بسم الله الرحمن الرحیم

دید کامپیوتری

خانم دکتر هدی محمدزاده

تمرین سری اول کامپیوتری

امیرحسین رستمی 96101635

دانشگاه صنعتی شریف

بخش الف)

سوال یک الف)

به کمک دستور ات موجود در لایبرری openCV عکس را لود می کنیم و متن مد نظر(شماره دانشجویی) را در گوشه ی چپ تصویر می نویسیم.توجه کنید چون در سوال اشاره نشده است که گوشه بالاچپ یا پایین چپ،بنده به دلخواه گوشه ی بالا چپ را انتخاب کردم.طبق توتوریال های موجود در اینترنت که اغلب از فونت FONT_HERSHEY_COMPLEX موجود در این کتاب خانه استفاده می کنند بنده نیز از این کتاب خانه استفاده کرده و به رنگ سفید شماره دانشجویی خود را در تصویر نوشتم.در ضمن به کمک دستور cv2.putText متن درتصویر اضافه می کنم و به کمک دستور cv2.cvtColor نیز نمایش عکس را از RGB به حالت GrayScale تبدیل می کنم.(تنظیم سایز فونت نوشته به سلیقه ی خودتان است).

نکته 1:درصورت فشردن کلید s عکس های تولید شده را در مکان خود فایل پایتون ذخیره می کند.
نکته2: درصورت فشردن کلید e پنجره ی نمایشگر تصاویر بلافاصله بسته می شود.
نکته3:در صورت فشردن کلیدی به جز موارد بالا برنامه کلید ورودی را ignore می کند.

شکل یک و دو: نتایج سوال یک بخش الف

سوال یک ب)

در این سوال ابتدا به کمک دستور cv2.HoughCircles دایره های تصویر را در می آوردم،اما همانطور که می دانید و اگر امتحان کنید خواهید که ورودی های این تابع به شدت در تشخیص دایره ها موثر است و اگر بابی دقتی تعیین شوند دایره های دیگری در شکل تشخیص داده می شود لذا با بررسی و مطالعه ی داکیومنت این تابع پارامتر هارا به شکل زیر تعیین می کنیم

قطعه کد زیر توپ های فوتبال همانند توپ موجود در شکل زیر را تشخیص می دهد(توجه کنید من پس از اضافه کردن تعدادی توپ مشابه توپ این شکل در همین شکل و اجرا کردن مجدد کد این مساله را چک کردم).

```
# detect circles in the image
# special setting: in order to extract the ball.
circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1, 20, param1 = 100,
                param2 = 45, minRadius = 10, maxRadius = 40)
```

حال از آنجا که درشکل زیر فقط یک عدد توپ داریم پس از اعمال روش HoughCircles داریم که circles ما یک دایره دارد(توجه کنید هر درایه ی موجود در ماتریس circles دارای یک tuple سه تایی از x,y,r است لذا با دانستن مرکز دایره و استفاده از دستور rectangle موجود در لایبرری عه opencv به ترسیم مستطیل محیط بر دایره می پردازیم.با دانستن مرکز دایره مختصات گوشه های مستطیل به راحتی به دست می آید(در قطعه کد زیر موجود است).

```
cv2.rectangle(output, (x - r-3, y - r-3), (x + r + 3, y + r + 3), (0, 255, 0), 3)
```

توجه کنید که عدد 3 را به این جهت اضافه کردم که خیلی مستطیل به دایره نچسبد!

شکل سه(توپ محصور در مستطیل)

حال برای اضافه کردن یک توپ دیگر به شکل از اینکه الگوی چمن ها به صورت خوبی به یکدیگر نزدیک اند استفاده میکنیم، برای این امر مستطیل حاوی توپ را را extract می کنیم در جای دیگری از تصویر آن را اضافه می کنیم.(کد سمت چپ مستطیل حاوی توپ است که در جای دیگری از تصویر(ماتریس سمت چپ) تزریق می کنیم.نتیجه حاصل در ادامه آورده شده است.

```
newx = x + 10*r
newy = y
output[newy-r-3:newy+r+3,newx-r-3:newx+r+3] = image[y-r-3:y+r+3,x-r-3:x+r+3]
```

شکل چهار: توپ اضافه شده!

جواب سوال دو:

در ابتدا ویدیو قرار داده شده در googleDrive را مورد بررسی قرار می دهیم و به نتایج زیر می رسیم.

