セットアップ

```
from google.colab import drive
drive.mount('/content/drive')
%cd "drive/Shareddrives/doradora-JPHacks2021/"
```

```
Mounted at /content/drive
/content/drive/Shareddrives/doradora-JPHacks2021
```

```
!pip install nbconvert
```

```
Requirement already satisfied: nbconvert in /usr/local/lib/python3.7/dist-packages (5.6.1)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.7/dist-packages (from nbc
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.7/dist-packages (from n
Requirement already satisfied: nbformat>=4.4 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-packages (from nbcon
Requirement already satisfied: testpath in /usr/local/lib/python3.7/dist-packages (from nbcon
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: bleach in /usr/local/lib/python3.7/dist-packages (from nbconve
Requirement already satisfied: jinja2>=2.4 in /usr/local/lib/python3.7/dist-packages (from nb
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from blea
Requirement already satisfied: webencodings in /usr/local/lib/python3.7/dist-packages (from b
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from ble
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7/dist-packages (fr
```

```
!!jupyter nbconvert --to=html './notebook/board_proc'
```

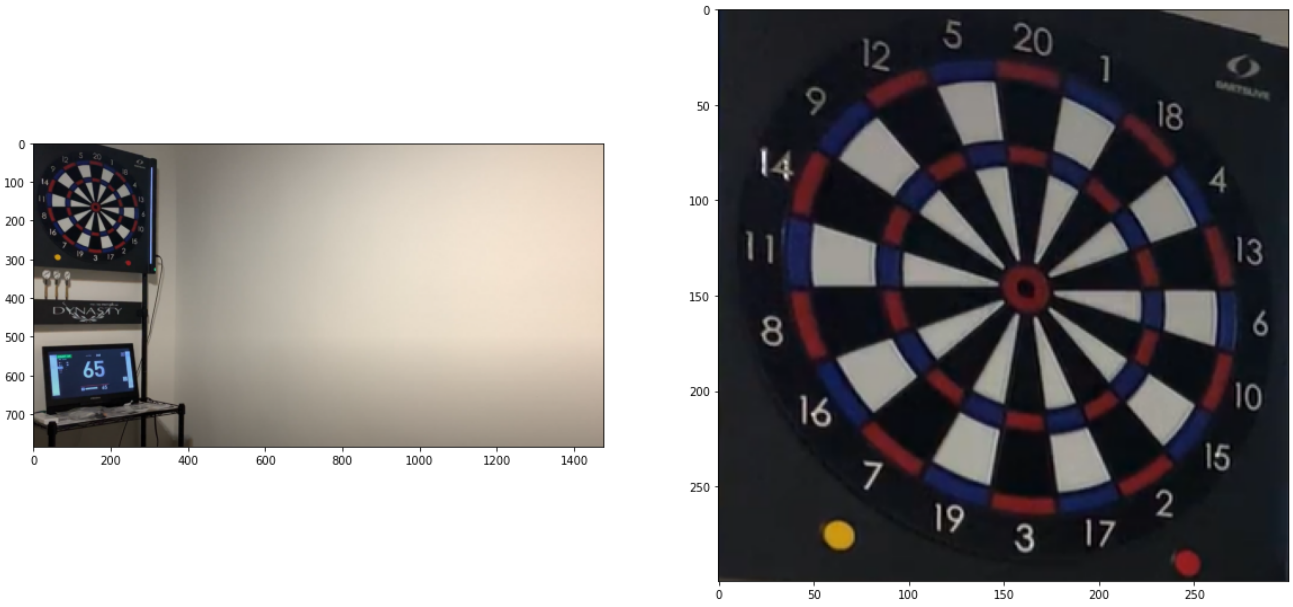
```
['[NbConvertApp] Converting notebook ./notebook/board_proc to html',
 "/usr/local/lib/python2.7/dist-packages/nbconvert/filters/datatypefilter.py:41: UserWarning:
 '  mimetypes=output.keys())',
 '[NbConvertApp] Writing 78118105 bytes to ./notebook/board_pro.html']
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
from scipy.ndimage import minimum_filter
from scipy.stats import gaussian_kde
from PIL import Image, ImageDraw
from scipy import interpolate
from sklearn.decomposition import PCA
```

```
# img = cv2.imread('video/IMG_0144_crop.png')
```

```
img_org = cv2.imread('video/IMG_0144.png')
img = img_org[20:320,:300]
img_center = img.shape[1]/2, img.shape[0]/2

plt.figure(figsize=(20,12))
plt.subplot(1,2,1)
plt.imshow(cv2.cvtColor(img_org, cv2.COLOR_BGR2RGB))
plt.subplot(1,2,2)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

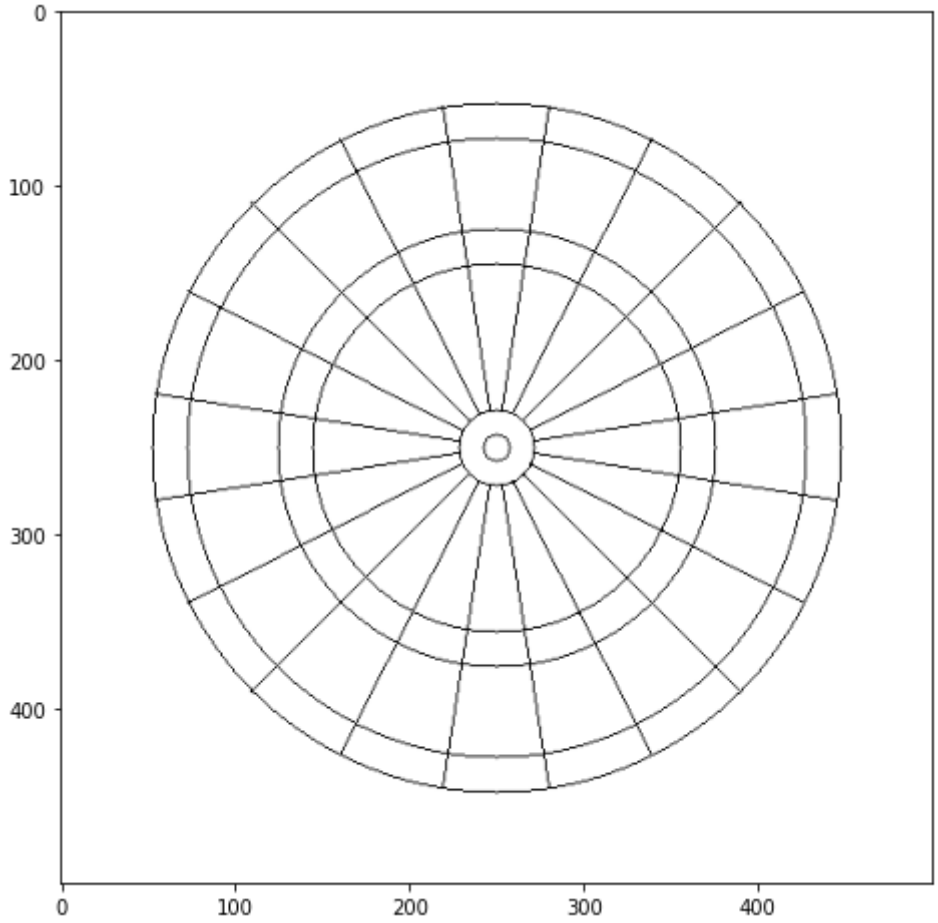
```
<matplotlib.image.AxesImage at 0x7fbdd9c11790>
```



