فرمت تهیه گزارش

مهدی پاس یگانه - 9712762409

اطلاعات گزارش

تاریخ:

واژگان کلیدی:
Sift
Surf
Detection
keypoints

چکیده

در این مقاله نمونه، روش تهیه مقاله، بخش‌های مختلف آن، انواع قلم‌ها و اندازه آن‌ها که در تهیه یک مقاله برای «مجله علمی و پژوهشی مدل سازی در مهندسی» بکار می‌روند، آمده است. کلیه شیوه‌های مورد نیاز برای بخش‌های مختلف مقاله، مانند عنوان، نام نویسندگان، چکیده و متن، از پیش تعریف شده و تنها کافی است که این شیوه‌ها بر مقاله تهیه شده توسط مؤلف تطبیق داده شوند. از نویسندگان محترم درخواست می‌شود این شیوه‌نامه را در هنگام تهیه مقاله به دقت رعایت فرمایند و مسئولین مجله را در راستای ارتقای کیفیت یاری نمایند. چکیده باید طی یک یا دو پاراگراف و حداکثر 250 کلمه بطور صریح موضوع، روش تحقیق، اهم نتایج تحقیق انجام شده و روش ارزیابی را مطرح کند. آوردن جدول، شکل یا فرمول در چکیده مجاز نمی باشد. متاسفانه مجله از چاپ مقالات ارسالی که خارج از روش ارائه شده در این شیوه نامه باشند، معذور است.

1-مقدمه

این پروژه به 3 فاز تقسیم می شود که هرکدام را به صورت جداگانه بررسی میکنیم (در قسمت اول به توضیح تئوری و در بخش کد ها به توضیح کد های ان می پردازم)

2-فاز 1

در این فاز به بررسی روش درست detect کردن تصاویر می پردازیم

sift -2.1

این روش سرعت پایینی دارد ولی در عین حال دقت ان از سایر روش های موجود بالاتر است

در این روش هر key point که تشخیص داده می شود با 128 ویژگی توصیف میشود

surf -2.2

این روش سرعت بالاتری دارد ولی دقت ان کمتر است

برای این پروژه هر دو روش بررسی شد و چون تصاویر فیلم تست حاوی عکس کتاب های دیگر بود و تصاویر شلوغی به حساب می اید دقت نکته اساسی به حساب می رود و پس از تست های بسیار متد surf برای این مسئله کارایی خوبی نداشت و تصاویر اشتباه را پیدا میکرد در صورتی که sift عملکرد به مراتب بهتری داشت و مجبور شدیم جهت دقت سرعت کمتر را تحمل کنیم تا در کل به نتایج بهتری برسیم

* پست الکترونیک نویسنده مسئول: ...@..
1. استادیار، دانشکده مهندسی عمران، دانشگاه سمنان
2. استادیار، دانشکده مهندسی عمران، دانشگاه سمنان
3. استادیار، دانشکده مهندسی عمران، دانشگاه سمنان

3-فاز 2

در این فاز به تشخیص کتاب اصلی که در ویدیو تست امده می پردازیم. برای این کار ابتدا تمام عکس ها و ویدیو ها را لود میکنیم سپس keypoint تمام تصاویر کتاب ها را بدست می اوریم.

حال باید با ویدیو مقایسه کنیم. چندین راه پیش رو داریم

1) مقایسه عکس کتاب ها با تمام فریم های ویدیو:

این روش با اینکه دقت بسیار خوبی دارد ولی مدت زمان بسیار زیادی طول میکشد و عملا کار اضافی زیادی انجام میدهد

2) مقایسه عکس با یک فریم کتاب:

این روش سرعت بسیار بالایی دارد ولی مسئله بزرگی وجود دارد" کدام فریم کتاب دارای بهترین شکل از کتاب است" برای تشخیص این موضوع یک ناظر انسانی نیاز است که فریم مناسب را مشخص کند و نمیتوانیم با اطمینان بگوییم همیشه فریم اول ، اخر یا وسط حاوی تصویر مناسبی از کتاب است و این مسئله برای هر ویدیو فرق میکند پس همین مشکل باعث می شود این روش روش خوبی نباشد

3) مقایسه تعدادی از فریم ها با کتاب ها:

من این روش که بهترین روش نیز می باشد استفاده کردیم چندین فریم را از نقاط مختلف ویدیو میگیریم سپس همه کتاب ها را با ان ها چک کرده و در نهایت از فاصله میان هر کتاب با فریم های مختلف میانگین میگیریم و کتابی که کمترین فاصله را داشته باشد کتاب مورد نظر است

با این روش سرعت نسبت به مورد اول افزایش می یابد

انتخاب تعداد فریم های انتخابی تجربی است و پس از تست های زیاد من به 40 فریم رسیدم که با دقت خوبی کتاب را تشخیص می دهد

4-فاز 3

در این فاز به ترکیب تریلر فیلم و ویدیو تست داده شده می پردازیم.