1ـ تصویر اصلی scale شده است و در سمت چپ آورده شده است.(scale هم سان(طول و عرض به یه اندازه scale شده اند)).
2ـ تصویر scale مجدد scale می شود منتها در این حالت فقط در راستای ارتفاع نصف می شود(بنده فرض کرده ام که تصویر غیر صفر سمت راست به اندازه ی نصف تصویر سمت چپ شده است)
3ـ تصویر نصف شده در مرحله ی قبل از بالا و پایین zeroExtend شده است و می دانیم که رنگ سیاه مشابه عددرنگی (0,0,0) است و سپس ماتریس های سیاه را با ماتریس verticalConcat 2 می کنیم.
4ـ حال در اثر تغییر trackBar موجود در زیر کل پنجره داریم که زاویه دوران تغییر می کند و طبق مشاهده ی انجام شده داریم که زاویه از مقدار صفر تا 360 درجه تغییر می کند.
5ـ پس از انجام دوران باید سایز تصویر دوران شده با تصویر سمت چپ(ارتفاعا) برابر باشد تا نهایتا آن ها را بایکدیگر HorizontalConCat بکنیم.
6ـ نقطه ی ثابت تصویر سمت چپ به نقطه ی متحرک تصویر سمت راست باید متصل باشد،به این شکل که در ابتدا پیکسل های ثابتی در دو تصویر چپ و راست در نظر میگیریم و سپس پیکسل سمت راست را در ماتریس دوران ضرب می کنیم و در نهایت آن هارا به یکدیگر متصل می کنیم،بدین ترتیب اتصال مدنظر برقرار باقی می ماند.

دستورهای استفاده شده:

1ـ از دستور createTrackbar جهت ترسیم trackbar استفاده می کنیم و تنظیم می کنیم که کل نوارش معادل با بازه ی 0 تا 360 درجه است.
2ـ تابعی به نام scale نوشته شده است که ماتریس ورودی را scale می کند و جهت انجام مراحل اول و دوم از آنها استفاده می کنیم.

3ـ از دستور cv2.vConCat جهت قرار دادن ستونی ماتریس سیاه در بالا و زیر ماتریس رنگی scale شده استفاده می کنیم.

4ـ هم چنین از تابع زیر جهت انجام مراحل دوران استفاده کرده ایم.

```
def rotate_image_size_corrected(image, angle):
    # Calculate max size for the rotated template and image offset
    image_size_height, image_size_width,noteUsed = image.shape
    image_center_x = image_size_width / 2
    image_center_y = image_size_height / 2

    # Create rotation matrix
    rotation_matrix = cv2.getRotationMatrix2D((image_center_x, image_center_y), angle, 0.8)
    # Apply rotation to the template
    image_rotated = cv2.warpAffine(image, rotation_matrix, (image_size_width, image_size_height))
    return [image_rotated,rotation_matrix]
```