```
coordinate_size = 500
coordinate_center = (250, 250)
coordinate_r2 = 198
coordinate_r = 178
coordinate_margine = coordinate_r/(coordinate_size/2)

img_coordinate = np.ones((coordinate_size,coordinate_size,3), np.float32)*255

cv2.circle(img_coordinate,coordinate_center,radius=8,color=(0,0,0),thickness=1)
cv2.circle(img_coordinate,coordinate_center,radius=22,color=(0,0,0),thickness=1)
cv2.circle(img_coordinate,coordinate_center,radius=106,color=(0,0,0),thickness=1)
cv2.circle(img_coordinate,coordinate_center,radius=106,color=(0,0,0),thickness=1)
cv2.circle(img_coordinate,coordinate_center,radius=126,color=(0,0,0),thickness=1)
```

```
cv2.circle(img_coordinate,coordinate_center,radius=coordinate_r,color=(0,0,0),thickness=1)
cv2.circle(img_coordinate,coordinate_center,radius=coordinate_r2,color=(0,0,0),thickness=1)

for i in range(20):
  cv2.line(img_coordinate,(int(coordinate_center[0]+22*np.cos(2*np.pi/20*i+2*np.pi/40)),int(coordin

plt.figure(figsize=(8,8))
plt.imshow(img_coordinate)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255
<matplotlib.image.AxesImage at 0x7fb1ecc55cd0>
```



正面化

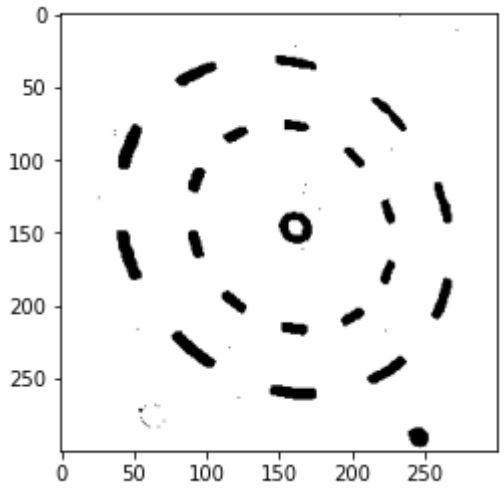
楕円近似案 (不採用)

近似楕円検出

```
img_filltered = img.copy()
# img2[:,:,0] = np.where(img2[:,:,0]<100, 255, 0)
# img2[:,:,2] = np.where(img2[:,:,2]<130, 255, 0)
# img2[:,:,1] = np.ones(img2.shape[:2])*0
```

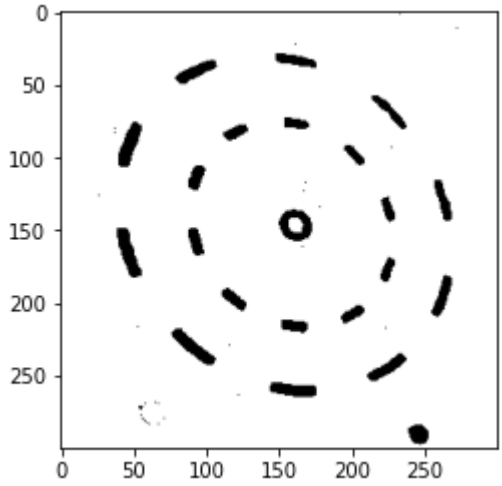
```
cond_red = (img_filltered[:,:,0]<70)&(img_filltered[:,:,1]<70)&(img_filltered[:,:,2]>70)
# cond_blue = (img_filltered[:,:,0]>90)&(img_filltered[:,:,1]<100)&(img_filltered[:,:,2]<100)
cond_blue = False
cond_red_blue = cond_red | cond_blue
# cond_black = (img_filltered[:,:,0]>120)&(img_filltered[:,:,1]>120)&(img_filltered[:,:,2]>120)

img_filltered[:,:,0] = np.where(cond_red_blue, 0, 255)
img_filltered[:,:,1] = np.where(cond_red_blue, 0, 255)
img_filltered[:,:,2] = np.where(cond_red_blue, 0, 255)

