IDENTIFICATION OF EVENT IN HUMAN OBJECT

Under the guidance of
Dr. Anup Kr. Kolya

GROUP - 23
Arindam Kundu CSE/2013/001
Prolay Kr. Saha CSE/2013/002
Saikat Mondal CSE/2013/003
Srijit Roy Chowdhury CSE/2014/L11

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to our faculty Dr. Anup Kumar Kolya for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark.

We are obliged to our project team members for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our assignment.

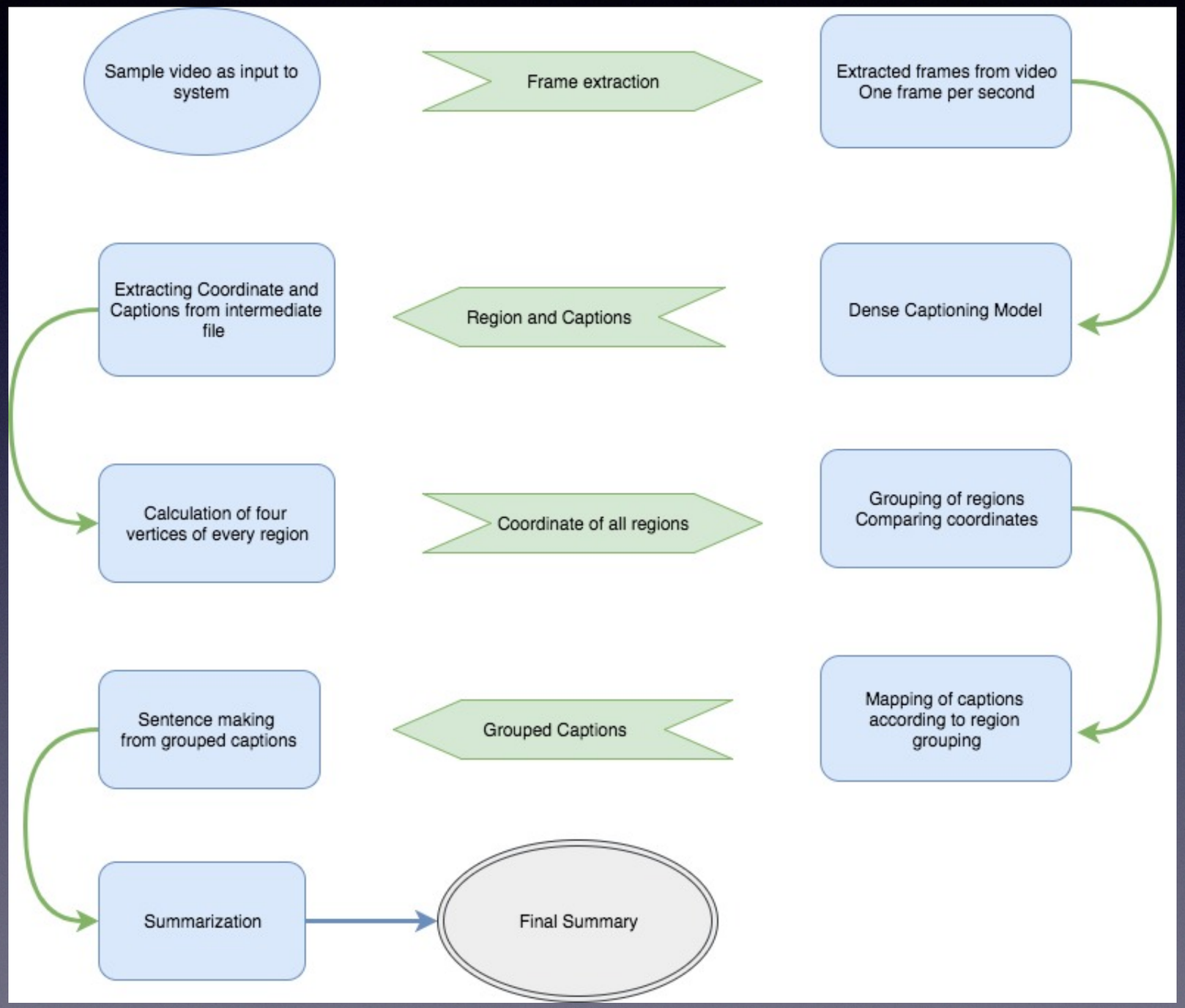
Contents

- Introduction
- Steps to follow
- Frame extraction
- Dense captioning
- Region grouping, caption mapping & sentence making
- Summarisation
- Limitation & Conclusion

Introduction

In this project we have taken an attempt to identify the event in which a human is involved. We have started our work introducing a real world video as input to the system. Our objective is to summarise this video to get the most relevant event in which the object of this video is involved. We have extracted frames from video, identified most generic to specific information related to an object involve in the video and finally summarise this information to get the result.

