A Technical Review on Online Video Document Reader

Khel Prakash Jayant*
Mayank Mittal**
Akansha Daiya***

ABSTRACT

Text recognition of documents in captured frames from video in real-time is an active research area that aims to develop a computer application with the ability to automatically read text from images. The huge amount of data stored in video format can be extracted to obtain further useful information. Online verification of data during video calls is necessary since people are not physically available most of the time. Generally, images can be scanned, stored in a computer system, and the text data can be extracted. Another way to extract text data from captured scanned images from video frames is to scan them. It is easy to retrieve details of documents and store such information in a database. This information can be stored in various formats, such as PDF. Extracting text data from images or scanned video frames can be challenging when the image or captured video frame quality is blurry or unclear. Sometimes, it is impossible to recognize characters while reading different font character formats. Thus, character recognition mechanisms are required to perform document image analysis, which transforms hard copy images of documents into electronic format. In this paper, we reviewed and analyzed different methods for text recognition from images. The objective of such techniques is to store real-time information in computer systems.

Key words *: Text recognition, Pattern recognition, Character recognition, video frames, online document recognition*

Introduction

Currently, there is a growing demand for software systems that can recognize characters in scanned information stored in video format on computers. As we are aware, everything is moving online, and identities or documents are verified through video calls. Therefore, there is a need to store video content on a computer disc and then later retrieve this information through a search process. One simple way to store information from these videos on a computer system is to first capture frames from the video and identify which frame contains text. Then, that frame can be stored as an image, and the document details can be scanned from that image. When we scan documents through a scanner, they are stored as images in the computer system. However, the text in these images cannot be edited by the user. Furthermore, it can be very difficult for the computer system to read and search the contents of these documents line-by-line and word-by-word for reuse of the information.

In this project, the focus is on scanning Aadhar Cards.

Litrature Review

- A Document Reader is a software application or script used to create a PDF document in real-time during a video consultation. The objective is to store useful information in the system on a real-time basis.
- Real-time Document Reader technology falls

*Professor, CSE Department, DVSIET, Meerut
**Assistant Professor, DVSIET, Meerut
***Graduate Engineer Trainee, HCL- PUNE

under the field of pattern recognition. Many researchers have invented various technologies to recognize text details from hardcopy images or video frames, but each approach or technology addresses the issues in a different way.

- Firstly, the Document Reader extracts frames from the video on a real-time basis and stores them for further processing.
- Next, all frames are passed for Aadhaar Card detection and face detection. After detecting the Aadhaar Card, the frame is passed to the PDF maker.
- At the end, the PDF maker is called to create the PDF file and store it in the database.
- Many approaches have been proposed by researchers to handle the issues related to text recognition.
- Some have proposed a novel adaptive binarization method based on wavelet filters. Such an approach is processed faster compared to others, making it more suitable for real-time processing and applicable for mobile devices. They evaluated this adaptive method on complex scene images from the ICDAR 2005 database.
- In this regard, some problems in text recognition are also discussed.
- Automated optical character recognition (OCR) tools do not provide a complete solution, and in most cases, human inspection is required.
- To overcome this issue, we used Tesseract OCR, Haarcascade XML Files, and some other Python libraries.

Tools And Technologies Used In Document Reader

In this section we will go through the overview of all the tools and technologies used in Document Reader which are as follows:

Tesseract: Tesseract is an optical character recognition engine that can be used on various operating systems. It is released under the Apache License and is free software. Originally developed as proprietary software by Hewlett-Packard in the 1980s, it was released as open-source in 2005. Google has sponsored its development since 2006. In 1995, Tesseract was one of the top three OCR engines in terms of character accuracy. It is available for Linux, Windows, and Mac OS X. However, due to limited resources, it is only rigorously tested by developers under Windows and Ubuntu.

Opencv: OpenCV is a vast open-source library used for computer vision, machine learning, and image processing. It supports a wide range of programming languages such as Python, C++, Java, and more. OpenCV can process images and videos to identify objects, faces, or even human handwriting. OpenCV-Python makes use of NumPy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from NumPy arrays.

OpenCV stands for Open Source Computer Vision Library. It is a huge open-source library that provides tools and algorithms for both academic and commercial applications in computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. OpenCV also has the capability to perform real-time computer vision, which makes it a popular choice for many projects in robotics, automotive, medical, and security industries.