plt.imshow(cv2.cvtColor(img_filltered, cv2.COLOR_BGR2RGB))
```

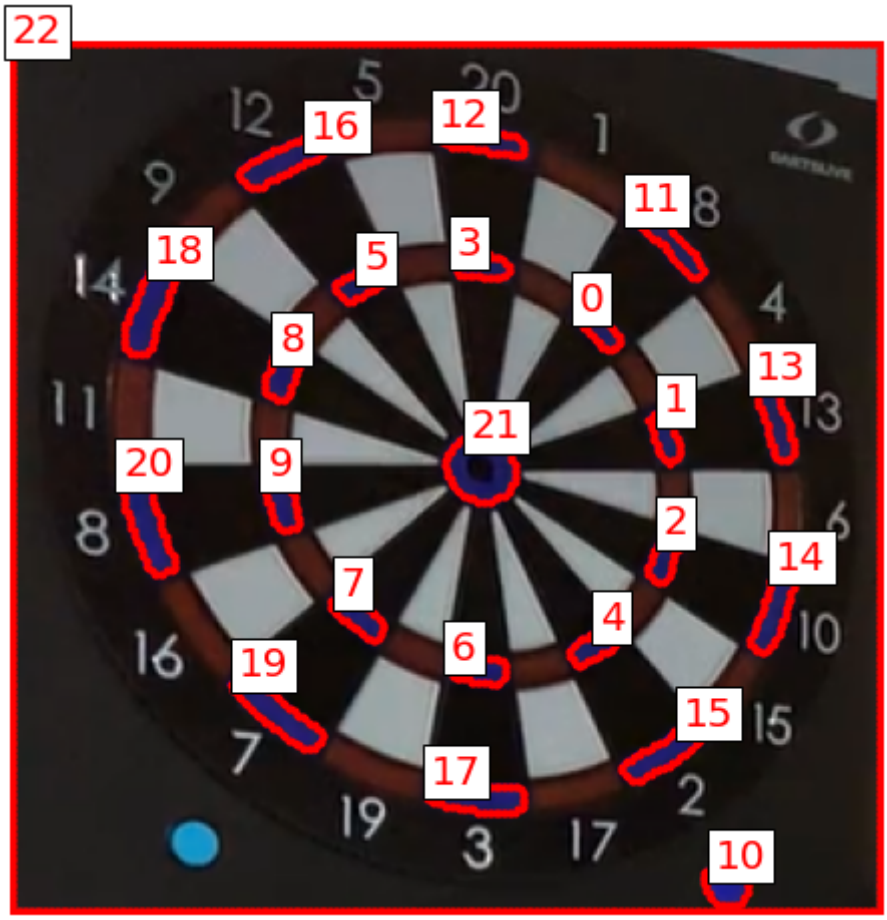
```
<matplotlib.image.AxesImage at 0x7f144020bed0>
```




```
def draw_contours(ax, img, contours):
    ax.imshow(img)
    ax.set_axis_off()

    for i, cnt in enumerate(contours):
        # 形状を変更する。(NumPoints, 1, 2) -> (NumPoints, 2)
        cnt = cnt.squeeze(axis=1)
        # 輪郭の点同士を結ぶ線を描画する。
        ax.add_patch(plt.Polygon(cnt, color="b", fill=None, lw=2))
        # 輪郭の点を描画する。
        ax.plot(cnt[:, 0], cnt[:, 1], "ro", mew=0, ms=4)
        # 輪郭の番号を描画する。
        ax.text(cnt[0][0], cnt[0][1], i, color="r", size="20", bbox=dict(fc="w"))


img_filltered_gray = cv2.bitwise_and(img_filltered[:,:,0], img_filltered[:,:,1])
img_filltered_gray = cv2.bitwise_and(img_filltered_gray, img_filltered[:,:,2])

contours, hierarchy =  cv2.findContours(img_filltered_gray, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE )
contours = list(filter(lambda x: 1e2 < cv2.contourArea(x) , contours))
contours = np.array(contours)[np.argsort([cv2.contourArea(x) for x in contours])]

plt.imshow(cv2.cvtColor(img_filltered_gray, cv2.COLOR_GRAY2RGB))
fig, ax = plt.subplots(figsize=(8, 8))
draw_contours(ax, img, contours)
plt.show()
```

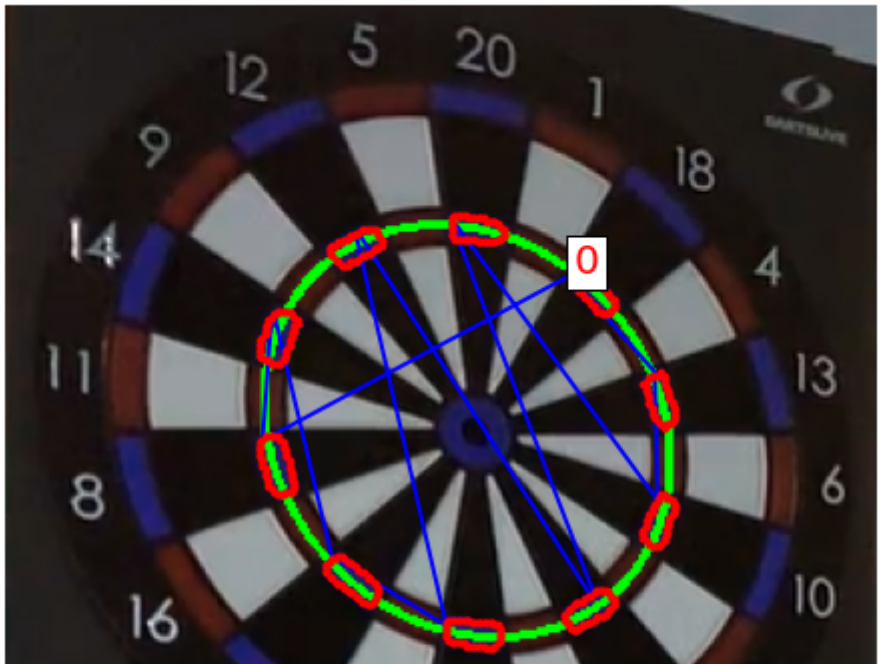
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:21: VisibleDeprecationWarning: C
```






```
merged_contours = np.vstack(contours[0:10])
ellipse = cv2.fitEllipse(merged_contours)

fig, ax = plt.subplots(figsize=(8, 8))
_ = draw_contours(ax, img, [merged_contours])
img_with_ellipse = cv2.ellipse(img.copy(), ellipse,(0,255,0),2)
plt.imshow(img_with_ellipse)
```