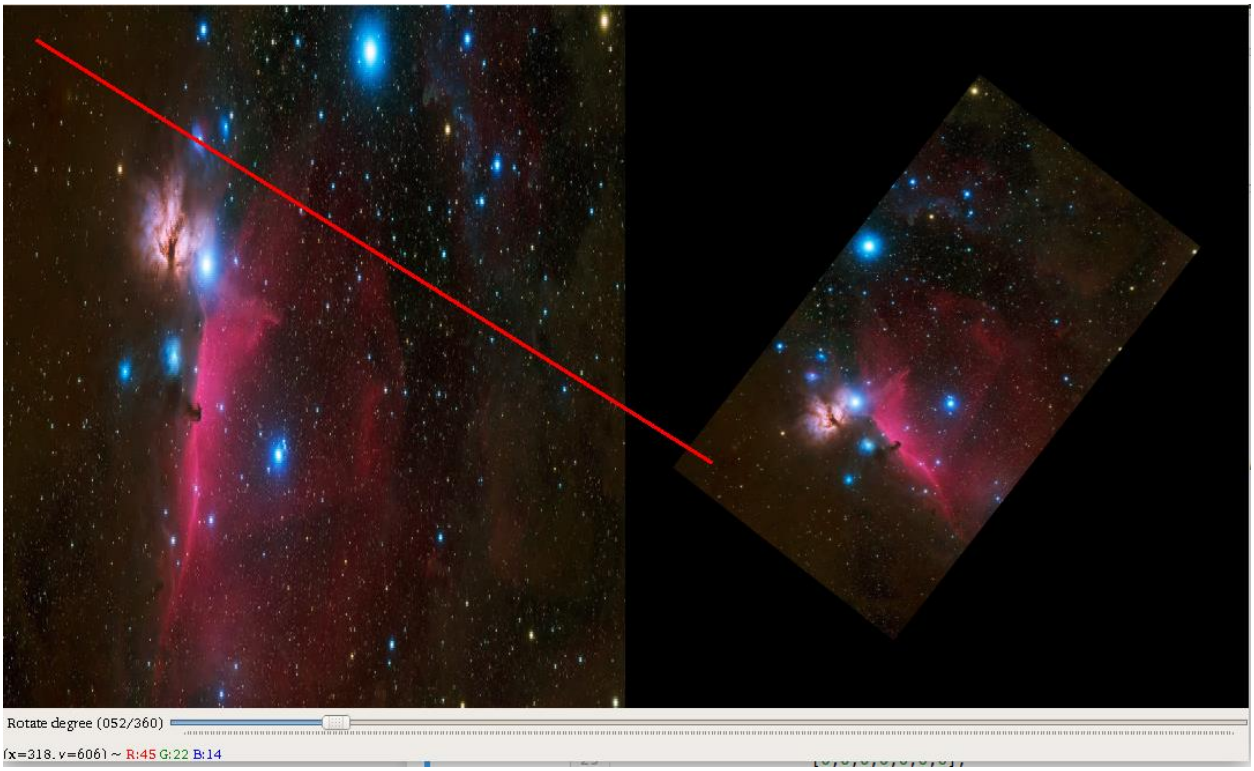
توجه کنید که این تابع،ماتریس image را به اندازه ی angle دوران می دهد.در این تابع از دستوران getRotationMatrix2D جهت به دست آوردن ماتریس دوران استفاده کرده ایم و به کمک تابع warpAffine این ماتریس دوران را بر ماتریس ورودی اعمال می کنیم.

5ـ تابع merge نوشتیم که می آید دو تا ماتریس left&right، horizontalConcat می کنیم.

6ـ در نهایت به کمک imshow به رسم تصاویر می پردازیم.

نمونه تست عملکرد:

(من بنا به دلخواه شکل سمت راست را scale ای قرار دادم که در کادر محیطش به راحتی بچرخد این ضریب به صورت hard input در کد زده شده است و به راحتی می توانید آن را تغییر دهید و شاهد نمونه تست های دیگر بشوید.



شکل پنجم: تست عملکرد

سوال 5:

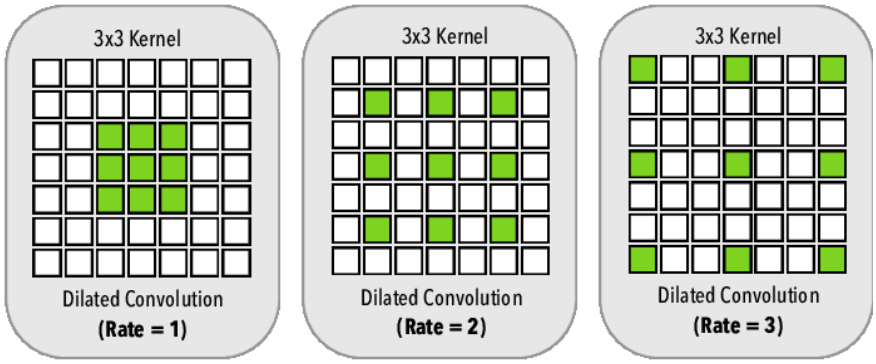
Dilation&Erosion:

طبق مطالعه های انجام شده دریافتیم که در Dilation کرنل برروی تصویر حرکت کرده و ماکسیمم مقدار خانه هایی که کرنل برروی آن قرار میگیرد را در خانه مشخص کرنل قرار می دهد،گاهی به این خانه مرکز کرنل نیز گفته می شود.و اما در حالت Erosion عملکرد دقیقا برعکس است و کمترین مقدار خانه ها در مرکز قرار میگیرند. حال چون ماکزیمم گیری حالت واگیری در محاسبه دارد(اثر کرنل رو روی خانه های مجاور فرض کنید)لذا انگار Dilation باعث پخش شدن و برعکس به علت خصلت غیرواگیری و انحصاری مینیمم(مجددا اثر کرنل را روی خانه های مجاور فرض کنید)لذا انگار Erosion باعث جمع شدن می گردد، به عبارت دیگر و به طور مثال Dilation باعث کامل شدن اتصال های محکم عکس و Erosion باعث جدا شدن اتصال های ضعیف عکس می گردد.(منظور از اتصال "تشابه رنگی مسیر بین دو پیکسل با پیکسل های دوسر مسیر" است).

