
بسم الله الرحمن الرحيم

دانشکده فنی و مهندسی

گروه مهندسی کامپیوتر

پردازش تصویر قطعات موتوری با استفاده از روش پردازش نقطه ای

پایان نامه کارشناسی مهندسی کامپیوتر

گرایش نرم افزار

استاد راهنما:

دکتر آزاده منصوری

دکتر رویا امجدی فرد

دکتر سید اشکان موسویان

نگارش:

زهرا عبادی

بهار/تابستان ۱۴۰۰

چکیده :

در بیشتر صنایع اتومبیل در سراسر جهان ، روند بازرسی عمدتا توسط بینایی انسان انجام می شود، که ناپایدار و ناکافی است چرا که همواره با **خطا** همراه می باشد و این خطا در بسیاری از قسمت های خط تولید، می تواند تبعات مالی و اعتباری زیادی را به کارخانه خودروسازی وارد نماید. لذا وجود یک ابزار کنترلی مستقل از انسان به منظور کنترل دقیق و سریع در خط تولید خصوصاً ایستگاه های حساس و کلیدی، لازم و ضروری است. یکی از روش های کنترلی مؤثر که اخیراً در صنایع مختلف به طور گسترده مورد استفاده قرار گرفته است، روش **بینایی ماشین** است. در این روش عملیات تصمیم گیری و کنترل، برپایهٔ بررسی تصاویر گرفته شده از فرایندها صورت می گیرد. در نتیجه یکی از ارکان اصلی این روش ، فرایند **پردازش تصویر** است که از چند روش عمده و اصلی تشکیل شده است که هر یک مشتمل بر تکنیک های متعدد و بسیاری است. یکی از این روش های اصلی، پردازش نقطه ای است که شامل عملیات تغییر روشنایی تصویر، تابع روشنایی مکمل، جلوه های ویژه و تحلیل هیستوگرام شامل (استخراج، تنظیم و تعدیل آن) است.

هدف ما در این پروژه یادگیری تئوری روش **پردازش نقطه ای** و کدنویسی آن است.

کلمات کلیدی : بینایی ماشین ، پردازش تصویر ، پردازش نقطه ای ، موتورسازی.

فهرست مطالب **صفحه**

فهرست اشکال **صفحه**

فصل اول

فهرست اشکال **صفحه**

۱ تحلیل تصویر واشر سرسیلندر :

۱-۱ لبه یابی :

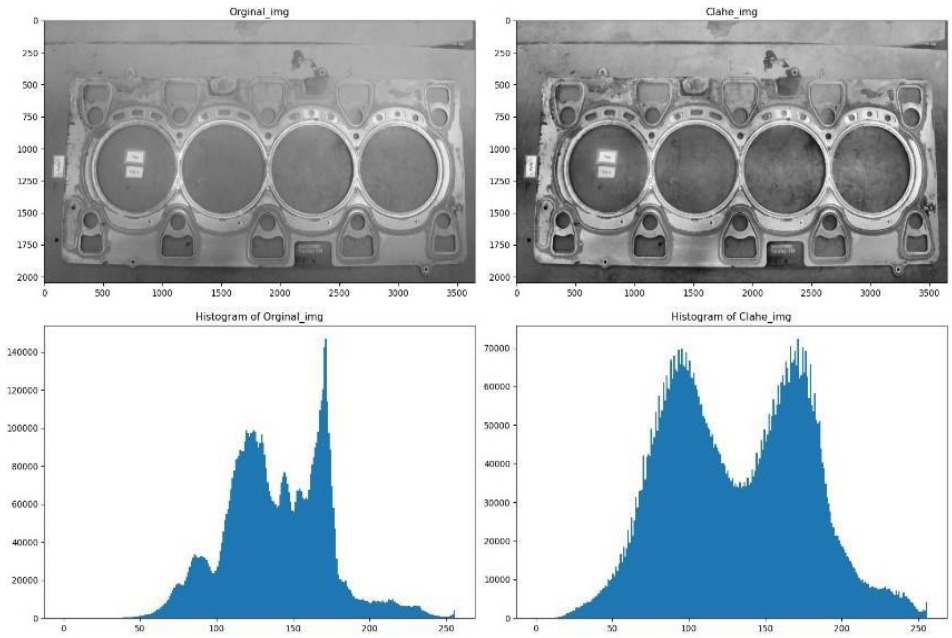
تصویر واشر سر سیلندر به صورت زیر است :

شکل ۱-۱) تصویر واشر سرسیلندر

برای تحلیل شکل ۱-۱ ابتدا روی آن عملیات همسان سازی هیستوگرام تطبیقی محدود **(CLAHE)** را که در فصل نهم این گزارش به آن پرداختیم، انجام می دهیم . تا کنتراست تصویر بهبود پیدا کند[۱].