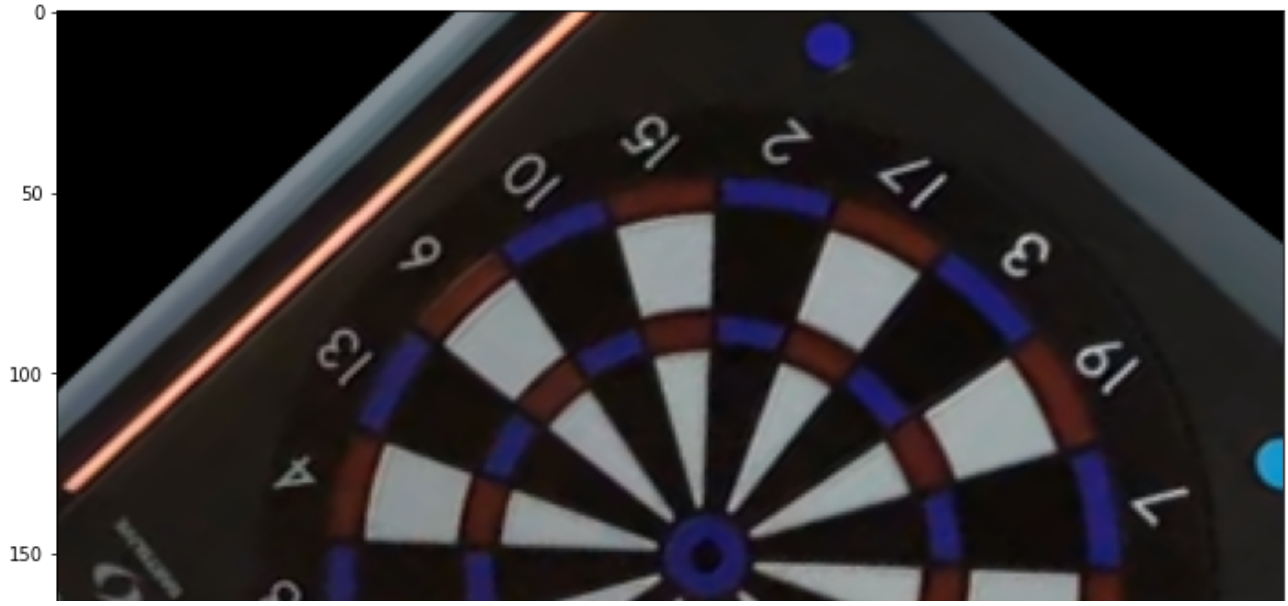
```
<matplotlib.image.AxesImage at 0x7f14400554d0>
```



正面画像に変換

```
angle = ellipse[2]
scale = ellipse[1]
scale = scale[0]/scale[1]

M = cv2.getRotationMatrix2D((img.shape[0]/2, img.shape[1]/2), angle, 1)
M[:,0:2] = np.array([[1,0],[0,scale]]) @ M[:,0:2]
M[1,2] = M[1,2] * scale
img_front = cv2.warpAffine(img, M, dsize=img.shape[:2])

fig = plt.figure(figsize=(12,12))
plt.imshow(img_front)
```

```
<matplotlib.image.AxesImage at 0x7f144ff15210>
```