قابل ذکر است که این فیلتر ها معمولا روی تصاویر باینری اجرا می شوند.

کرنل های مختلفی برای این عملیات استفاده می شود از قبیل کرنل های 3در3،5در5 و 7در7.

نکته ی مهم:کرنل ها لزوما همه ی درایه هایشان یک نیستند بلکه طبق معیاری به نام DilationRate میزان تراکم 0و1 های کرنل مشخص می گردد.



شکل 9: قسمت های سبز برابر یک میباشند

نمونه های متعددی از کرنل های مختلف(علاوه برساختار های مطرح شده در بالا) را بر روی عکس ها اعمال کردیم و حال به جای پرداختن به جزییات هر کدام از کرنل های تست شده و ذکر ساختار و ریاضی هرکدام، به بررسی اثرات آن ها روی تصاویر و ویژگی های عملی آنها می پردازیم.

توجه:جهت کار کردن با تصویر لازم است تصویر داده شده با یک ترشهولدی باینری گردد و این ترشهولد باید به گونه ای باشد که جزییات تصویر حفظ شود.

نتایج مهم پس از بررسی و اعمال کرنل های با شدت های متفاوت در عکس ها:

حال به مقایسه میپردازیم، برای کرنل های تمام یک هرچه که سایر بزرگتر می شود در Dilation بخش های سفید بیشتر میشود و در Erosion بخش های سیاه بیشتر می شوند و این رخ داد هم منطقی است چرا که درکرنل بزرگتر با احتمال بیشتری اینکه یک خانه با مقدار 1 یا صفر در آن قرار داشته باشید رخ می دهد، برای همین نیز در Dilation هرچه که کرنل را بزرگتر می کنیم بخش های سفید احاطه شده خیلی بزگتر می شوند چون درایه های سیاه احاطه کننده آنها که در همسایگی های دورتر از مرکز سفید رنگ خود هستند با کرنل به هم مرتبط می شوند و رنگ آنرا به خود می گیرند.

مثال:اگر پس از ران کردن فایل کد موجود دقت کنید ملاحظه خواهید کرد که چشم های limbo با افزایش کرنل در Dilation بزرگ و بزرگ تر میشود. از طرف دیگر می بینیم که بین گرد و غبار های پایین نیز رنگ سفید نفوذ پیدا می کند و حالت های گرد و غباری که به صورت ضعیفی به هم وصل اند از هم باز جدا تر می شوند.که قابل انتظار است.

برعکس همین امر هم در Erosion رخ می دهد و بخش های سیاه تر بزرگتر می شوند و این به این دلیل است که مینیمم گیری ذاتا عکس را به سمت متصل تر شدن بخش های سیاه سوق می دهد.حال اگر مجدد به چشم های Limbo دقت کنیم داریم که در این حالت چشمان سفید با هربار بزرگ کردن کرنل کوچک تر می شوند چون که الان خانه های وسط تر نیز با کرنل به بخش های سیاه رابطه پیدا می کنند و سیاه می گردند و به صورت کاملا برعکس قسمت قبل بخش های غباری گسترش پیدا می کند و اتصالات ضعیف را قوی تر می کند.

نکته ی مهم: ابعاد بخش های مختلف تصویر نیز در این دو روش تحت تاثیر قرار می گیرد و با تغییر سایز کرنل بزرگتر/کوچکتر می شود(بسته به فیلتر استفاده شده) و Dilation و Erosion اثر Scale عی وارون هم بر روی عکس دارند.

نکته راجع به Rate کرنل ها:با بررسی هایی که انجام شد متوجه شدم که به صورت تقریبی کرنل x&x با i Rate مشابه با کرنل (x+2i)&(x+2i) با Rate 1 دارد.دقت کنید که 3>=i>=1 است! و توجه کنید این استنتاج به صورت آماری با عکس limbo انجام شده است.