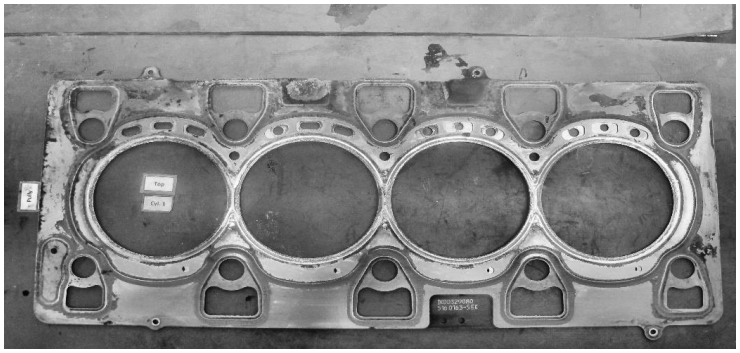
```
import cv2
from matplotlib import pyplot as plt

#khandane aks
img = cv2.imread("images/IMG_0010.jpg", 0)

#sakhtane CLAHE
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl = clahe.apply(img)

#namayeshe tasavir va histogram an ha
f, subplt = plt.subplots(2,2, figsize=(16,10))
subplt[0,0].imshow(img, cmap="gray")
subplt[0,0].set_title("Orginal img")
subplt[0,1].imshow(cl, cmap="gray")
subplt[0,1].set_title("Clahe img")
subplt[1,0].hist(img.flatten(),256,[0,256])
subplt[1,0].set_title("Histogram of Orginal img")
subplt[1,1].hist(cl.flatten(),256,[0,256])
subplt[1,1].set_title("Histogram of Clahe img")
plt.show()

#save kardan tasvir hamsansazi shode
clahe_img = cv2.imwrite('images/clahe_img.jpg', cl)
```

شکل ۱-۲) کد CLAHE



شکل ۱-۳) خروجیCLAHE

خروجی این مرحله را که تصویر همسان سازی شده است، را ذخیره می کنیم:

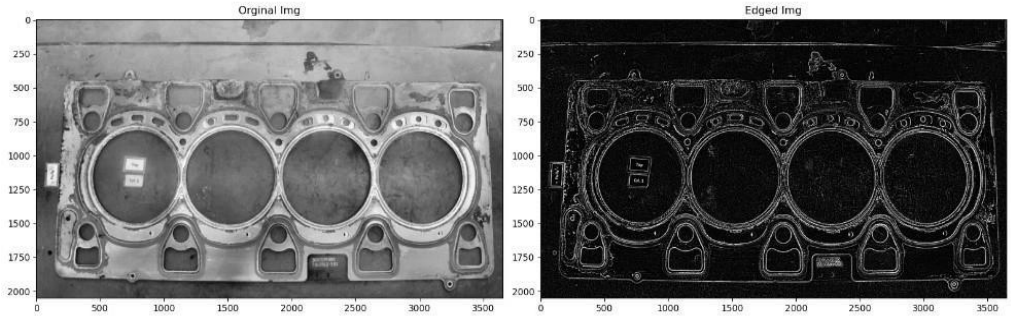


شکل ۱-۴) تصویر همسان سازی شده واشر سرسیلندر

سپس در مرحله بعد برای جداسازی قطعه واشر سرسیلندر از زمینه آن، عملیات **آشکارسازی لبه** را با استفاده از روش **LoG** بر روی شکل ۱-۴انجام می دهیم.

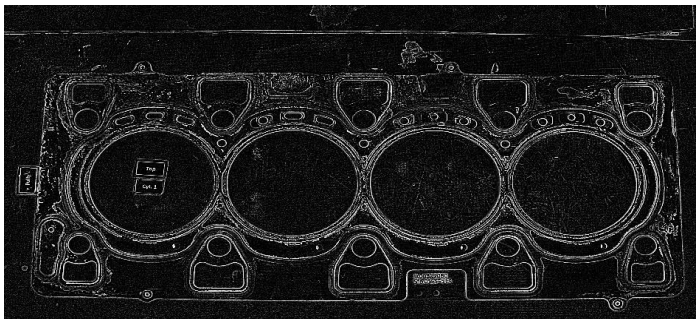
```python
import cv2
from matplotlib import pyplot as plt

#khandane aks
img = cv2.imread("images/clahe_img.jpg", 0)

#emale filtere gaussian barave kaheshe noise
gaussian = cv2.GaussianBlur(img, (11,11), 0)

#emale filtere laplacian bar roove tasvir hamvar sazi shode
LoG = cv2.Laplacian(gaussian, cv2.CV_8UC1, ksize=5)

#namayeshe tasavir
f, subplt = plt.subplots(1,2,figsize=(16,5))
subplt[0].imshow(img, cmap='gray')
subplt[0].set_title("Orginal Img")
subplt[1].imshow(LoG, cmap='gray')
subplt[1].set_title("Edged Img")
plt.show()

#save kardan tasvir labe yabi sjode
Edged_img = cv2.imwrite('images/Edged_img.jpg', LoG)
```

شکل ۱-۵) کد لبه یابی به روش LoG



شکل ۱-۵) خروجی LoG

خروجی نهایی به صورت زیر است :



شکل ۱-۶) سر سیلندر لبه یابی شده

۱-۲ اعمال الگوریتم GrabCut بر روی تصویر واشر سر سیلندر :

۱-۲-۱ مرحله اول :

```python
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

#khandane aks
img = cv.imread("images/silandr.jpg")

#ijad mask
mask = np.zeros(img.shape[:2],np.uint8)
#ijad asaraye bgModel va fgModel
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)

#ijad mostatil
rect = (10,24,494,260)

#emale grabcut
cv.grabCut(img,mask,rect,bgdModel,fgdModel,2,cv.GC_INIT_WITH_RECT)
mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
img = img*mask2[:,:,np.newaxis]

#save kardan tasvir jodasazi shode(marhaleye aval)
Siland_GrabCut1 = cv.imwrite('images/Siland_GrabCut1.jpg', img)
```

شکل ۱-۸)کد الگوریتم GrabCut (مرحله اول)

شکل ۱-۹) خروجی الگوریتم GrabCut (مرحله اول)

همان طور که در شکل ۱-۹ مشاهده می کنید، همچنان قسمتی از پس زمینه در تصویر مشاهده می شود که تشخیص داده نشده است. بنابراین لازم است به صورت دستی مناطقی را که باید حذف گردند را مشخص کنیم.