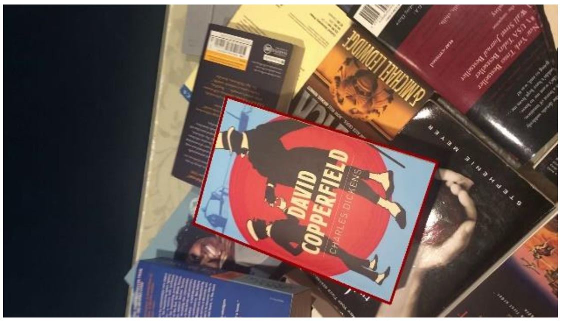
برای این کار از کتابخانه opencv استفاده میکنیم

ابتدا هر فریم ویدیو تست را میگیریم سپس کتاب را در فریم ویدیو تست پیدا میکنیم و مختصات ان را به عنوان خروجی برمیگردانیم سپس با اسفاده از تابع findHomography در کتابخانه cv دو تصویر را برهم منطبق میکنیم خروجی این تابع یک ماتریس 3 در 3 است که هر نقطه در تصویر 1 را به تصویر 2 رجیستر میکند

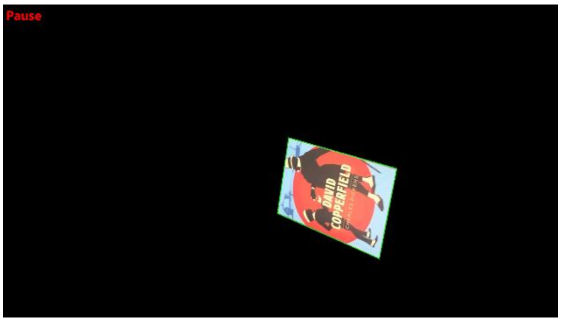
سپس با استفاده از طابع warpperspective فریم موجود در تریلر را به فرم کتاب موجود در فریم ویدیو تست تبدیل میکنیم. یک ماسک درست میکنیم از جایی از ویدیو که تریلر باید در ان قرار داده شود سپس با استقاده از توابع منطقی ان را به فریم تست اضافه میکنیم

نتایج)

ویدیو تست 1)

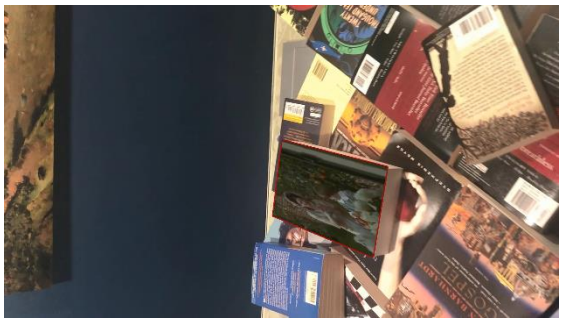


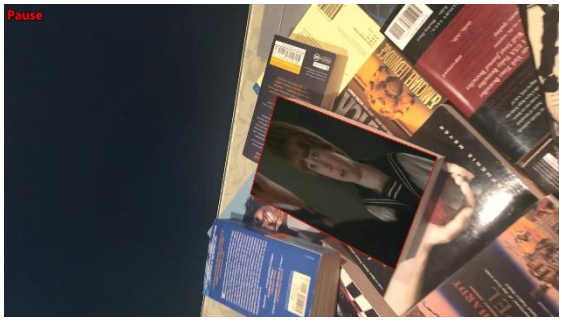
دور کتاب خط برای نمایش بهتر خط کشیده شده است