Steps to follow



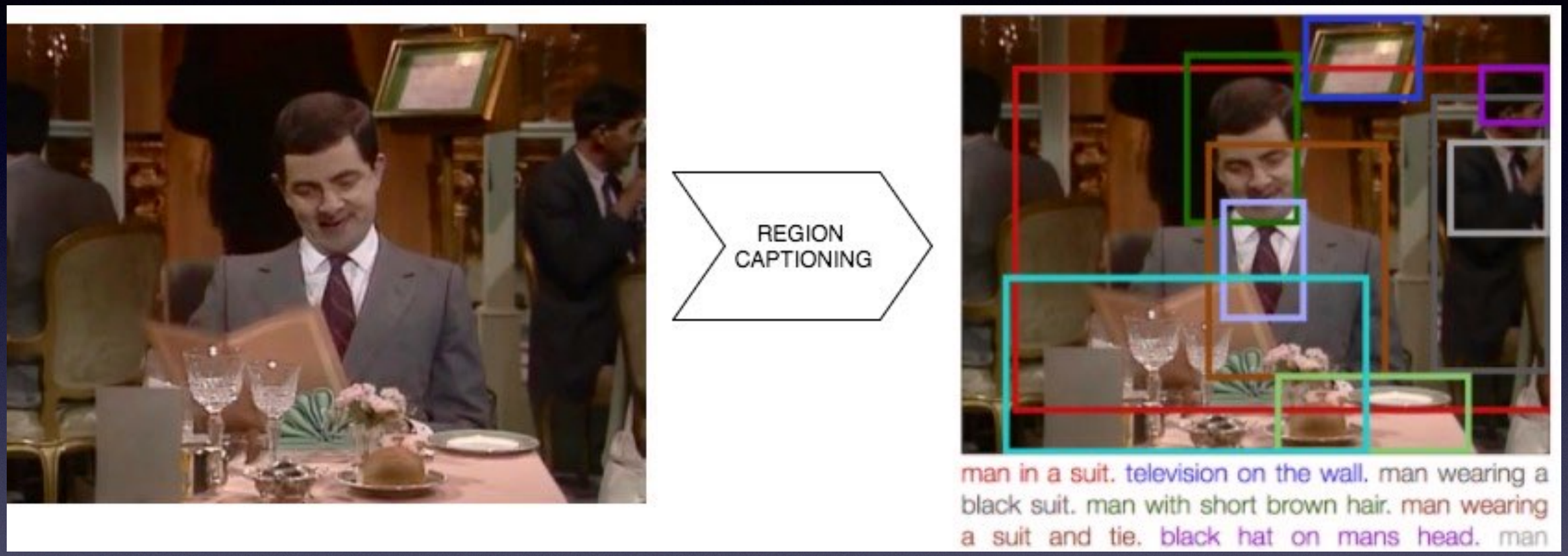
Frame extraction

Video is given as input to frame extraction module.
It extracts one frame per second from a video(in this case 20 frames per second video). We only take frames in which object change is prominent.

Code for frame extraction(python)