شکل ۱-۱۰)ماسک مورد نیاز برای مرحله دوم

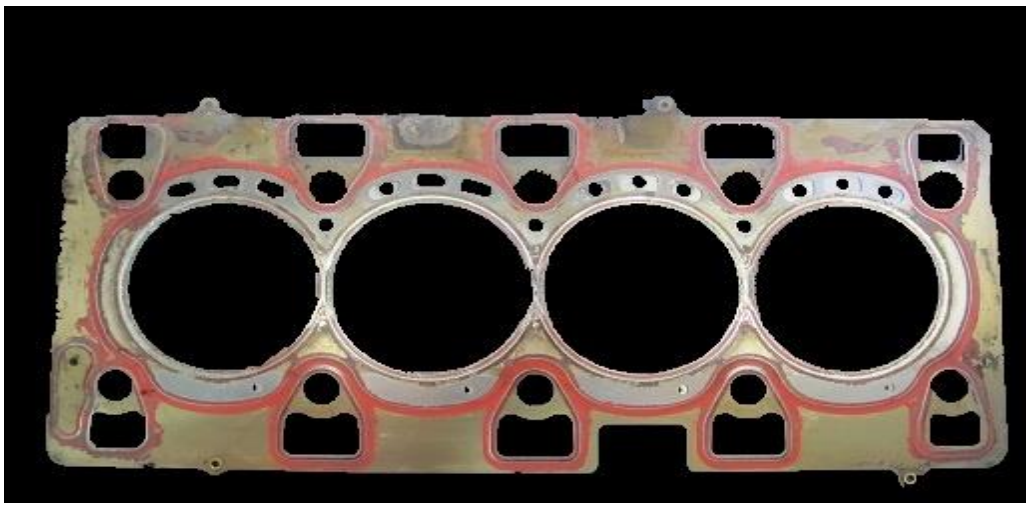
در این شکل مناطقی که با رنگ مشکی مشخص شدند، پس زمینه هستند و باید حذف گردند.

و مناطقی که با رنگ سفید مشخص شده اند، پیش زمینه هستند و باید بازیابی شوند.

۱-۲-۲ **مرحله دوم :**

```
#khandane maski ke tavasato khodeman ijad shode
newmask = cv.imread('images/mask.jpg',0)

#har ja k ba sefid moshakhas shode hatman pishzamine ast, taghir mask=1
#har ja k ba meshki moshakhas shode hatmn paszamine asr, taghir mask=0
mask[newmask == 0] = 0
mask[newmask == 255] = 1

#emale grabcut
mask, bgdModel, fgdModel = cv.grabCut(img,mask,None,bgdModel,fgdModel,5,cv.GC_INIT_WITH_MASK)
mask = np.where((mask==2)|(mask==0),0,1).astype('uint8')
img = img*mask[:,:,np.newaxis]

#namayeshe tasvir
img1 = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img1)
plt.show()

#save kardan tasvir jodasazi shode(marhaleye dovom)
Siland_GrabCut = cv.imwrite('images/Siland_GrabCut.jpg', img)
```

شکل۱-۱۱) کد الگوریتم GrabCut (مرحله دوم)



شکل۱-۱۲) خروجی الگوریتم GrabCut (مرحله دوم)

۱-۳ مراحل جداسازی واشر سر سیلندر :

در فصل قبل این جداسازی را با استفاده از الگوریتم GrabCut انجام دادم که در قسمت دوم آن بخشی از پس زمینه را که باقی مانده بود به صورت دستی و با ایجاد ماسک حذف کردم .

در این فصل می خواهیم قسمت باقی مانده را با استفاده از ایجاد ماسک به وسیله آستانه گذاری انجام دهیم .

۱-۳-۱ الگوریتم GrabCut :

در اولین مرحله روی تصویر این الگوریتم را اعمال می کنیم تا حد خوبی سر سیلندر را از پس زمینه آن جدا کند .

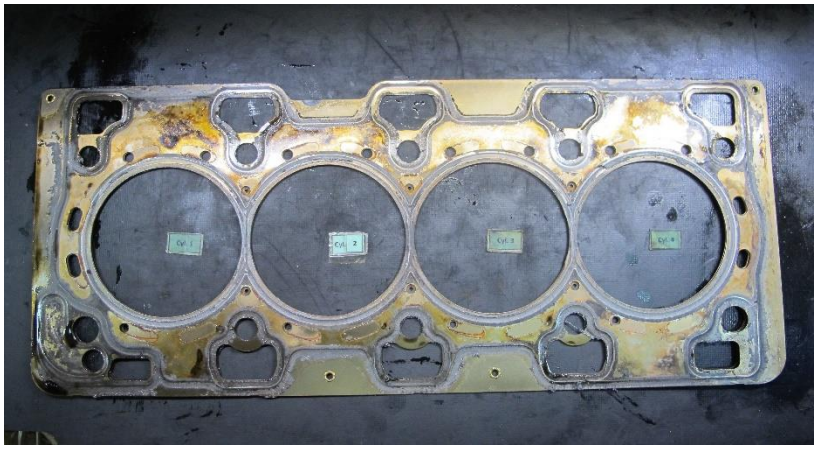
چگونگی عملکرد این الگوریتم در فصل چهاردهم این گزارش آمده است [۱].

