دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)
دانشکده مهندسی برق

پروژه کارشناسی

محمد حسین امینی ـ محمد مهدی مرادی

استاد راهنما:
دکتر محمد اعظم خسروی ـ دکتر ابوالقاسم اسداله راعی

وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ...

سوره مبارکه یوسف (ع) ـ آیه ۷۶

فهرست مطالب

۱ مقدمه ۳

۲ پردازش تصویر ورودی ۴

۱.۲ OpenCV ۴

۱.۱.۲ معرفی کلی OpenCV ۴

۲.۱.۲ تابع imread ۴

۳.۱.۲ تابع fastNlMeansDenoisingColored ۴

۴.۱.۲ تابع cvtColor ۴

۵.۱.۲ تابع adaptiveThreshold ۵

۶.۱.۲ تابع findContours ۵

۷.۱.۲ تابع drawContours ۵

۲.۲ پیش پردازش ۵

۱.۲.۲ کد های پیش پردازش تصویر ۷

۱ مقدمه

۲ پردازش تصویر ورودی

تصاویر در کامپیوتر به دو صورت ذخیره می شوند:

۱. Raster Graphics
توضیحات...

۲. Vector Graphics
توضیحات...

در دستگاه هایی همچون Printer ها به دلیل نحوه عملکرد دستگاه، تصاویر باید در قالب Raster وارد دستگاه شوند؛ اما در دستگاه هایی چون روبات های نقاش با توجه به طبیعت این روبات ها، به تصاویر Vectorized نیازمندیم. از طرفی با توجه به این که خروجی اکثر وسایل تصویربرداری از جمله دوربین های دیجیتال و همچنین تصاویر ذخیره شده در کامپیوتر بصورت Raster هستند در اولین گام به تبدیل تصاویر Raster به Vectorized پرداخته ایم.

۱.۲ OpenCV

جهت پیاده سازی الگوریتم های پردازش تصویر از ماژول OpenCV در پایتون استفاده شده است. ابتدا به معرفی کلی این ماژول می پردازیم و سپس توابع استفاده شده از آن را توضیح می دهیم.

۱.۱.۲ معرفی کلی OpenCV

توضیحات...

۲.۱.۲ تابع imread

توضیحات...

۳.۱.۲ تابع fastNlMeansDenoisingColored

توضیحات...

۴.۱.۲ تابع cvtColor

توضیحات...

۵.۱.۲ تابع adaptiveThreshold

توضیحات...

۶.۱.۲ تابع findContours

توضیحات...

۷.۱.۲ تابع drawContours

توضیحات...

۲.۲ پیش پردازش

جهت توضیحات بهتر مراحل لازم در مرحله پیش پردازش را با شکل ۱ انجام می دهیم و خروجی هر مرحله را نمایش می دهیم.
ابتدا لازم است تا لبه [1] ها و کانتور [2] های تصویر را بدست آوریم. جهت افزایش کیفیت کار، ابتدا به

شکل ۱: تصویر ورودی جهت پیش پردازش

کاهش نویز موجود در تصویر می پردازیم که خروجی آن را در شکل ۲ می بینیم. پس از مرحله کاهش نویز، جهت یافتن کانتور ها ابتدا تصویر را به فرمت سیاه و سفید [3] تبدیل می کنیم که خروجی حاصل را در شکل ۳ می بینیم.
سپس عملیات Thresholding را بر روی آن اعمال می کنیم. خروجی حاصل از این عملیات درشکل ۴ آمده است. پس از عملیات Thresholding نوبت به یافتن کانتور های تصویر می رسد.

[1] Edge
[2] Contour
[3] Grayscale

شکل ۲: خروجی حاصل از کاهش نویز تصویر

شکل ۳: خروجی حاصل از سیاه و سفید کردن تصویر

شکل ۴: خروجی حاصل از عملیات Thresholding

با انجام این عملیات نقاط متوالی مربوط به هر کانتور را در قالب یک List پایتون خواهیم داشت که جهت هدایت روبات از این نقاط استفاده خواهیم کرد.

شکل ۵: خروجی حاصل از یافتن کانتور ها

۱.۲.۲ کد های پیش پردازش تصویر

```
## Image file name...
filename='Sample6.jpg'

## Reading the image...
im = cv2.imread(filename)
page_size=[im.shape[0], im.shape[1]]
page_center=[int(page_size[0]/2),int(page_size[1]/2)]

## Denoising the image...
dst = cv2.fastNlMeansDenoisingColored(im,None,10,10,7,21)
cv2.imwrite('Denoised{}'.format(filename),dst)

## Grayscaling the image...
imgray = cv2.cvtColor(dst,cv2.COLOR_BGR2GRAY)
cv2.imwrite('Grayscale{}'.format(filename),imgray)

## Thresholding the image...
thresh = cv2.adaptiveThreshold(imgray,255,cv2.↩
    ADAPTIVE_THRESH_GAUSSIAN_C,
                                cv2.THRESH_BINARY,15,2)
cv2.imwrite('Thresholded{}'.format(filename),thresh)

## Finding the image contours...
im2, contours, hierarchy = cv2.findContours(thresh,cv2.↩
    RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

```
۲۴
۲۵ ##  Sorting contours...
۲۶ contours = sorted(contours, key=cv2.contourArea, reverse = ↩
      True)[:10]
۲۷
۲۸ imcontours=im.copy()
۲۹ cv2.drawContours(imcontours, contours, -1, (0,255,0), 3)
۳۰ cv2.imwrite('Contours{}'.format(filename),imcontours)
```