Haarcascade: HaarCascade is a machine learning-based approach that uses a large number of positive and negative images to train the classifier. It is an algorithm that can detect objects in images, regardless of their scale in the image and location. This algorithm is not very complex and can run in real-time. We can train a HaarCascade detector to detect various objects such as cars, bikes, buildings, fruits, and more. Haar Cascade is a machine learning-based object detection approach used to identify objects in images or videos. It is an algorithm that uses a trained classifier to detect objects, regardless of their size or location within the image or video. This approach is based on Haar-like features, which are patterns of contrast differences in the pixel intensities of an image. The Haar Cascade classifier uses these features to identify objects in the image. Haar Cascade classifiers have been trained to detect objects such as faces, eyes, and cars, and can be trained to detect custom objects as well.

Computer Vision:

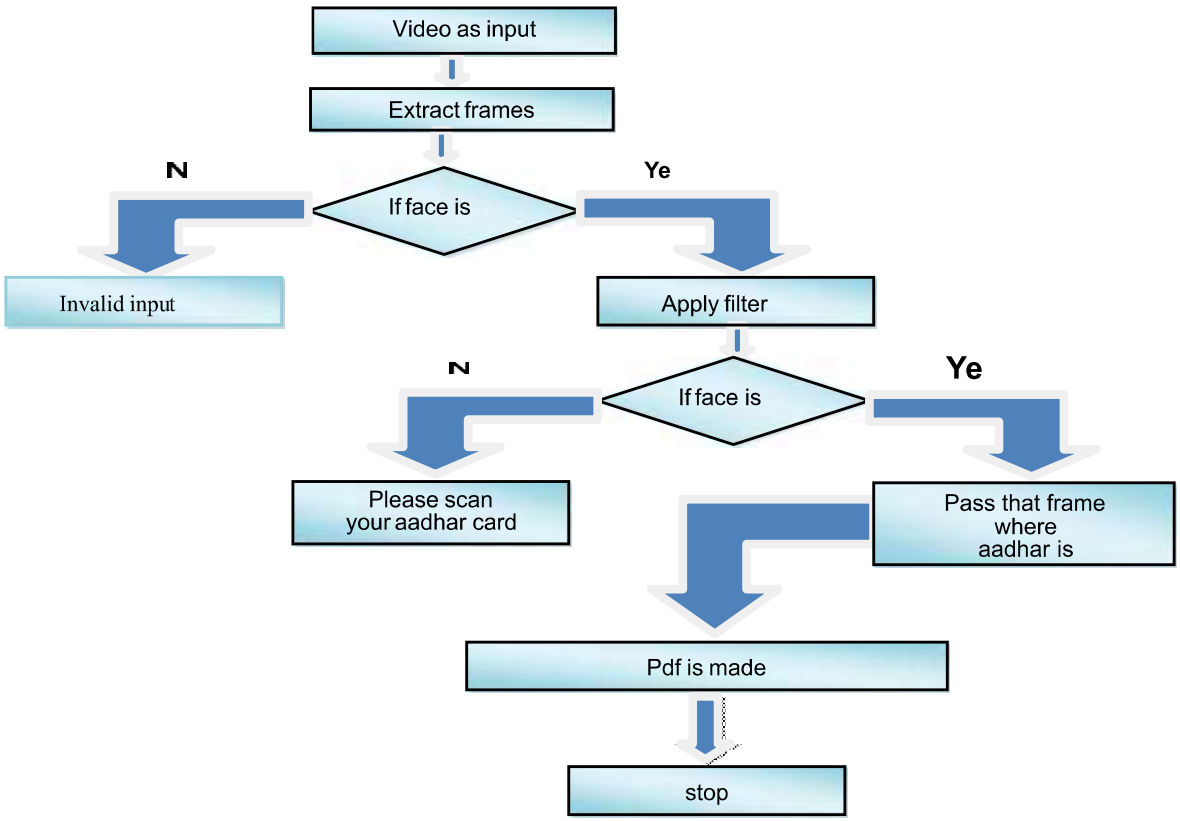
Computer vision is concerned with modeling and replicating human vision using computer software and hardware. It is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs. Based on that information, they can take actions or make recommendations.