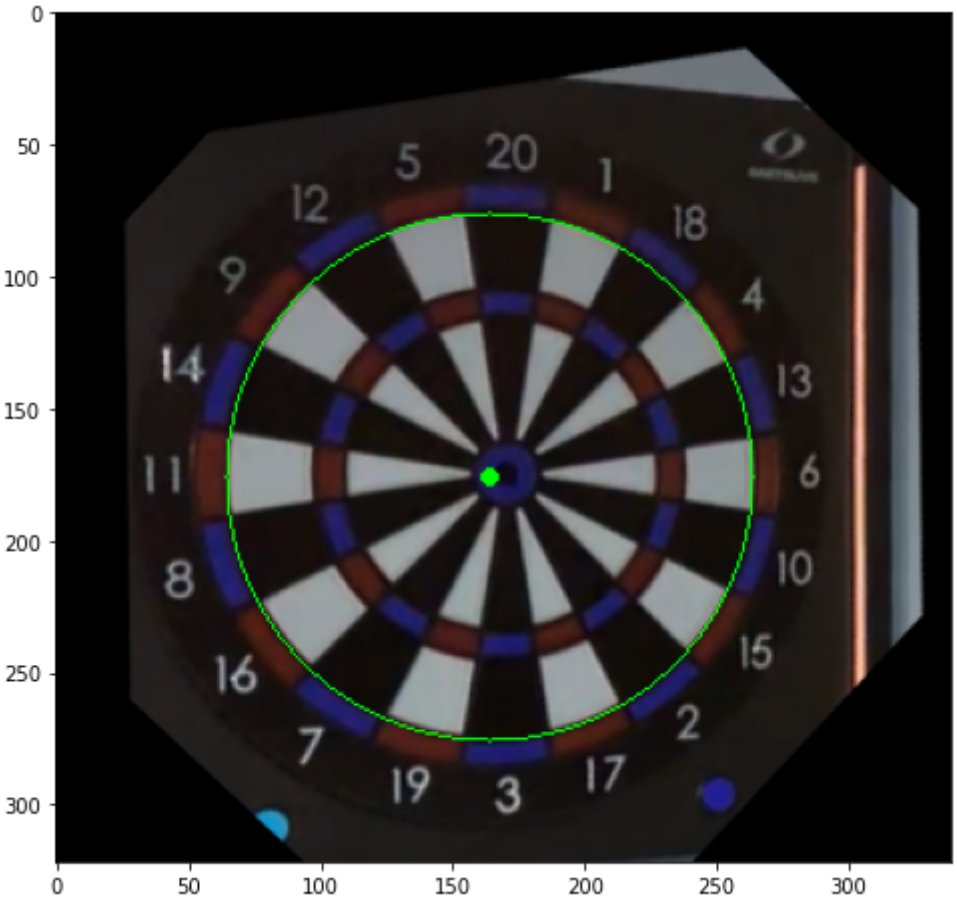



```
M = cv2.getRotationMatrix2D(center=(img.shape[0]/2, img.shape[1]/2), angle=227, scale=1)
img_front2 = cv2.warpAffine(img_front, M, img.shape[:2])

fig = plt.figure(figsize=(20,12))
plt.subplot(1,2,1)
plt.imshow(img)
plt.subplot(1,2,2)
plt.imshow(img_front2)
```

```
<matplotlib.image.AxesImage at 0x7f14506fe550>
```

```
img_front2_gray = cv2.cvtColor(img_front2, cv2.COLOR_BGR2GRAY)
circles = cv2.HoughCircles(img_front2_gray , cv2.HOUGH_GRADIENT, dp=1.05, minDist=200, param1=100,

img_front3 = img_front2.copy()
for circle in circles:
  cv2.circle(img_front3, (circle[0], circle[1]), circle[2], (0, 255, 0), 1)
  cv2.circle(img_front3, (circle[0], circle[1]), 2, (0, 255, 0), 2)
plt.figure(figsize=(8,8))
plt.imshow(img_front3)
```

```
<matplotlib.image.AxesImage at 0x7f144f184810>
```

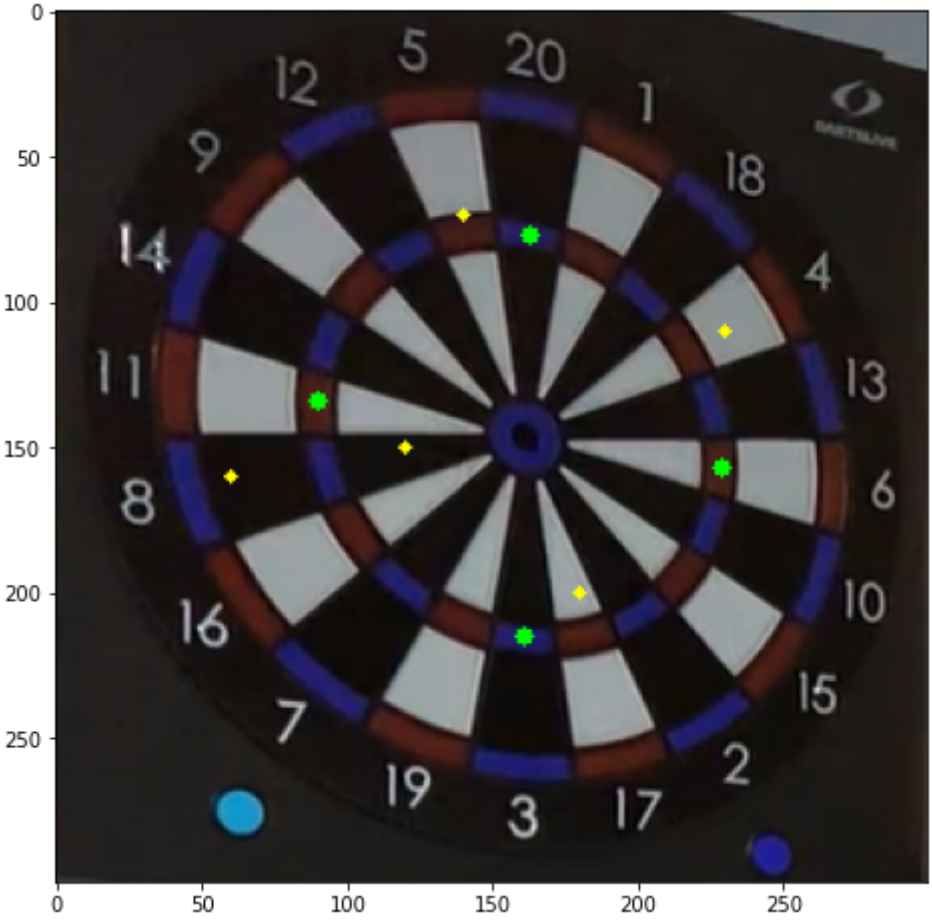


タップでマーキング案 (保留)

```
img_marked = img.copy()
marked_points = [[163, 77],[161, 215],[90, 134],[229, 157]]
for mp in marked_points:
  cv2.circle(img_marked, tuple(mp), 3, (0, 255, 0), -1)

test_points = [[180, 200], [120, 150], [60, 160], [230, 110], [140, 70]]
for tp in test_points:
  cv2.circle(img_marked, tuple(tp), 2, (255, 255, 0), -1)
```