برای پرهیز از پیچیدگی و اطناب بیشتر از حد گزارش،عکس های نکته ی آخر را در گزارش نیاوردم و فقط نتیجه را ذکر کردم،مصحح به راحتی می تواند با دستکاری درایه های کرنل های موجود در کد ضمیمه شده، به نتایج مرتبط برسد.

Opening&Closing:

همانطورکه می دانید عملکردکرنل های Opening&Closing وارون یکدیگر اند و لذا در فیلتر های این بخش چون طوری انگار ما fog میگیریم! لذا کاملا منطقی است که هر یک اثر انحصاری دیگری را روی تصویر کاهش می دهد.

در تبدیل opening ما Erosion(Dilated) را محاسبه می کنیم طبیعتا منطقی است که این کار تیزی های عکس های قبل که ناشی از عملکرد تک عی این فیلتر هاست را کاهش می دهد و به اصطلاح فرکانس بالا های عکس را تاحدی حذف می کند و عکس نرم تر می گردد و چون Dilted اول می آید اتصال های ضعیف عکس ضعیف تر می گردد و هرچه اتصال ضعیف تر باشد به قطعی پس از اعمال فیلتر بالا بیشتر منتها می گردد.

در تبدیل closing ما Dilation(Erosed) را محاسبه می کنیم و از این روش جهت استخراج اتصالات بخش های مختلف تصویر با یکدیگر (یعنی اتصالات را opening تقویت می کند)و حذف کردن فرکانس های بالا استفاده می شود.

Opening = Weak Dilation & Closing = Weak Erosion

نتایج مهم پس از بررسی و اعمال کرنل های با شدت های متفاوت در عکس ها:

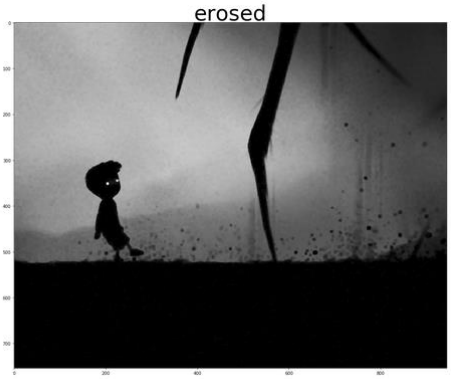
از آنجاکه این دو فیلتر در اثر کانولوشن دو فیتلر وارون(تقریبی!) هم می باشند لذا منطقی است که شکل کلی تصویر تغییر نکند و ابعاد segment های مختلف برخلاف دو روش dilation , Erosion تغییر محسوس نمی کند.

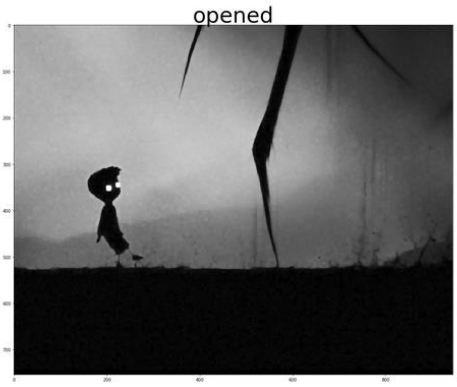
همانطورکه در توضیحات بالا مطرح شد داریم که بیشترین اثر را این دو فیلتر روی اتصالات و سوراخ ها و حوزه ای خیلی کوچک در تصویر دارند.به اثر آن ها روی بخش های "چشم" و"گرد و غبار" تصویر دقت کنید.

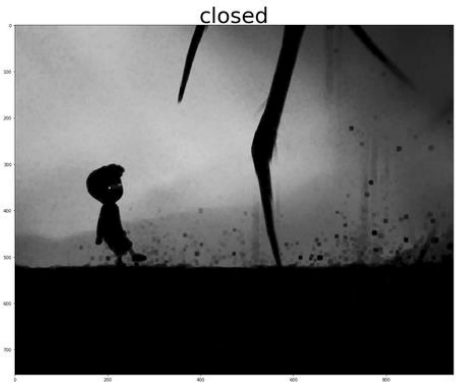
نکته ی مهم:در امتحان کردن با سایز های کم،داشتیم که اثر opening & closing ضعیف بود مخصوصا در کرنل 3 تایی،اما با افزایش سایز کرنل چشم گیرا هر فیلتر اثر خود را نشان داد،با افزایش سایز کرنل(مخصوصا در حالت 7 تایی) دیگر اثر خنثی کردن dilation&Erosion ها همدیگر را تا حد خوب خنثی نمی کنند و این باعث عملکرد بهتر opening&closing در کرنل سایز های بالاتر می شود و مشاهده کردیم که در کرنل سایز های بالا(7 مثلا) فیلتر Closing باعث تقویت هرچه بیشتر اتصالات ضعیف گردید و فیلتر Opening نیز باعث قطع کردن هرچه بیشتر اتصالات ضعیف می گردد و به اصطلاح(وجه تسمیه این فیلتر ها) که Opening اتصالات ضعیف را "باز = open" می کند و هم چنین Closing اتصالات ضعیف را "می بندد = Close" می کند یعنی متصل تر و قوی تر می کند.