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
#khandane aks
img = cv.imread("images/IMG_0001_resize.jpg")
#ijad mask
mask = np.zeros(img.shape[:2],np.uint8)
#ijad asaraye bgModel va fgModel
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)
#ijad mostatil
#silandr 0010 : rect = (10,24,494,260)
#silandr 0492 : rect = (3,20,545,300)
#silandr 0082 : rect = (3,20,493,255)
#silandr 2933 : rect = (3,3,505,450)
#silandr 0001 :
rect = (0,50,500,255)
#emale grabcut
cv.grabCut(img,mask,rect,bgdModel,fgdModel,2,cv.GC_INIT_WITH_RECT)
mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
img = img*mask2[:,:,np.newaxis]
#namayeshe aks
img1 = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img1)
plt.show()
```

شکل ۱-۱۳) کد الگوریتم GrabCut

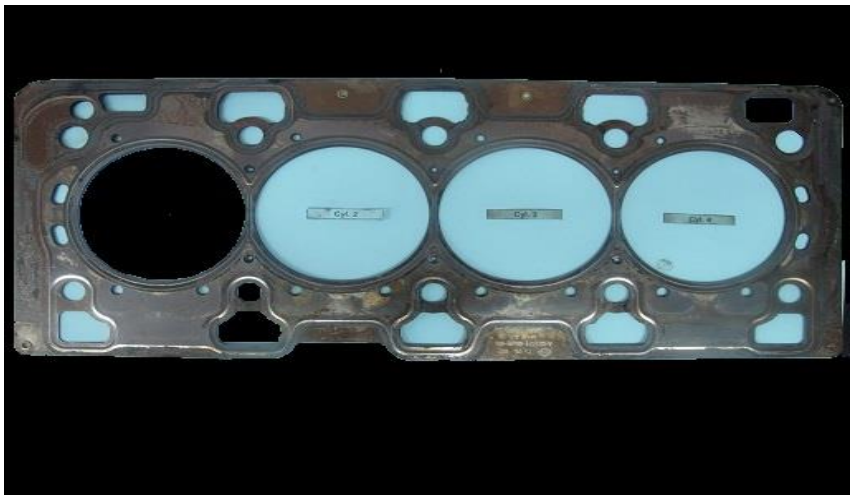
در کد شکل ۱-۱۳ مختصات مستطیلی که شامل سیلندر می باشد، متناسب با هر تصویر انتخاب شده است .

شکل ۱-۱۴) سیلندر ۰۴۹۲

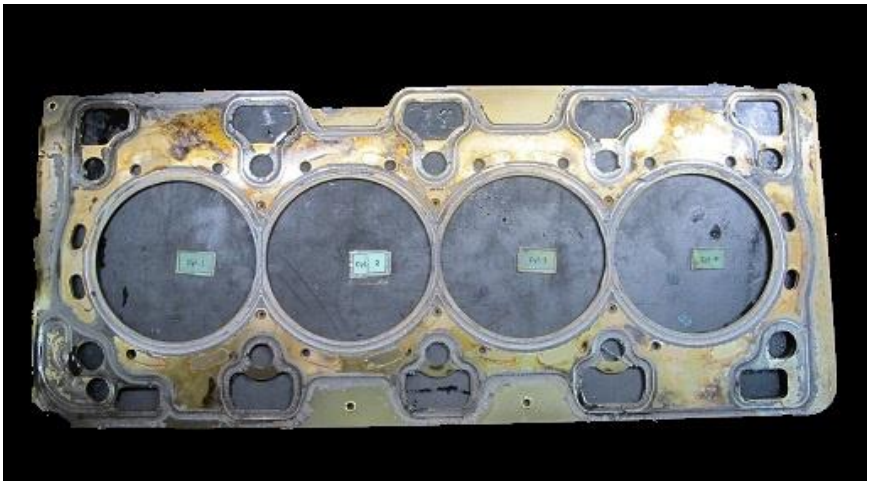


شکل ۱-۱۵) سیلندر ۲۹۳۳

حال نتیجه اعمال الگوریتم Grabcut را روی دو سیلندر بالا نمایش می دهیم :



شکل ۱-۱۶) خروجی سیلندر **۰۴۹۲** با اعمال الگوریتم GrabCut



شکل ۱-۱۷) خروجی سیلندر **۲۹۳۳** با اعمال الگوریتم GrabCut

۱-۳-۲ آستانه گذاری (**Thresholding**) :

در مرحله دوم برای این که قسمت های باقی مانده از پس زمینه نیز حذف شود ، از تابع آستانه گذاری استفاده می کنیم.

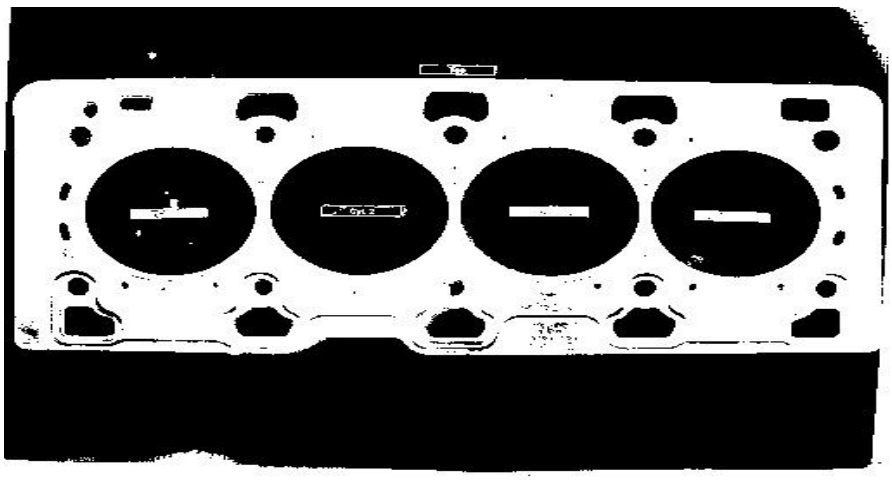
با توجه به تصویر مقدار و نوع آستانه را تعیین کرده و سپس نتیجه به دست آمده از این تابع را را بر روی تصویر به دست آمده از مرحله اول اعمال می کنیم.

توضیحات تابع آستانه گذاری در فصل پانزدهم این گزارش آمده است [۱].