Computer vision is a field of artificial intelligence (AI) that is concerned with enabling computers and systems to replicate and model human vision using computer software and hardware. It enables computers to derive meaningful information from digital images, videos, and other visual inputs and take actions or make recommendations based on that information. The ultimate goal of computer vision is to enable computers to perform tasks that typically require human visual perception and understanding, such as object recognition, face detection, image and video analysis, and more.

Empirical Study of Document Reader:

We have proposed and described an overall architecture of a Document Reader. The first step of the Document Reader involves capturing images from a video in the form of frames. A frame is a single image in a sequence of pictures. The Document Reader receives input in the form of frames, which contain some text information. At this stage, we have the data in the form of frames, and these frames can be further analyzed so that important information can be retrieved.

After identifying the text information from the frames, the Document Reader fetches essential information from the Aadhar card, such as name, DOB, gender, Aadhar number, etc. The output of this system is in PDF format, and it also stores the information in a database, i.e., text information frames are stored in a computer-readable form.



Functional Code

```
tessdata_dir_config = r'--tessdata-dir "C:\Program Files\Tesseract-OCR\tessdata"'
tess.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
mouth_Cascade = cv2.CascadeClassifier('haarcascade_mcs_mouth.xml')

#Path  of where the frames is stored for aadhar detection
images = r"D:\python\doc_reader\New folder\data"

#path of photos folder
data_face = r"D:\python\doc_reader\New folder\data_face"

#Path of folder where pdf converter is used
final_image = r"D:\python\doc_reader\New folder\final_img"

#n = sys.argv[1]
n = r"D:\python\doc_reader\New folder\4.mp4"
#n=r"D:\python\doc_reader\akansha.mp4"
path_pdf, file_name = os.path.split(n)

file_name = file_name.replace(".mp4",".pdf")

#Path of  where pdf is made
pdf = r"D:\python\doc_reader\New folder\pdf/{}".format(file_name)
cap = cv2.VideoCapture(n)

frame_rate = 1
prev = 0
i = 0
get_adhar_no = []
def adhaar_read_data(text):
    res = text.split()
    name = None
    dob = None
    adh = None
    sex = None
    nameline = []
    dobline = []
    text0 = []
    text1 = []
    text2 = []
    lines = text.split('\n')
    for lin in lines:
        s = lin.strip()
        s = lin.replace('\n', '')
        s = s.rstrip()
        s = s.lstrip()
        text1.append(s)

    if 'female' in text.lower():
        sex = "FEMALE"
    else:
        sex = "MALE"

    text1 = list(filter(None, text1))
    text0 = text1[:]
    try:
        # Cleaning first names
        name = text0[0]
        name = name.rstrip()
        name = name.lstrip()
        name = name.replace("8", "B")
        name = name.replace("0", "D")
        name = name.replace("6", "G")
        name = name.replace("1", "I")
        name = re.sub('[^a-zA-Z] +', ' ', name)

        # Cleaning DOB
        dob = text0[1][-10:]
        dob = dob.rstrip()
        dob = dob.lstrip()
        dob = dob.replace('l', '/')
        dob = dob.replace('L', '/')
        dob = dob.replace('I', '/')
        dob = dob.replace('i', '/')
        dob = dob.replace('|', '/')
        dob = dob.replace('\"', '/1')
        dob = dob.replace(":", "")
        dob = dob.replace(" ", "")
        # Cleaning Adhaar number details
        aadhar_number = ''
        for word in res:
            if len(word) == 4 and word.isdigit():
                aadhar_number = aadhar_number + word + ''
        #if len(aadhar_number)>=14:
                #       print("Aadhar number is :"+ aadhar_number)
        #else:
```

```
        #    print("Aadhar number not read")
        adh = aadhar_number
        data = {}
        data['Name'] = name
        data['Date of Birth'] = dob
        data['Adhaar Number'] = adh
        data['Sex'] = sex
        data['ID Type'] = "Adhaar"
        return data
    except Exception as e:
        print(e)
def findword(textlist, wordstring):
    lineno = -1
    for wordline in textlist:
        xx = wordline.split()
        if ([w for w in xx if re.search(wordstring, w)]):
            lineno = textlist.index(wordline)
            textlist = textlist[lineno + 1:]
            return textlist
    return textlist