تمامی نتایج در صفحات بعدی آورده شده است.

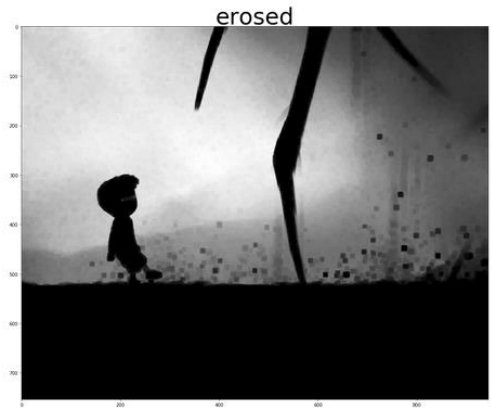
نتایج کرنل 3&3:

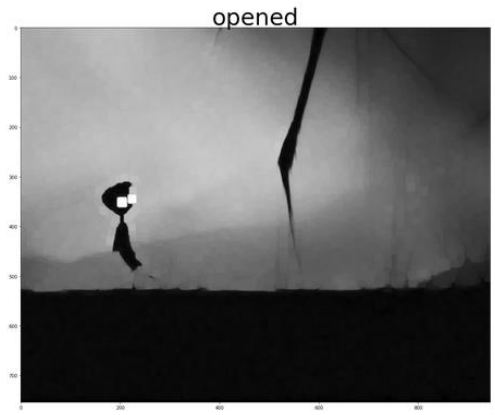


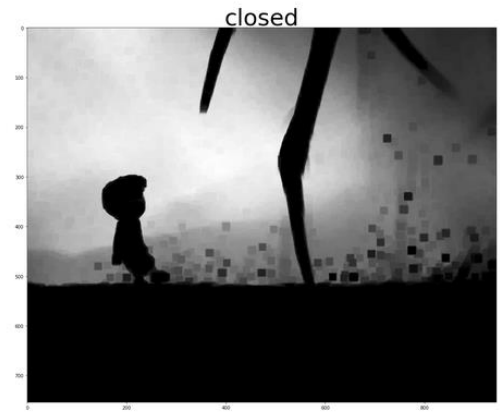




نتایج بررسی با کرنل 7&7:







جواب سوال 7:

ابتدا در نظر بگیرید که ما پترن بینی را به صورت مستطیلی که دورتادور آن 1 و داخل آن کاملا صفر است درنظر میگیریم.حال اگر این مستطیل را با تصویرمان کانوالو کنیم خواهیم دید که در نقاط مرکز هر بینی ما به نوعی انگار ما autoCorr داریم با خود فیلتر که این باعث می شود که این نقاط ماکزیمم پس از عبور از فیلتر ماکزیمم گردد اما توجه کنید به شرطی این عبارت ماکزیمم می گردد که اولا:

1ـ تصویر را به صورت GrayScale در بیاوریم.

2ـ تصویر را پس از ترشهولدی باینری کنیم(این ترشهولد به صورت تجربی از بررسی ما به دست می آید) که مثلا بنده پس از بررسی به این نتیجه رسیدم که Threshold = 80 بگیرم(توجه کنید که این باینری کردن واقعا مهم!)