```python
import cv2 as cv
from matplotlib import pyplot as plt

#khandane aksha
img_cut = cv.imread("images/SilandrCut0001.jpg")
img_cut1 = cv.cvtColor(img_cut, cv.COLOR_BGR2RGB)

img_gray = cv.imread('images/SilandrCl0001.jpg',0)

org_img = cv.imread('images/IMG_0001.JPG')
org_img1 = cv.cvtColor(org_img, cv.COLOR_BGR2RGB)

#emale threshold bar royr aksha
#silandr 0010 : ret,trs = cv.threshold(img_gray,140,255,cv.THRESH_BINARY)
#silandr 0492 : ret,trs = cv.threshold(img_gray,165,255,cv.THRESH_BINARY_INV)
#silandr 0082 : ret,trs = cv.threshold(img_gray,155,250,cv.THRESH_BINARY)
#silandr 2933 : ret,trs = cv.threshold(img_gray,163,250,cv.THRESH_BINARY)
#silandr 0001 :
ret,trs = cv.threshold(img_gray,120,250,cv.THRESH_BINARY_INV)

#emale mask threshold bar roye aks cut shode
res = cv.bitwise_and(img_cut1, img_cut1, mask=trs)
res2 = cv.cvtColor(res, cv.COLOR_BGR2RGB)
```

شکل ۱- ۱۸)کد آستانه گذاری و اعمال خروجی آن بر روی تصویر بریده شده

در کد شکل ۱-۱۸ برای هر سیلندر مقدار آستانه و نوع آستانه های متناسب با هر تصویر انتخاب شده است .



شکل ۱- ۱۹)خروجی سیلندر ۰۴۹۲ با اعمال آستانه گذاری

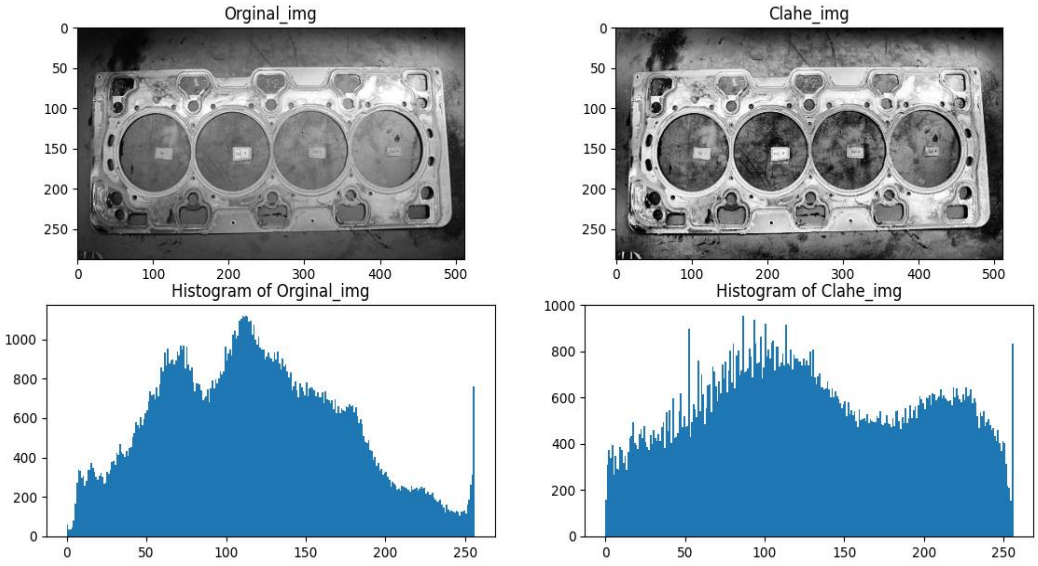
شکل ۱-۲۰) اعمال خروجی آستانه گذاری بر روی تصویر سیلندر ۰۴۹۲ بریده شده

تقاوت رنگ پس زمینه سیلندر ۲۹۳۳ با خودش کم است، پس برای این که ماسک بهتری از آستانه گذاری آن داشته باشیم، ابتدا روی آن نرمال سازی هیستوگرام تطبیقی محدود(CLAHE) را اعمال می کنیم و سپس به عنوان ورودی به تابع آستانه گذاری می دهیم .

درباره چگونگی عملکرد تابع نرمال سازی هیستوگرام تطبیقی محدود (CLAHE) در فصل نهم این گزارش آمده است[۱].

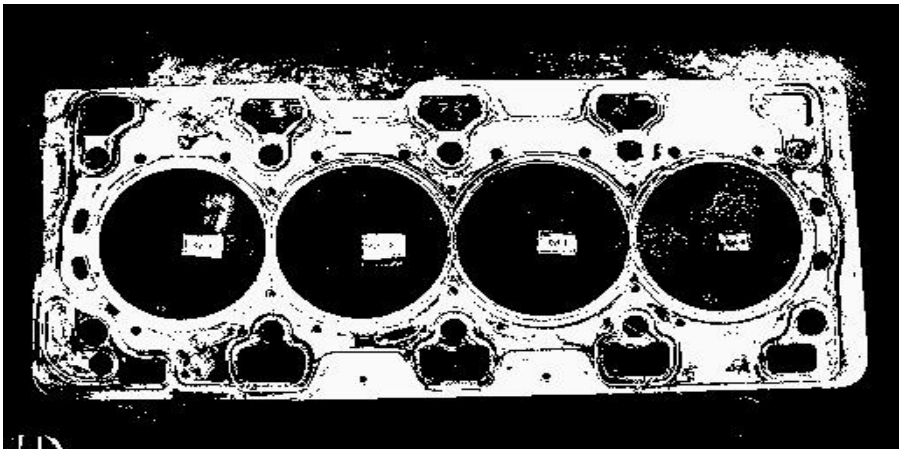
```
import cv2
from matplotlib import pyplot as plt