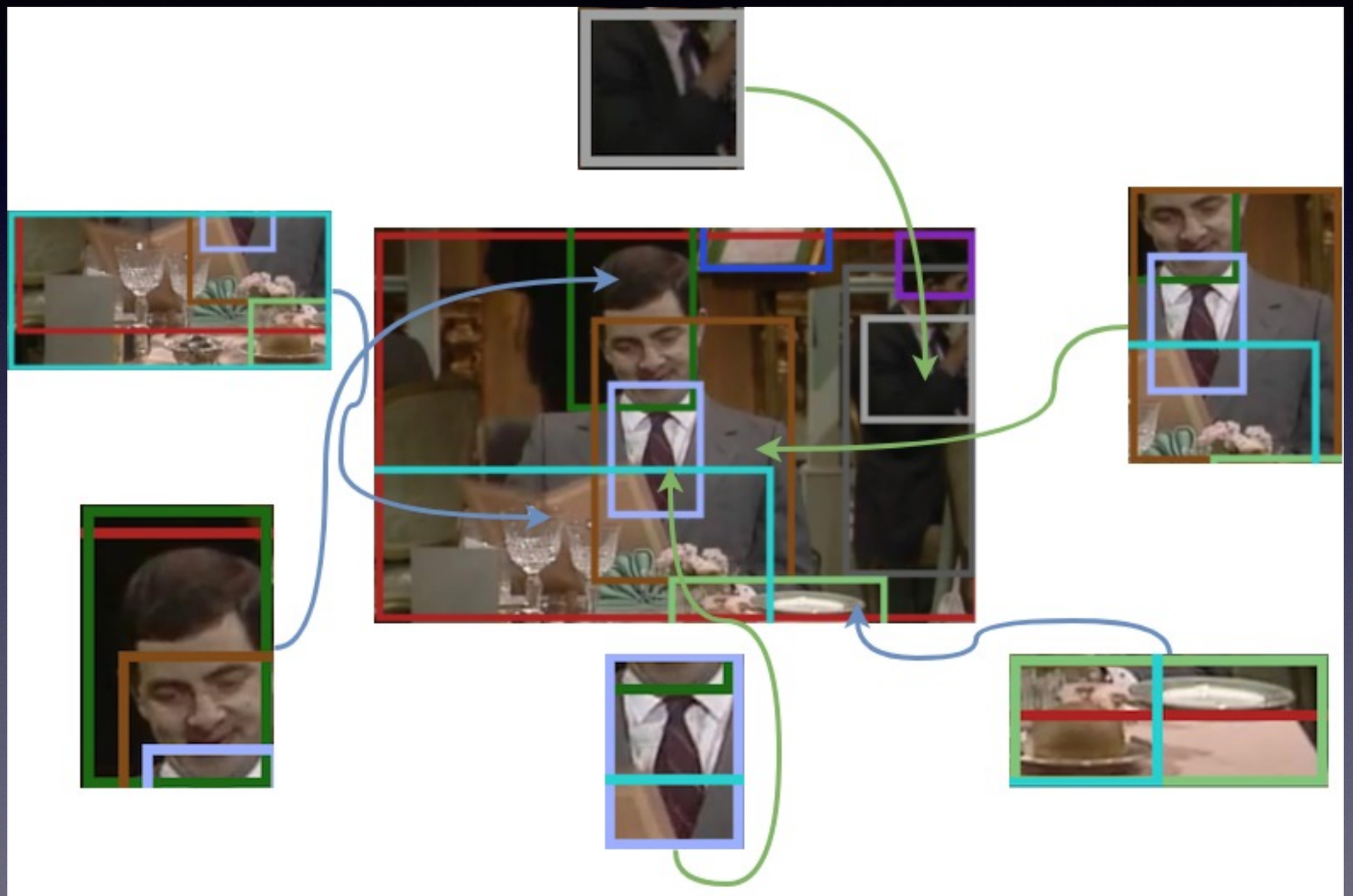
```
import cv2
import math
# input path to video
videoFile = str(input("Enter input to video:"))
# output path for frames
framesFolder = str(input("Enter output path for frames:"))
cap = cv2.VideoCapture(videoFile)
# find frame rate of video
frameRate = cap.get(5)
while cap.isOpened() :
#     current frame number
    frameId = cap.get(1)
#     read frames
    ret, frame = cap.read()
    if (ret != True) :
        break
#         frameid is divisible by framerate
    if int(frameId) % math.floor(frameRate) == 0 :
        filename = framesFolder + "/image_" + str(int(frameId)) + ".jpg"
#         save frames to a folder
        cv2.imwrite(filename, frame)
cap.release()
print "Done!"
```

Dense Captioning



Every frame passes through dense captioning model. This model divides the frame into regions. Every region has a caption associated with it.