# storing image path
# i m g _ p a t h                                    =
"D:/office_programs/doc_scanner/myImage0.jpg"

# storing pdf path
# p d f _ p a t h                                    =
"D:/office_programs/doc_scanner/file.pdf"

def face_reader(face):
    got_eyes = []
    got_mouth = []
    final = []

    img = cv2.imread(r"D:\python\doc_reader\New
folder\data_face/"+face)
        g r a y   =   c v 2 . c v t C o l o r ( i m g   ,
cv2.COLOR_BGR2GRAY)
    eyes = eye_cascade.detectMultiScale(gray , 1.3, 5)
     mouths = mouth_Cascade.detectMultiScale(gray,
1.3,5)
    if len(eyes) != 0:
        got_eyes.append(face)
    if len(mouths) != 0:
        got_mouth.append(face)
    for im in got_eyes:
        if im in got_mouth:
            final.append(im)
    if len(final) == 0:
        for i in got_eyes:
            final.append(i)
        for j in got_mouth:
            final.append(j)

    i = 0
    for file in final:
                loc = r"D:\python\doc_reader\New
folder\data_face/"+file
                dest = r"D:\python\doc_reader\New
folder\final_img/"+file
        shutil.move(loc, dest)
        if i == 0:
            break
def convert2pdf(final_image):
      imagelist = listdir(final_image) # get list of all
images
     pdf = FPDF('P','mm','A4') # create an A4-size pdf
document

    x,y,w,h = 20,0,100,100

    for image in imagelist:
        pdf.add_page()
              pdf.image(r"D:\python\doc_reader\New
folder/final_img/" +image,x,y,w,h)

        pdf.output(r"D:\python\doc_reader\New
folder\pdf/{}".format(file_name), 'F')
    print("pdf successful made")

try:
    while True:
        time_elapsed = time.time() - prev
        ret, frame = cap.read()
        if  time_elapsed >1./ frame_rate:
            prev = time.time()
            cv2.resize(frame , (500,300))
                      gray = cv2.cvtColor(frame,
cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray,
1.1, 4)
            for (x,y,w,h) in faces:
                # To draw a rectangle in a face
                            #cv2.rectangle(frame,
(x,y),(x+w,y+h),(255,255,0),2)
                roi_gray = gray[y:y+h, x:x+w]
                roi_color = frame[y:y+h, x:x+w]

cv2.imwrite('data/frame{}.jpg'.format(str(i)),
frame)

cv2.imwrite('data_face/frame{}.jpg'.format(str(i)),
```

```
roi_color)
        i += 1
  cap.release()
  cv2.destroyAllWindows()
except Exception as e:
  print()

get_adhar_no = []
tolist = []

try:
  folder_length = len(os.listdir(data_face))
  if folder_length != 0:
    for face in os.listdir(data_face):
      print(face_reader(face))
      break


  for image in os.listdir(images):
    if image.endswith('.jpg'):
      if len(tolist) == 0:
        os.chdir(images)
                    # tessdata_dir_config =
'/usr/local/share/tessdata/'
        img = cv2.imread(image)
         img = cv2.resize(img, None, fx=1.2, fy=1.2,
interpolation=cv2.INTER_CUBIC)
                    img = cv2.cvtColor(img,
cv2.COLOR_BGR2GRAY)

        kernel = np.ones((1, 1), np.uint8)
        img = cv2.dilate(img, kernel, iterations=1)
        img = cv2.erode(img, kernel, iterations=1)
        #cv2.imshow('bgr', img)

                    d = tess.image_to_data(img,
output_type=Output.DICT, lang='eng',
config=tessdata_dir_config)
        n_boxes = len(d['level'])
        all_text = tess.image_to_string(img)
        print(len(all_text))
        if len(all_text) == 0:
          os.remove(img)
        if len(all_text) > 10:
          # print(adhaar_read_data(all_text))
          all_data = (adhaar_read_data(all_text))
          print(all_data)

          for key, value in all_data.items():
            #print(key, value)
            if key == 'Adhaar Number':
              # get_ac = value.replace(" ", "")
              if len(value) > 10:
                get_adhar_no.append(value)
                tolist.append(value)
                value = value[:14]
                v = "'" + value + "'"
                #print(v)
                print("AADHAR NO -->", value)

#cv2.imwrite('data_face/frame{}.jpg'.format(str(i)),
image)
                                image_path =
os.path.abspath(r"D:\python\doc_reader\New
folder\data/" + image)
                                shutil.copy(image_path ,
final_image)
                                #shutil.copy(photo_path ,
final_image)
                print('PDF making start')
                print(convert2pdf(final_image))


        else:
          os.remove(image)

except Exception as e:
  print()
if len(tolist) == 0:
    print("plz scan your adhaar card , this ain't your
adhaar card.")
#for image in os.listdir(images):
#  os.remove(image)
```