#khandane aks
img = cv2.imread("images/IMG_0001_resize.jpg", 0)

#sakhtane CLAHE
clahe = cv2.createCLAHE(clipLimit=70.0, tileGridSize=(2,2))
cl = clahe.apply(img)

#namayeshe tasavir va histogram an ha
f, subplt = plt.subplots(2,2, figsize=(16,10))
subplt[0,0].imshow(img, cmap="gray")
subplt[0,0].set_title("Orginal_img")
subplt[0,1].imshow(cl, cmap="gray")
subplt[0,1].set_title("Clahe_img")
subplt[1,0].hist(img.flatten(),256,[0,256])
subplt[1,0].set_title("Histogram of Orginal_img")
subplt[1,1].hist(cl.flatten(),256,[0,256])
subplt[1,1].set_title("Histogram of Clahe_img")
plt.show()

#save kardan tasvir hamsansazi shode
clahe_img = cv2.imwrite('images/SilandrCl0001.jpg', cl)
```

شکل ۱-۲۱) کد CLAHE

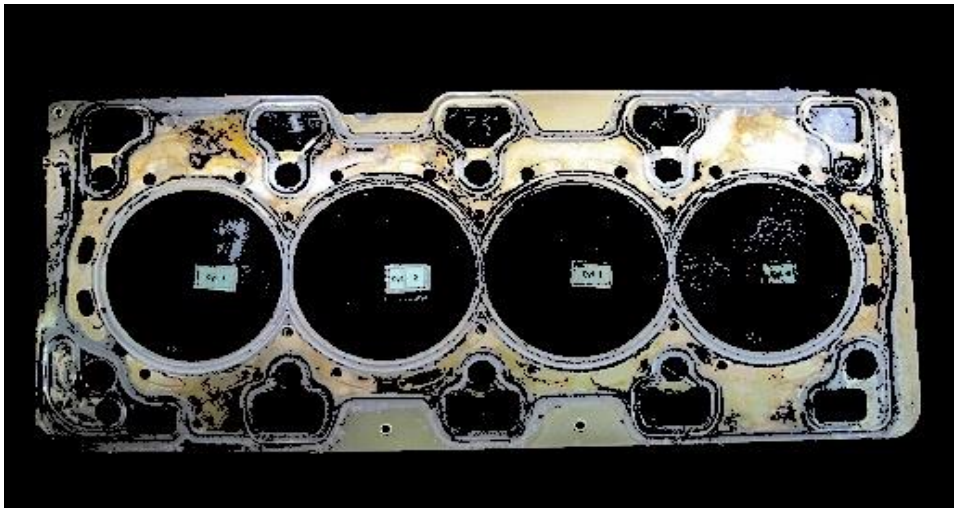


شکل ۱-۲۲) خروجی کد CLAHE بر روی سیلندر ۲۹۳۳

حال این خروجی را به عنوان ورودی به تابع آستانه گذاری می دهیم .

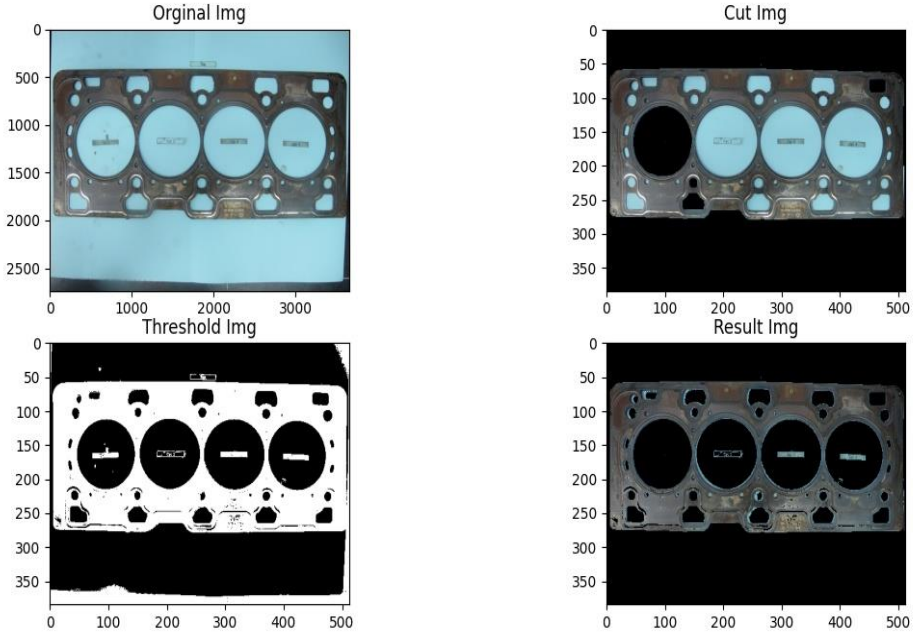


شکل ۱-۲۳) خروجی سیلندر ۲۹۳۳ با اعمال آستانه گذاری