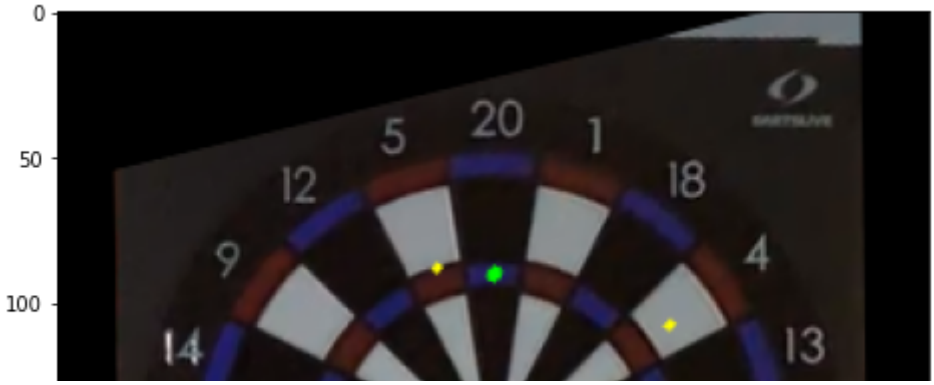
```
plt.figure(figsize=(8,8))
plt.imshow(img_marked)
```

```
<matplotlib.image.AxesImage at 0x7f333e094c10>
```




```
# 変換前後の対応点を設定
p_original = np.float32(marked_points)
p_trans = np.float32([[150,90], [150,210], [90,150], [210,150]])

# 変換マトリクスと射影変換
M1 = cv2.getPerspectiveTransform(p_original, p_trans)
img_marked_front = cv2.warpPerspective(img_marked, M1, img.shape[:2])

plt.figure(figsize=(8,8))
plt.imshow(img_marked_front)
```

```
<matplotlib.image.AxesImage at 0x7f751b24a650>
```



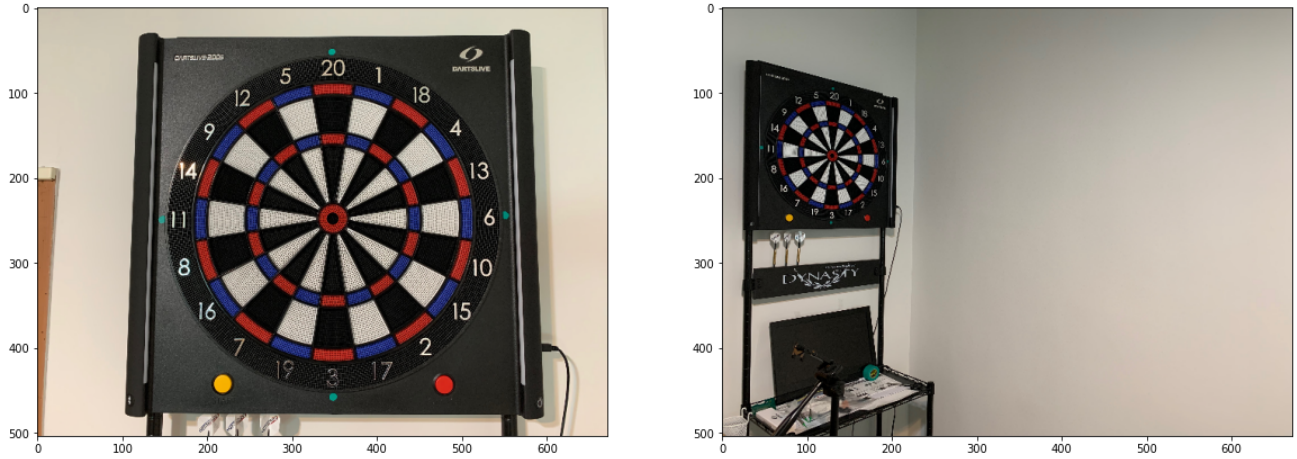

```
img_marked_front_gray = cv2.cvtColor(img_marked_front, cv2.COLOR_BGR2GRAY)
circles = cv2.HoughCircles(img_marked_front_gray , cv2.HOUGH_GRADIENT, dp=1.13, minDist=300, param1

img_marked_front2 = img_marked_front.copy()
for circle in circles:
  cv2.circle(img_marked_front2, (circle[0], circle[1]), circle[2], (0, 255, 0), 1)
  cv2.circle(img_marked_front2, (circle[0], circle[1]), 2, (0, 255, 0), 2)
plt.figure(figsize=(12,12))
plt.imshow(img_marked_front2)
```

```
<matplotlib.image.AxesImage at 0x7f751b1b5790>
```

マーキング画像案 (採用)

```
img1 = cv2.imread('video/IMG_0155.jpeg')
# img1 = cv2.imread('video/IMG_0153.png')
img2 = cv2.imread('video/IMG_0158.jpeg')
# img_center = img.shape[1]/2, img.shape[0]/2

n = 6
img1 = cv2.resize(img1, (int(img1.shape[1]/n),int(img1.shape[0]/n)))
img2 = cv2.resize(img2, (int(img2.shape[1]/n),int(img2.shape[0]/n)))

plt.figure(figsize=(20,12))
plt.subplot(1,2,1)
plt.imshow(cv2.cvtColor(img1, cv2.COLOR_BGR2RGB))
plt.subplot(1,2,2)
plt.imshow(cv2.cvtColor(img2, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f0d3b4ec6d0>
```

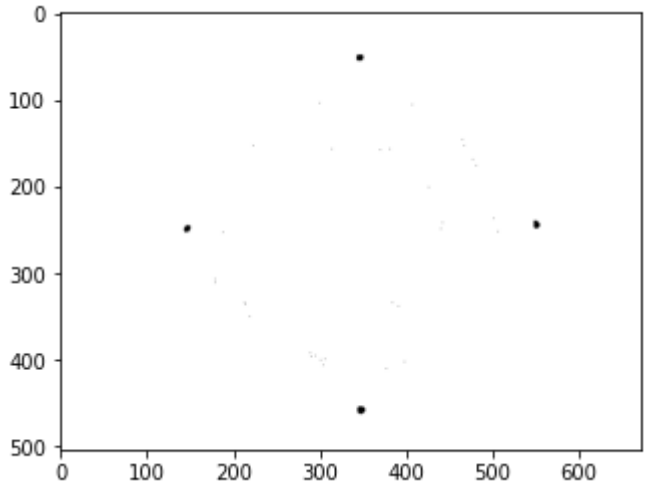


img1

```
img = img1.copy()
```

```
img_filltered = img.copy()

cond_green = (img_filltered[:,:,0]>60)&(img_filltered[:,:,1]>80)&(img_filltered[:,:,2]<60)
# cond_green = True

img_filltered[:,:,0] = np.where(cond_green, 0, 255)
img_filltered[:,:,1] = np.where(cond_green, 0, 255)
img_filltered[:,:,2] = np.where(cond_green, 0, 255)

plt.imshow(cv2.cvtColor(img_filltered, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f333df4a310>
```