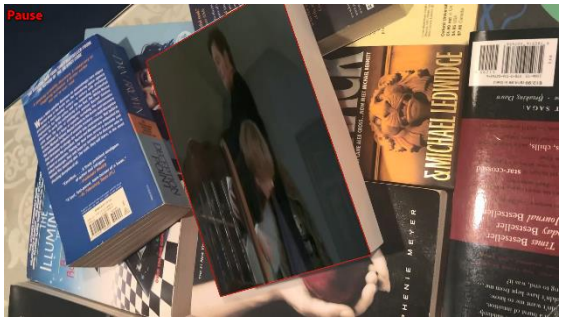


کتاب تشخیص داده شده و اماده قرار گیری فریم تریلر بر ان است

فریم های تریلر بر روی فریم فیلم اصلی قرار میگیرد و با بقیه تصویر جمع می شود

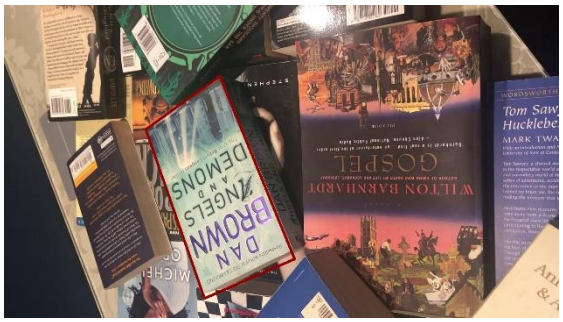




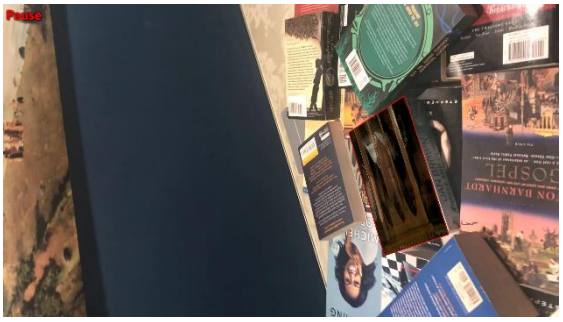


فریم های تریلر روی فریم های فیلم قرار گرفته شده است

ویدیو تست 2)



کتاب تشخیص داده شده



فریم فیلم روی فریم تست قرار گرفته اس

کدها

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
```

لود کردن کتابخانه های مورد نیاز

```
books = []
trailers = []
books.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipped
Files)/Twilight - Eclipse.jpg'))
trailers.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipp
ed Files)/Twilight - Eclipse.mp4'))
books.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipped
Files)/To Kill a Mockingbird.jpg'))
trailers.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipp
ed Files)/To Kill a Mockingbird.mp4'))
books.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipped
Files)/David Copperfield.jpg'))
trailers.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipp
ed Files)/David Copperfield.mp4'))
books.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipped
Files)/Anne of Green Gables & Anne of Avonlea.jpg'))
trailers.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipp
ed Files)/Anne of Green Gables & Anne of Avonlea.mp4'))
books.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipped
Files)/Angels and Demons.jpg'))
trailers.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipp
ed Files)/Angels and Demons.mp4'))
books.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipped
Files)/Dracula.jpg'))
trailers.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipp
ed Files)/Dracula.mp4'))
books.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipped
Files)/Pickwick Papers.jpg'))
trailers.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipp
ed Files)/Pickwick Papers.mp4'))
books.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipped
Files)/Tom Sawyer and Huckleberry Finn.jpg'))
trailers.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipp
ed Files)/Tom Sawyer and Huckleberry Finn.mp4'))
books.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipped
Files)/Twilight - New Moon.jpeg'))
trailers.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipp
ed Files)/Twilight - New Moon.mp4'))
```

```
books.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipped
Files)/Twenty Thousand Leagues Under the Sea.jpg'))
trailers.append(cv2.imread('/content/drive/MyDrive/data (1).zip (Unzipp
ed Files)/Twenty Thousand Leagues Under the Sea.mp4'))
```

لود کردن عکس ها و تریلرها

```
video1 = cv2.VideoCapture('/content/drive/MyDrive/data (1).zip (Unzippe
d Files)/Test/0.MOV')
tests = []
n = 0
while success:
  success,image = video1.read()
  if n % 30 == 0:
   tests.append(image)
  n+=1
```

20 تا 30 فریم برای تشخیص کتاب انتخاب می شود

```
def find_key_and_des(img):
  sift = cv2.xfeatures2d.SIFT_create()
  gray = rgb2gray(img)
  keypoints, descriptors = sift.detectAndCompute(img, None)
  return keypoints, descriptors
```

تابع برای پیدا کردن نقاط کلیدی و دسکریپتور

