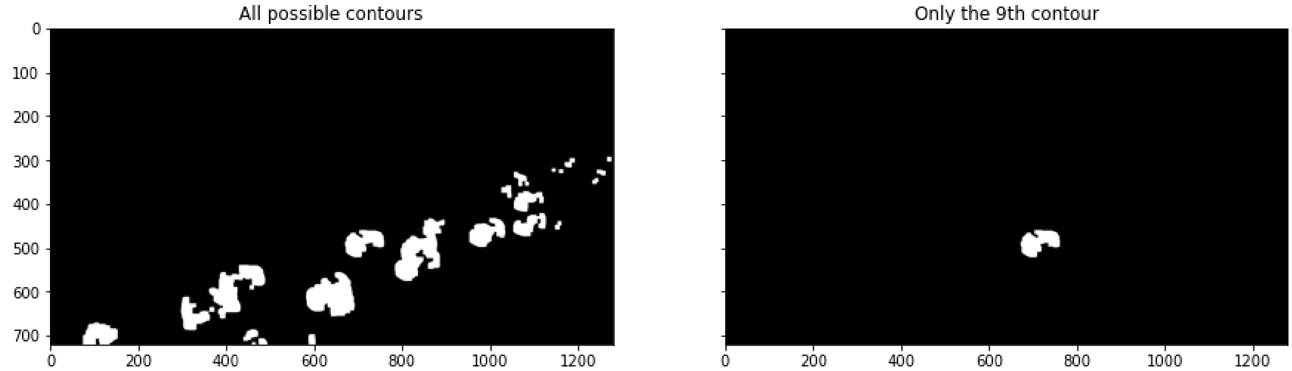
Till now, you have a binary image. Next, we will segment the image and find all possible contours(possible vehicles). The shape of the contours will tell us the number of contours that has been identified. Define *drawAndShowContours()* function to plot the contours. You will see that the threshold image above and the countour image will look alike. So, additionally, we also plot a particular '9th' countour for further clarity.

```
def drawAndShowContours(wd,ht,contours,strImgName):
    global SCALAR_WHITE
    global SHOW_DEBUG_STEPS
```

Defining a blank frame

```
# Defining a blank frame.Since it is initialised with zeros, it will be black. Will add all the coutours in this image.

blank_image = np.zeros((ht,wd,3), np.uint8)

#cv2.drawContours(blank_image,contours,10,SCALAR_WHITE,-1)

    # Adding all possible contour to the blank frame. Contour is white

 cv2.drawContours(blank_image,contours,-1,SCALAR_WHITE,-1)


    # For better clarity, lets just view countour 9

blank_image_contour_9 = np.zeros((ht,wd,3), np.uint8)

    # Let's just add contour 9 to the blank image and view it

cv2.drawContours(blank_image_contour_9,contours,8,SCALAR_WHITE,-1)

    # Plotting

plot_image([blank_image, blank_image_contour_9], cmap='gray', captions = ["All possible contours", "Only the 9th contour"])

        return blank_image

# Now, we move on to the contour mapping portion

contours, hierarchy = cv2.findContours(imgThreshCopy,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
im2 = drawAndShowContours(wd,ht,contours,'imgContours')

# Printing all the coutours in the image.

if(SHOW_DEBUG_STEPS):
```

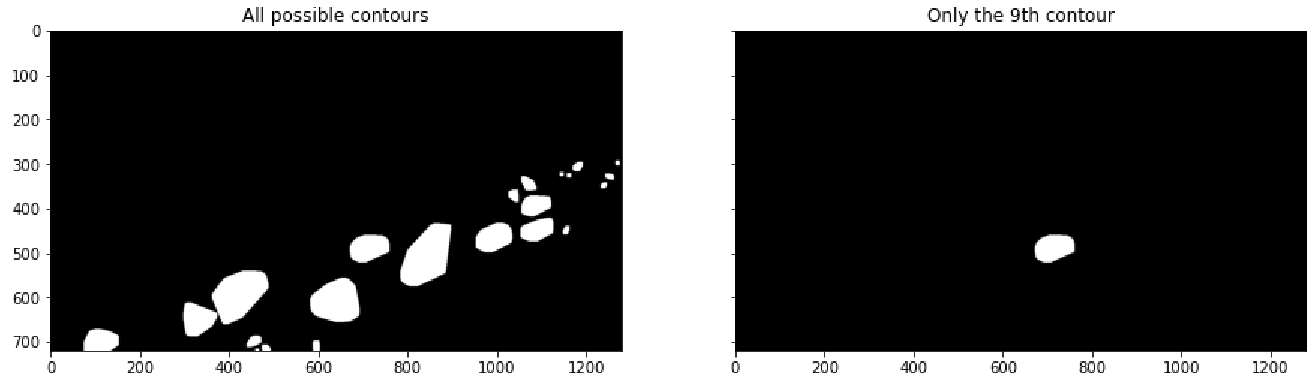
```
print ('contours.shape: ' + str(len(contours)))
```
contours.shape: 22



Hulls

Hulls are contours with the "convexHull".

```
# Next, we define hulls.
# Hulls are contours with the "convexHull" function from cv2

hulls = contours # does it work?
for i in range(len(contours)):
    hulls[i] = cv2.convexHull(contours[i])


# Then we draw the contours

im3 = drawAndShowContours(wd,ht,hulls,'imgConvexHulls')
```



Conclusion:

In Phase 4, we're developing a cutting-edge virtual cinema platform on IBM Cloud Video Streaming, laying the foundation for a dynamic user experience. We've defined essential features like live streaming, video-on-demand, user profiles, interactive chat, content categorization, and payment integration. The user-friendly interface ensures easy navigation on various devices, while robust registration and authentication mechanisms prioritize security and privacy. Moving forward, we'll focus on seamless integration, scalability, and user feedback to refine and enhance the platform, meeting evolving audience demands. Phase 4 puts us on track to create a secure, user-friendly, and feature-rich virtual cinema experience for the digital age.