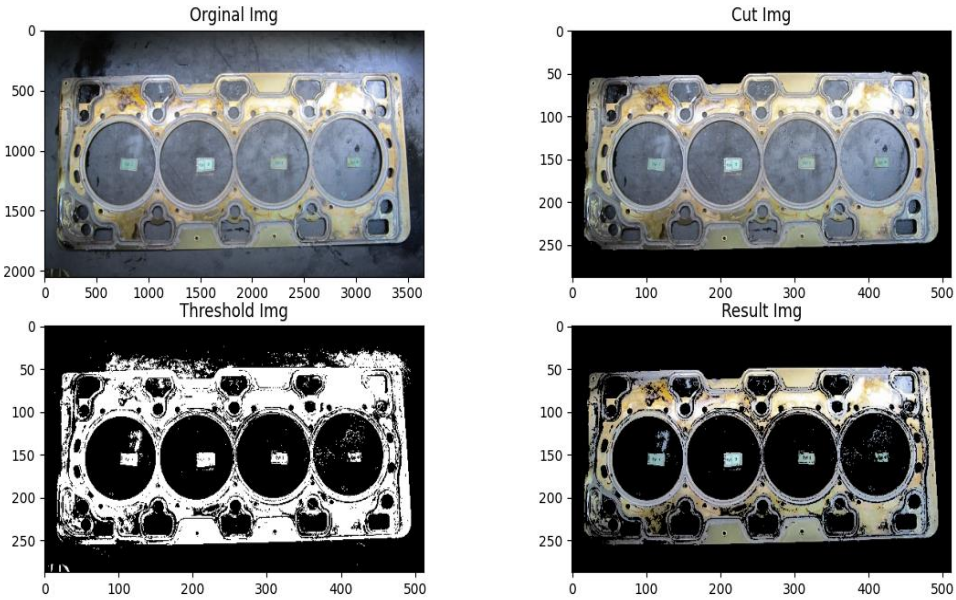


شکل ۱-۲۴) اعمال خروجی آستانه گذاری بر روی تصویر سیلندر **۲۹۳۳** بریده شده

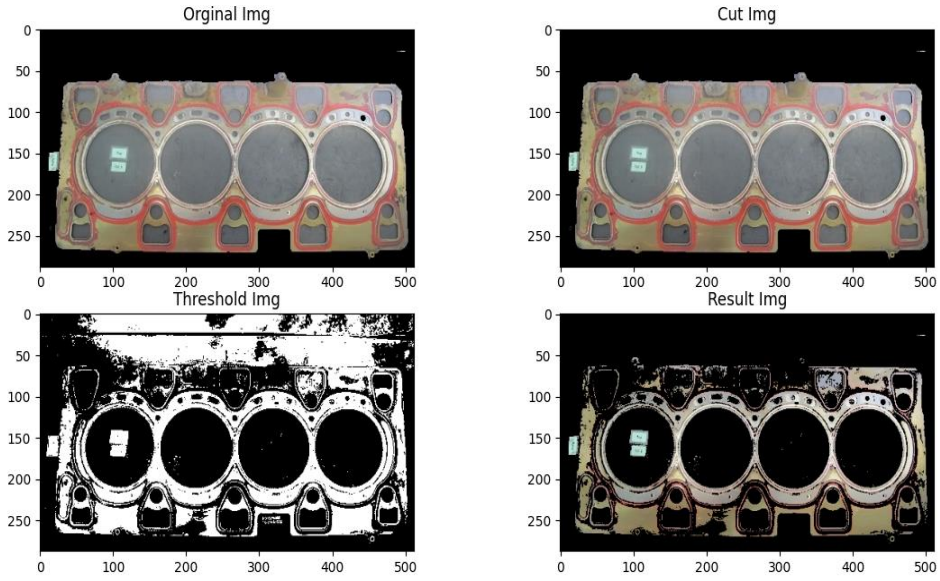
مراحل جداسازی واشر سر سیلندرها ی متفاوت را در ادامه مشاهده می کنیم :



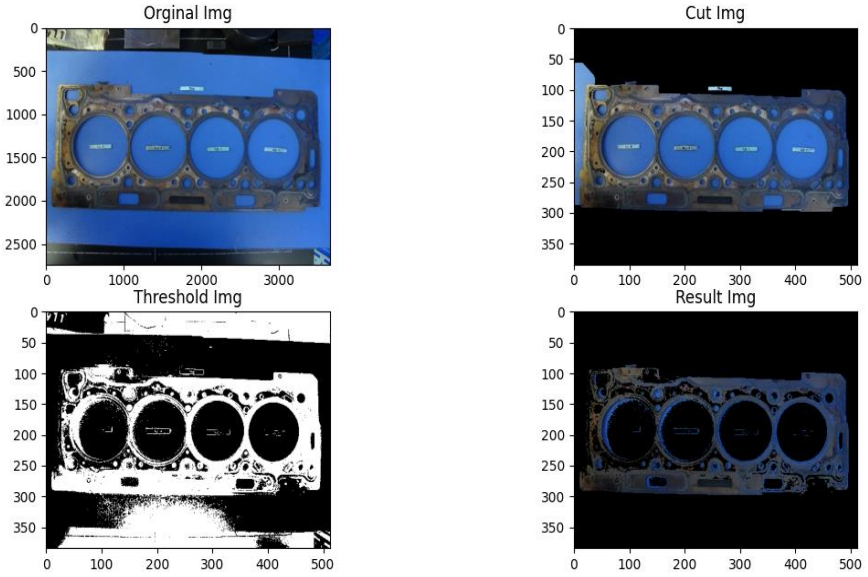
شکل ۱-۲۵) مراحل جداسازی سر سیلندر **۰۴۹۲** از پس زمینه آن



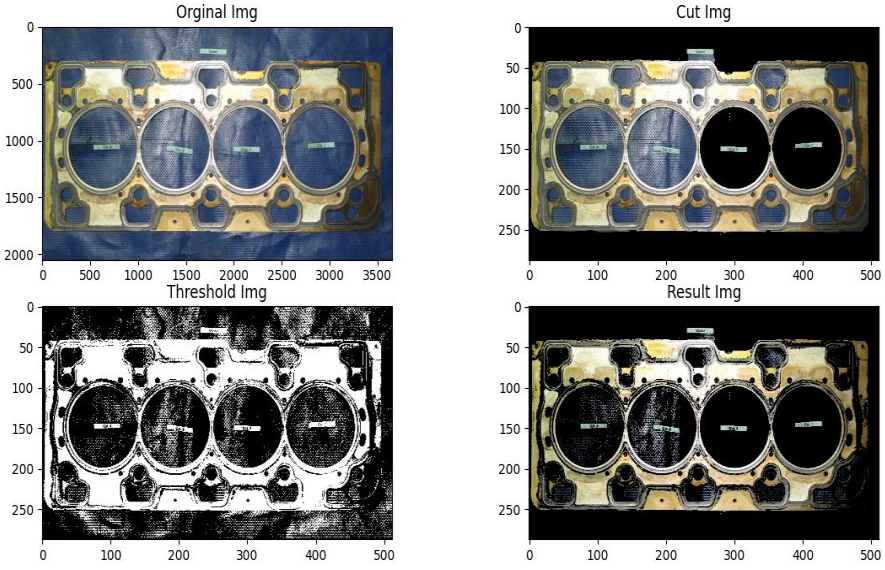
شکل ۱-۲۶) مراحل جداسازی سر سیلندر ۲۹۳۳ از پس زمینه آن



شکل ۱-۲۷) مراحل جداسازی سر سیلندر ۰۰۱۰ از پس زمینه آن



شکل ۱-۲۸) مراحل جداسازی سر سیلندر ۰۰۰۱ از پس زمینه آن



شکل ۱-۲۹) مراحل جداسازی سر سیلندر ۰۰۸۲ از پس زمینه آن

۲ نتیجه گیری :

امروزه کمتر کارخانه پیشرفته ای وجود دارد که بخشی از خط تولید آن توسط برنامه های هوشمند بینایی ماشین کنترل نشود. خطای بسیار کم، سرعت زیاد، هزینه نگهداری بسیار پایین، عدم نیاز به حضور اپراتور ۲۴ ساعته و خیلی مزایای دیگر باعث شده که صنایع و کارخانه ها به سرعت به سمت بینایی ماشین روی بیاورند. یکی