Future Aspects • The document reader can be used for online document verification during admission to colleges or schools. It can also be used in the healthcare profession as it deals with a large volume of forms for each patient. • It is widely used in the legal industry for scanning documents and entering their information into a computer database. We can use document scanner for this purpose.

Conclusion • The main objective of this system is to create a PDF document file using real-time video interaction through a webcam. • It takes less time compared to physical documentation on the spot. • This technology is used in many areas such as education, banking, government and private organizations, etc. • It is useful for online document verification, which saves time and cost. It is also useful in business financial records and activities

such as tracking backlog payments to prevent them from piling up. It also helps in advanced technologies like artificial intelligence and machine learning applications.

Applications Online verification via document readers from video frame applications is now widely used. This technology may apply throughout the entire spectrum of industries, revolutionizing document management. o It enables scanned documents to become more than just image files. It turns them into fully searchable documents with text content that is recognized by computers. o With the help of this technology, people no longer need to manually retype important documents when entering them into electronic databases. Instead, the document reading system extracts relevant information and enters it automatically. o The result is accurate, efficient information processing in less time. o Some applications of text recognition systems are Banking, Consultancy, Healthcare, Medical Science, Education, Legal Industries, etc.

Summary In this paper, we have reviewed and analyzed different techniques to find text characters from video frames. We have reviewed the methods and architecture of the document reader from images and video frames. We investigated image processing operations, specifically a sequence of frames for a document reader. We have also used some applications of document reader systems.

REFERENCES

Pratik Madhukar Manwatkar , Shashank H. Yadav, " Text Recognition from Images", IEEE Sponsored 2nd International Conference on Innovations in Information, Embedded and Communication systems (ICIIECS)2015

PM Manwatkar , Dr. Kavita R. Singh, " A Technical Review on Text Recognition from Images," , IEEE Sponsored 9th International Conference on Intelligent Systems and Control (ISCO) 2015

Ntirogiannis, Konstantinos, Basilis Gatos, and Ioannis Pratikakis. "A Performance Evaluation Methodology for Historical Document Image Binarization.," IEEE International Conference on Document Analysis and Recognition, 2013.

Malakar, Samir, et al. "Text line extraction from handwritten document pages using spiral run length smearing algorithm," IEEE International Conference on Devices and Intelligent Systems (CODIS), 2012.

T.Som, Sumit Saha, "Handwritten Character Recognition Using Fuzzy Membership Function", International Journal of Emerging Technologies in Sciences and Engineering, Volume 5, December 2011.

M. Cai, J. Song and M. R. Lyu, "A New Approach for Video Text Detection," International Conference On Image Processing, Rochester, New York, USA, pp. 117-120, 2002

Naveen Sankaran and C.V Jawahar, "Recognition of Printed Devanagari Text Using BLSTM Neural Network," IEEE, 2012.

Text Recognition from Images, "Pratik Madhukar Manwatkar". https://ieeexplore.ieee.org/document/7193210

Stuart Taylor, Chris Dance, William Newman, Alex Taylor, "Advances in Interactive Video Scanning of Paper Documents", CiteSeer, October 2004

Taylor M. J., Zappala A., Newman, W. M. and Dance C. R. Documents Through Cameras. To appear in Image and Vision Computing, 1999.

L. Agnihotri and N. Dimitrova. "Text Detection for Video Analysis," International Conference on Multimedia Computing and Systems, Florence, Italy, pp. 109-113, 1999.