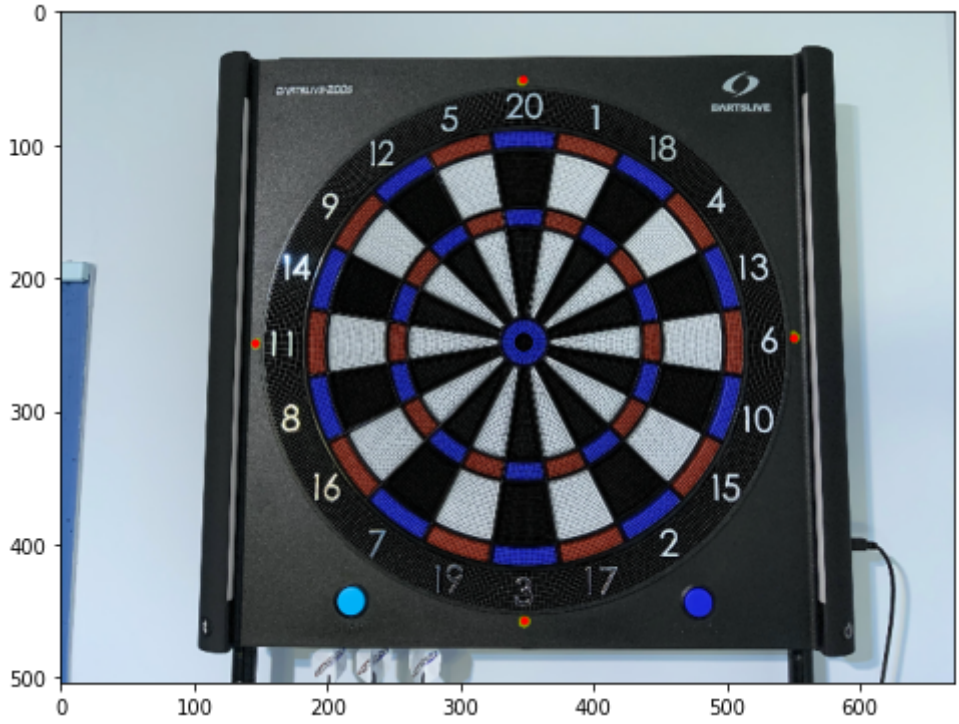



```
img_filltered_gray = cv2.cvtColor(img_filltered, cv2.COLOR_BGR2GRAY)
circles = cv2.HoughCircles(img_filltered_gray , cv2.HOUGH_GRADIENT, dp=0.4, minDist=50, param1=100,
circles = [circle for circle in circles if circle[2] < 10]

img_filltered_circle = img_filltered.copy()
for circle in circles:
  cv2.circle(img_filltered_circle, (circle[0], circle[1]), circle[2], (0, 255, 0), 1)
  cv2.circle(img_filltered_circle, (circle[0], circle[1]), 2, (0, 255, 0), 2)

plt.figure(figsize=(8,8))
plt.imshow(img_filltered_circle)
```

```
<matplotlib.image.AxesImage at 0x7f333df30790>
```

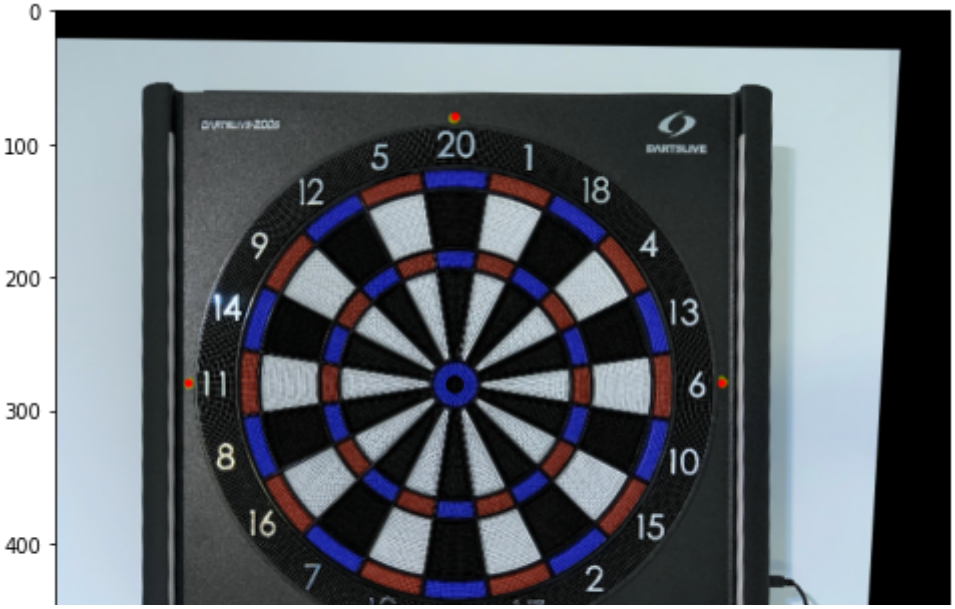
```
img_marked = img.copy()
marked_points = [circle[:2] for circle in circles]
for mp in marked_points:
  cv2.circle(img_marked, tuple(mp), 3, (255, 0, 0), -1)

# test_points = [[180, 200], [120, 150], [60, 160], [230, 110], [140, 70]]
# for tp in test_points:
#   cv2.circle(img_marked, tuple(tp), 2, (255, 255, 0), -1)

plt.figure(figsize=(8,8))
plt.imshow(img_marked)
```

```
<matplotlib.image.AxesImage at 0x7f333de9f410>
```




```
# 変換前後の対応点を設定
p_original = np.float32(marked_points)
p_trans = np.float32([[100,280], [300,480], [300,80], [500,280]])

# 変換マトリクスと射影変換
M1 = cv2.getPerspectiveTransform(p_original, p_trans)
img_marked_front = cv2.warpPerspective(img_marked, M1, (img.shape[1],img.shape[0]))

plt.figure(figsize=(8,8))
plt.imshow(img_marked_front)
```

```
<matplotlib.image.AxesImage at 0x7f333de06710>
```