```
d=[]
for i in range(len(books)):
  key_book , des_book = findfeatures(books[i])
  d.append(0)
  for j in range(len(tests)):
    key_test , des_test = findfeatures(tests[j])
    bfm = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
    matches = bfm.match(des_book,des_test)
    d[i]+=matches[(i*len(books))+j].distance

min_d = d.index(min(d))
finalbook = books[min_d]
finaltrailer= trailers[min_d]
```

پیدا کردن بهترین کتاب که کمترین فاصله را دارد

```
def find_book(book, frame, movie_frame):
    detector=cv2.xfeatures2d.SIFT_create()
    FLANN_INDEX_KDITREE=0
    flannParam=dict(algorithm=FLANN_INDEX_KDITREE,tree=5)
```

```
    flann=cv2.FlannBasedMatcher(flannParam,{})
    trainKP,trainDesc = findfeatures(book)
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray_frame = cv2.normalize(gray_frame, None, 0, 255, cv2.NORM_MINMA
X).astype('uint8')
    queryKP,queryDesc=detector.detectAndCompute(gray_frame,None)
    matches=flann.knnMatch(queryDesc,trainDesc,k=2)
    goodMatch=[]
    for m,n in matches:
        if(m.distance<0.8*n.distance):
            goodMatch.append(m)
    if(len(goodMatch)>19):
        tp=[]
        qp=[]
        for m in goodMatch:
            tp.append(trainKP[m.trainIdx].pt)
            qp.append(queryKP[m.queryIdx].pt)
        tp,qp=np.float32((tp,qp))
        H,status=cv2.findHomography(tp,qp,cv2.RANSAC,.0)
        h,w,c=book.shape
        book_Border=np.float32([[[0,0],[0,h-1],[w-1,h-1],[w-1,0]]])
        frame_Border=cv2.perspectiveTransform(book_Border,H)
        is_closed=True
        cv2.polylines(frame,[np.int32(frame_Border)],is_closed,(0,0,150
),7)
        frame_Border_normal = np.array([[np.int32(frame_Border[0][0])],
 [np.int32(frame_Border[0][1])], [np.int32(frame_Border[0][2])], [np.in
t32(frame_Border[0][3])]])
        # cv2_imshow(frame)
        return add_trailer_frame(frame_Border_normal, movie_frame, fram
e)
```

در این تابع کتاب پیدا می شود و دور ان خط کشیده میشود و در نهایت تابعی که باید تریلر را اضافه کند صدا می زند

```
def add_trailer_frame(normal_border, movie_frame, book_frame):
        h1, w1, c = movie_frame.shape
        trailer_frame_border = np.int32([[0, 0], [0, h1 - 1], [w1 - 1,
h1 - 1], [w1 - 1, 0]])
        H, status = cv2.findHomography(trailer_frame_border, normal_bor
der, cv2.RANSAC, 5.0)
        warped_image = cv2.warpPerspective(movie_frame, H, (book_frame.
shape[1], book_frame.shape[0]))
        mask = np.zeros(book_frame.shape, dtype=np.uint8)
        mask = cv2.fillPoly(mask, [normal_border], (255,) * 3)
        mask = cv2.bitwise_not(mask)
        masked_image = cv2.bitwise_and(book_frame, mask)
        return cv2.bitwise_or(warped_image, masked_image)
```

اضافه کردن فریم تریلر به فریم اصلی

```
def main():
  video_test = cv2.VideoCapture('/content/drive/MyDrive/data (1).zip (U
nzipped Files)/Test/0.MOV')
  height, width, c = tests[0].shape
  out = cv2.VideoWriter('test.avi',cv2.VideoWriter_fourcc(*'DIVX'),34,
(width,height))
  trainKP,trainDesc = find_key_and_des(finalbook)
  count=0
  while True:
      r_book,frame_book = vidcap.read()
      if not r_book:
        print('Test video finished')
        break
      for i in range(3):
        r_movie,frame_movie = finaltrailer.read()
      if not r_movie:
        print('trailer just finished!')
        break
      video.write(find_book(finalbook, frame_book, frame_movie))
  out.release()
```

تابع اصلی که هر فریم از ویدیو تست را خوانده و 2 فریم در میان از تریلر را روی ان قرار میدهد(چون مدت زمان تریلر بیشتر از مدت زمان فیلم های تست می باشد) در کنار این فایل یکی از فیلم های تست برای بررسی نتیجه نهایی قرار داده میشود

منابع)

داکیومنت opencv

Stackoverflow.com