Region grouping



Blue arrow shows partially inclusive boxes and Green arrow shows completely inclusive boxes

Coordinates of 4 vertices

```
import os
import json
i = 0
for file in os.listdir('/Users/arin/Desktop/final_yr_project/'
                       +'frame_extract/citmore/imginfodir/') :
    s = json.load(open('/Users/arin/Desktop/final_yr_project/'
                       +'frame_extract/citmore/imginfodir/{}'.format(file)))
    l = []
    for crd in s['boxes'] :
        x1, y1 = crd[0], crd[1]
        h, w = crd[2], crd[3]
#         print('x1:{} y1:{} h:{} w:{}'.format(x1, y1, h, w))
        x2, y2 = x1 + h, y1
        x3, y3 = x1 + h, y1 + w
        x4, y4 = x1, y1 + w
        t = (x1, y1, x2, y2, x3, x3, x4, y4)
#         print('x1:{} y1:{} x2:{} y2:{} x3:{} y3:{} x4:{} y4:{}\n'.format(x1,y1,x2,y2,x3,y3,x4,y4))
        l.append(t)
    json.dump(l, open('/Users/arin/Desktop/final_yr_project/'
                      +'frame_extract/citmore/imgcrd/imgcrdinfo{}.json'.format(i), 'w'))
    print(l)
    print('----------------------')
    i = i + 1
    l = []
```

From (x1, y1) and (height, width) :
(x2, y2), (x3, y3), (x4, y4)

Comparison of vertices

Coordinate of regions:

```
[(448.91363525391, 227.10025024414, 559.01666259766, 227.10025024414, 559.01666259766, 559.01666259766, 448.913635253
91, 364.63320922852), (297.74209594727, 70.234451293945, 339.472137451176, 70.234451293945, 339.472137451176, 339.472
137451176, 297.74209594727, 235.529373168945), (5.6920547485352, 66.218231201172, 167.2067947387652, 66.218231201172,
167.2067947387652, 167.2067947387652, 5.6920547485352, 200.411987304692), (625.05224609375, 244.04396057129, 701.5667
72460938, 244.04396057129, 701.566772460938, 701.566772460938, 625.05224609375, 356.79931640625), (695.56787109375, 2
```

Grouping of regions by comparison of coordinate information. Grouping of completely and partially inclusive boxes to a region.

Completely inclusive:

(X1' >= X1 AND Y1' >= Y1) AND (X2' <= X2 AND Y2' >= Y2)
AND (X3' <= X3 AND Y3' <= Y3) AND (X4' >= X4 AND Y4' <= Y4)

Partially inclusive:

(X1' + 10 >= X1 AND Y1' + 10 >= Y1) AND (X2' – 10 <= X2 AND Y2' + 10 >= Y2)
AND (X3' – 10 <= X3 AND Y3' – 10 <= Y3) AND (X4' + 10 >= X4 AND Y4' – 10 < = Y4)

Code for region grouping(python)

```
import os
import json
k = 0
for file in os.listdir('/Users/arin/Desktop/final_yr_project/'
                       +'frame_extract/citmore/imgcrd/'):
    l = json.load(open('/Users/arin/Desktop/final_yr_project/'
                       +'frame_extract/citmore/imgcrd/{}'.format(file)))
    inclv = {}
    i = 0
    for crd in l[:]:
        j = 0
        temp = {}
        for test in l[:]:
#             for complete inclusive boxes
            if (
                (test[0]>=crd[0] and test[1]>=crd[1]) and
                (test[2]<=crd[2] and test[3]>=crd[3]) and
                (test[4]<=crd[4] and test[5]<=crd[5]) and
                (test[6]>=crd[6] and test[7]<=crd[7]) and
                i != j
                ):
                temp[j] = test
#           for partly inclusive boxes
            elif (
                (test[0] + 10 >= crd[0] and test[1] + 10 >= crd[1]) and
                (test[2] - 10 <= crd[2] and test[3] + 10 >= crd[3]) and
                (test[4] - 10 <= crd[4] and test[5] - 10 <= crd[5]) and
                (test[6] + 10 >= crd[6] and test[7] - 10 <= crd[7]) and
                i != j
                ):
                temp[j] = test
            j = j + 1
        if len(temp.keys()) == 0:
            inclv[i] = {}
        else :
            inclv[i] = temp
        i = i + 1
    print(inclv)
    print('------------------')
    json.dump(inclv, open('/Users/arin/Desktop/final_yr_project/'
                          +'frame_extract/citmore/grouping/boxgrouping{}.json'.format(k), 'w'))
    k = k + 1
```