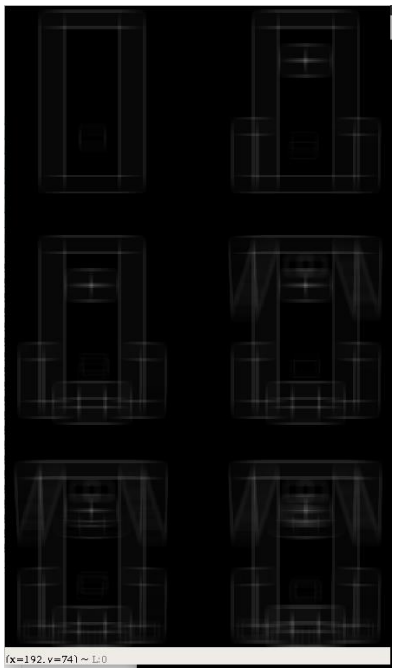
3ـ تصویر را از فیلتر بینی رد می کنیم،در این حالت داریم که نقاطی که مرکز بینی اند ماکزیمم شده اند و از لحاظ رنگی سفید تر از نقاط دیگر اند.

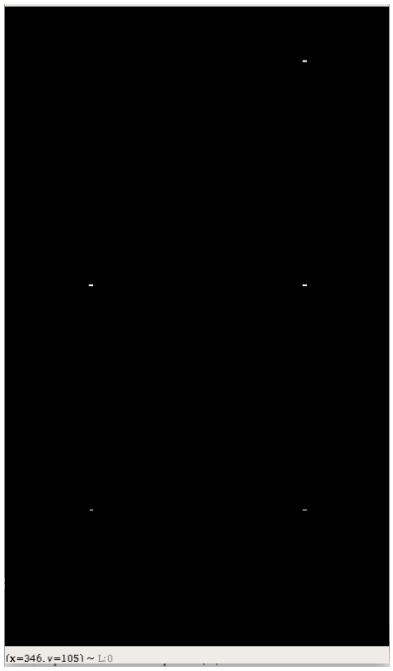
4ـ مجددا خروجی فیلتر را با تعیین ترشهولدی باینری می کنیم،توجه کنید که این مورد هم مهم است چون من به کمک برابری pixel = 255 به تعیین مرکز بینی می پردازم و توجه کنید که اگر ترشهولد از یه حداقلی کمتر باشد چون به جای مرکز بینی،خود مرکز بینی و همسایگی هایی از آن نیز در پاسخ احتساب می شوند لذا تعداد بیشتری نقطه در pixel = 255 صادق می گردند و ضخامت دایره ای که بینی را در برمیگیرد زیاد می شود!

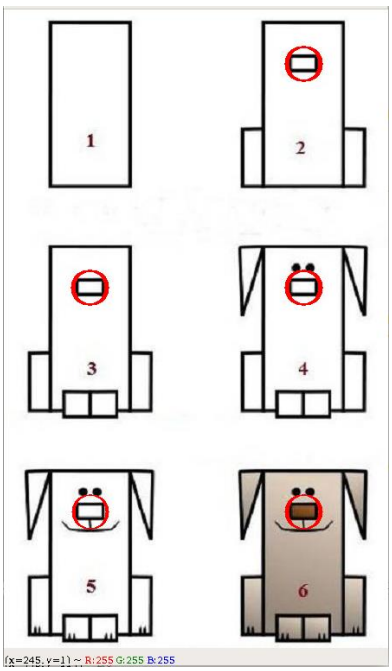
5ـ حال می آییم و حول مرکز بینی ها دایره می کشیم! تا بینی هارا مشخص کنیم.

6ـ کاری مستحب!!!: اگر خیلی حوصله ی یافتن ترشهولد دقیق نیستید!! تبدیل erosion بزنید به خروجی بخش 3 و راحتتر باینری کنید.

توجه:به ترتیب خروجی فیلتر(سمت چپ ترین)،اعمال ترشهولد تعیین NoseCenter(وسط) و خروجی نهایی بینی محصور.

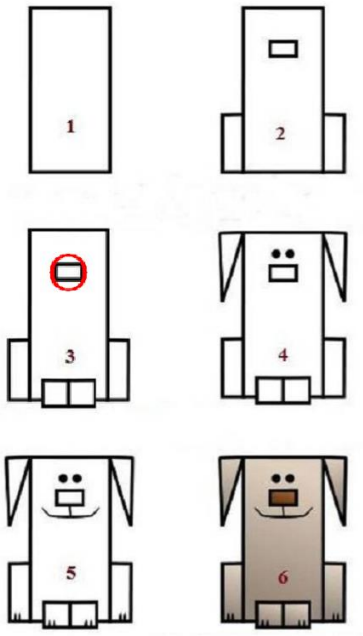






نکته:روشی که بنده برای استخراج فقط مربع خاص عه ذکر شده در صورت سوال استفاده کردم این است که بین مرکز های به دست امده در قسمت قبل آن بینی ای را انتخاب کردم که:

1 ـ کوچکترین x را دارد و از لحاظ y در دومین مرتبه قرار دارد و سپس آن را نیز جدا کردم!!!