از ارکان اصلی سامانه های بینایی ماشین، فرایند پردازش تصویر است. با استفاده از پردازش تصویر، اطلاعات اصلی مورد نیاز به منظور تصمیم گیری به دست می آید و دیگر ویژگی ها و اطلاعاتی که در یک تصویر خام وجود دارد، حذف می شوند. پردازش تصویر از چند روش عمده و اصلی تشکیل شده است که هر یک مشتمل بر تکنیک های متعدد و بسیاری است. یکی از این روش های اصلی، پردازش نقطه ای است که شامل عملیات تغییر روشنایی تصویر، تابع روشنایی مکمل، جلوه های ویژه و تحلیل هیستوگرام (شامل استخراج، تنظیم و تعدیل آن) است.

در این پروژه تلاش بر یادگیری تکنیک های پردازش تصویر برای استفاده در پروژه های بینایی ماشین، تئوری روش پردازش نقطه ای فراگیری شده و در عین حال اقدام به کدنویسی آن ها برای کمک به ایجاد بانک کُد پردازش تصویر قطعات موتوری و همچنین برای تحلیل و پردازش تصاویر قطعات موتوری مانند واشر سر سیلندر کرده ایم.

منابع :

۱. پروژه کارشناسی رشته کامپیوتر_گزارش نهایی _زهرا عبادی

لینک گیت هاب کد های پروژه : https://github.com/ZahraEk/Tahlil_SarSilandr