Result of region grouping

{0: {}, 1: {}, 2: {17: [21.303546905518, 56.571365356445, 104.890872955323, 56.571365356445, 104.890872955323, 104.89
0872955323, 21.303546905518, 115.705902099609]}, 3: {}, 4: {6: [717.78485107422, 220.7709197998, 720.48358154297, 22
0.7709197998, 720.48358154297, 720.48358154297, 717.78485107422, 292.442535400385961}, 5: {}, 6: {}, 7: {0: [448.9136
3525391, 227.10025024414, 559.01666259766, 227.10025024414, 559.01666259766, 559.01666259766, 448.91363525391, 364.63
320922852], 1: [297.74209594727, 70.234451293945, 339.472137451176, 70.234451293945, 339.472137451176, 339.4721374511
76, 297.74209594727, 235.529373168945], 2: [5.6920547485352, 66.218231201172, 167.2067947387652, 66.218231201172, 16
7.2067947387652, 167.2067947387652, 5.6920547485352, 200.411987304692], 5: [164.02105712891, 244.55447387695, 222.870
239257816, 244.55447387695, 222.870239257816, 222.870239257816, 164.02105712891, 322.217254638669031, 8: [2.628334045
4102, 232.95195007324, 163.3941421508802, 232.95195007324, 163.3941421508802, 163.3941421508802, 2.62833404541021, 39
6.174072265621], 13: [369.5666809082, 297.72595214844, 559.09118652343, 297.72595214844, 559.09118652343, 559.09118652
343, 369.5666809082, 391.540832519534], 16: [138.25224304199, 228.68853759766, 195.42448425292798, 228.68853759766, 1
95.42448425292798, 195.42448425292798, 138.25224304199, 333.967285156251], 17: [21.303546905518, 56.571365356445, 104.

Grouped regions are mapped to captions as every region has a caption associated with it. Therefore, mapping of grouped region to caption results in grouping of captions for a region.

Code for region to caption mapping(python)

```
import os
import json
x = 0
y = 1
g = 0
i = 0
cpt = {}
for file in os.listdir('/Users/arin/Desktop/final_yr_project/'
                       +'frame_extract/citmore/grouping/'):
    l = json.load(open('/Users/arin/Desktop/final_yr_project/'
                       +'frame_extract/citmore/grouping/{}'.format(file)))
    if i == 2:
        i = 0
        y = y + 1
    r = json.load(open('/Users/arin/Desktop/final_yr_project/'
                       +'frame_extract/citmore/citmoreresults/results{}.json'.format(y)))
    c = r['results'][i]['captions']
    for key in l:
        cptlist = []
        for k in l[key].keys():
            t = {k:c[int(k)]}
            cptlist.append(t)
        cpt[str((key, c[int(key)]))] = cptlist
    i = i + 1
    print(cpt)
    print('----------------')
    json.dump(cpt, open('/Users/arin/Desktop/final_yr_project/'
                        +'frame_extract/citmore/grptocpt/cpt{}.json'.format(g), 'w'))
    g = g + 1
```

Result of region to caption mapping


```
{"('0', 'a man on a motorcycle')": [], "('1', 'a light pole')": [], "('2', 'a sign on the building')": [{'17': 'a blu
e and white sign'}], "('3', 'a man on a motorcycle')": [], "('4', 'white van parked on the street')": [{'6': 'white v
an parked on the street'}], "('5', 'a man walking down the street')": [], "('6', 'white van parked on the street')":
 [], "('7', 'a street scene')": [{'0': 'a man on a motorcycle'}, {'1': 'a light pole'}, {'2': 'a sign on the buildin
g'}, {'5': 'a man walking down the street'}, {'8': 'man with a black shirt'}, {'13': 'a motorcycle on the road'}, {'1
6': 'a man walking on the sidewalk'}, {'17': 'a blue and white sign'}, {'18': 'a street light'}, {'19': 'green trees
 in the background'}, {'21': 'a fence in the background'}, {'22': 'a tree in the distance'}, {'23': 'people walking o
n the sidewalk'}, {'25': 'a person wearing a helmet'}, {'28': 'a white sign on the sidewalk'}, {'29': 'man wearing a
 hat'}, {'35': 'a sign on the wall'}, {'36': 'a sign on the side of the road'}, {'37': 'a sign on the building'}, {'3
8': 'a light pole'}, {'39': 'a light pole'}, {'40': 'man with black hair'}, {'41': 'a blue and white sign'}, {'42':
 'trees in the background'}, {'43': 'power lines above the street'}, {'44': 'a blue and white sign'}, {'45': 'a man w
earing a hat'}, {'46': 'a white car'}, {'47': 'a man in a black shirt'}, {'49': 'a man with a black hat'}, {'50': 'a
```