به سادگی بر روی متغیر Centers موجود در کد که نشان دهنده ی مرکز مستطیل های داخل شکل اصلی اند جاروب می زنیم و آن مرکزی را انتخاب می کنم که کمترین x و دومین y از کوچکی را دارد و برخلاف خروجی صفحه ی قبل،فقط دور این بینی دایره می کشم و خروجی حاصله در بالا نشان داده شده است.

سوال 2 قسمت 1:

دستورات استفاده شده در این قسمت:

1ـ برای نوشتن ویدیو و ذخیره کردن frame ها از VideoWriter استفاده می کنیم.(موجود در لایبرری openCV)

2ـ برای باز کردن WebCam از دستورVideoCapture موجود در لایبرری openCV استفاده می کنیم.

3ـ حال به کمک متغیرBoolean عه record و خواندن کلید از کاربر تشخیص می دهیم که آیا شروع به ضبط کنیم یا خیر،در نظر بگیرید که اگر s وارد شود record که به صورت دیفالت false است true می گردد و طبق این متغیر عملیات ضبط کردن فریم ها انجام می شود.

نکته مهم:فایل با نام عه RecordedQB_1.avi در دایرکتوری پروژه ذخیره می گردد.

سوال 2 قسمت 2:

فیلتر Median کاری که می کند این است که داده های داده شده به آن را مرتب کرده(برحسب احتمال،تکرر...) و داده وسط را برمی گرداند، با توجه اینکه تصویر ما دارای تصویر پس زمینه ی خیابان ثابتی است و فقط ماشین ها و شکل های متحرک از آن عبور می کند لذا منطقی است که احتمال دیده شدن پیکسل های پس زمینه نسبت به پیکسل های دیگر بیشتر است لذا به کمک فیلتر median(به علت تراکم تکرار بالای پیسکل های پس زمینه و احتمال بالای دیده شدن پیکسل های آن در هر فریم) داریم که که در این ویدیو با این شرایط با احتمال زیاد گرفتن median ایده ی خوبی برای استخراج پس زمینه است.

نتیجه ی اعمال فیلتر Median بر روی فریم های این ویدیو:

همانطور که ملاحظه می کنید پیش زمینه با کیفیت خیلی بالایی! به دست آمده است!.
در ادامه روش های دیگری را جهت استخراج پیش زمینه مطرح کرده ایم.

روش های دیگر استخراج پس زمینه:

1 ـ هدف این است که وزن حضور و احتمال خانه های پس زمینه افزایش یابد لذا یک روش آن است که تکرر هر فریم را افزایش دهیم یعنی اینکه بیایم از هر فریم چند نمونه قرار بدهیم تا در نهایت احتمال حضور پس زمینه افزایش و سپس داده ی میانه(یا داده با بیشترین تکرر) را در خروجی به عنوان پس زمینه می دهیم.

2 ـ می توانیم تفاضل فریم هارا حساب کنیم و با ترشهولدی{تعیین تجربه ای ترشهولدی} از یه حدی کمتر بودن فاصله بین فریم،آن فریم هارا منتخب می کنیم و به مرحله ی بعد انتقال می دهیم و این کار را آنقدر ادامه می دهیم و ترشهولد رفته رفته تنگ تر می شود و در نهایت عکس حاصل ویژگی این را دارد که در روند هایی که ماکزییم تکرار را بر می گزیدند منتخب شده است لذا اگر پس زمینه داشته باشد عکس به احتمال بالا این خروجی اشتراک زیادی با آن خواهد داشت.

استخراج تصاویر متحرک: به سادگی هر فریم را از پس زمینه کم کرده ام و آن را در هر مرحله رسم کرده ام،به راحتی با ران کردن section آخر شاهد ویدیو حرکت اجسام متحرک خواهید بود.(فقط اجسام متحرک).

```
diff = cv2.subtract(frame,Background)
cv2.imshow('frame',diff)
```