From grouped captions we get chunk of sentences for a particular region and summarising results final summary.

Summarisation

Input to summarisation algorithm:

```
text = """
        a white box. a pair of brown pants. a man wearing a tie.
        the shirt is black. black hat on mans head. a person standing in the background.
        a wooden door. man has white shirt. a white door. a wooden table.
        a man wearing a white shirt. man wearing a black suit. the man has glasses.
        a man in a black shirt. a white shirt on a man. a man in a white shirt.
        man wearing a white shirt. the mans hair is black. man wearing a suit and tie.
        a pair of blue jeans. a glass of wine. a white box on the table.
        a picture of a woman. the man is wearing a tie. a white napkin in the background.
        a chair with a white seat.
"""
```

Output from summarisation algorithm:

```
A man wearing tie, black suit and white shirt
```

Code for summarisation(python)

Tokenise the sentence

```
for line in sents:
    parse = nltk.word_tokenize(line)
    tags.append(nltk.pos_tag(parse))
counts = Counter([item for line in tags for item in line])
  print tags
```

```
[('a', 'DT'), ('man', 'NN'), ('wearing', 'VBG'), ('a', 'DT'), ('white', 'JJ'), ('shirt', 'NN')]
------------------
[('a', 'DT'), ('white', 'JJ'), ('shirt', 'NN'), ('on', 'IN'), ('a', 'DT'), ('man', 'NN')]
------------------
[('man', 'NN'), ('wearing', 'VBG'), ('a', 'DT'), ('suit', 'NN'), ('and', 'CC'), ('tie', 'NN')]
------------------
[('a', 'DT'), ('man', 'NN'), ('in', 'IN'), ('a', 'DT'), ('white', 'JJ'), ('shirt', 'NN')]
------------------
[('man', 'NN'), ('has', 'VBZ'), ('white', 'JJ'), ('shirt', 'NN')]
------------------
[('a', 'DT'), ('man', 'NN'), ('in', 'IN'), ('a', 'DT'), ('black', 'JJ'), ('shirt', 'NN')]
```

Code for summarisation(python)

Find most common noun, article & verb:

```
for item in counts.most_common():
    if item[0][1] == 'NN':
        targetNN = item[0][0]
        break

for item in counts.most_common() :
    if item[0][1] == 'DT':
        targetDT = item[0][0]
        break
```

```
for item in Counter([item for line in target_sents_NN for item in line]).most_common():
    if item[0][1] == 'VBG':
        targetVBG = str(item[0][0])
        break
```

Code for summarisation(python)

Join most common noun, article and verb to form the summary

```
req_objects = filter(lambda x: [x for i in obj if x in i and x != i] == [], obj)
final_text = ' '.join([targetDT, prefix]) + ', '.join(req_objects[:-1]) + ' and ' + req_objects[-1]
```

Final Summary:

```
A man wearing tie, black suit and white shirt
```

Limitation

- Accuracy of dense captioning depends on how well the model is trained. Image dataset of size 9 -12 gb and machine with premium hardware is needed to train this model.

- Designing of a summarisation algorithm is itself an open research which takes significant time and effort. Our approach of summarisation is not perfect and mostly it is input specific.

Conclusion

Starting from the frame extraction step to the summarisation step, we have faced lot of difficulties to achieve our short term goal or target result of each step. To over come our hurdles, we take help from various algorithms and mathematical formulas. Though the final result is not accurate but we have gradually identified the limitations and the area of betterment of this project. Better design of summarisation algorithm and correct information extraction from the video may reasonably increase the performance of this system.

Thank you

#same3new1