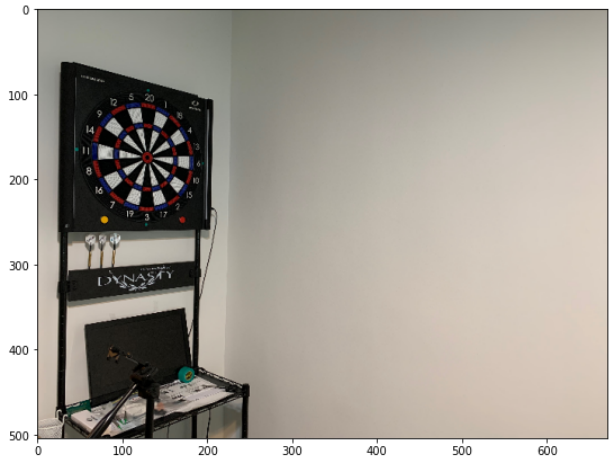

```
img_marked_front_gray = cv2.cvtColor(img_marked_front, cv2.COLOR_BGR2GRAY)
circles = cv2.HoughCircles(img_marked_front_gray , cv2.HOUGH_GRADIENT, dp=1.13, minDist=300, param1

img_marked_front2 = img_marked_front.copy()
for circle in circles:
  cv2.circle(img_marked_front2, (circle[0], circle[1]), circle[2], (0, 255, 0), 1)
  cv2.circle(img_marked_front2, (circle[0], circle[1]), 2, (0, 255, 0), 2)
plt.figure(figsize=(12,12))
plt.imshow(img_marked_front2)
```

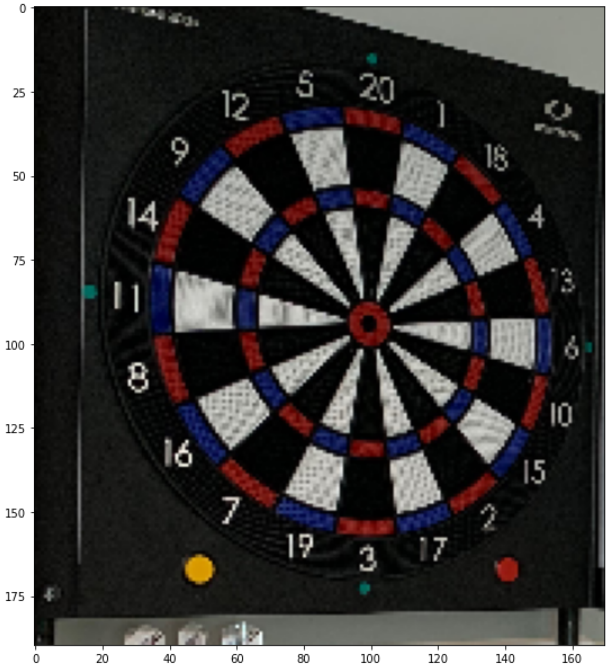
```
<matplotlib.image.AxesImage at 0x7f333ddf0f50>
```

img2

```
img = img2.copy()[80:270,30:200,:]
# img = cv2.imread('video/IMG_0158_crop.png')

plt.figure(figsize=(20,12))
plt.subplot(1,2,1)
plt.imshow(cv2.cvtColor(img2, cv2.COLOR_BGR2RGB))
plt.subplot(1,2,2)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f333dda0090>
```

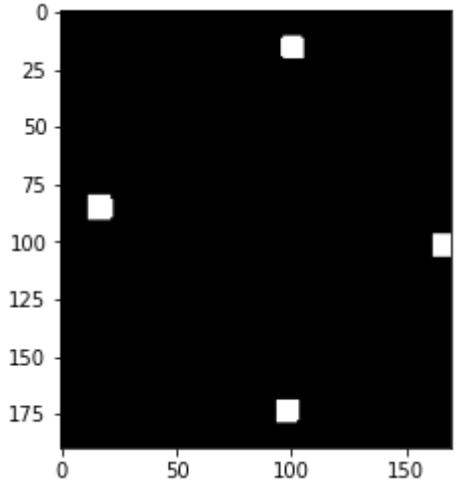




```
img_filltered = img.copy()

cond_green = (img_filltered[:,:,0]>30)&(img_filltered[:,:,1]>70)&(img_filltered[:,:,2]<60)
# cond_green = True
```

```
img_filltered[:,:,0] = np.where(cond_green, 255, 0)
img_filltered[:,:,1] = np.where(cond_green, 255, 0)
img_filltered[:,:,2] = np.where(cond_green, 255, 0)

img_filltered = cv2.blur(img_filltered, (8,8))

img_filltered[:,:,0] = np.where(img_filltered[:,:,0]>0, 255, 0)
img_filltered[:,:,1] = np.where(img_filltered[:,:,1]>0, 255, 0)
img_filltered[:,:,2] = np.where(img_filltered[:,:,2]>0, 255, 0)

plt.imshow(cv2.cvtColor(img_filltered, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f333e0a0a90>
```

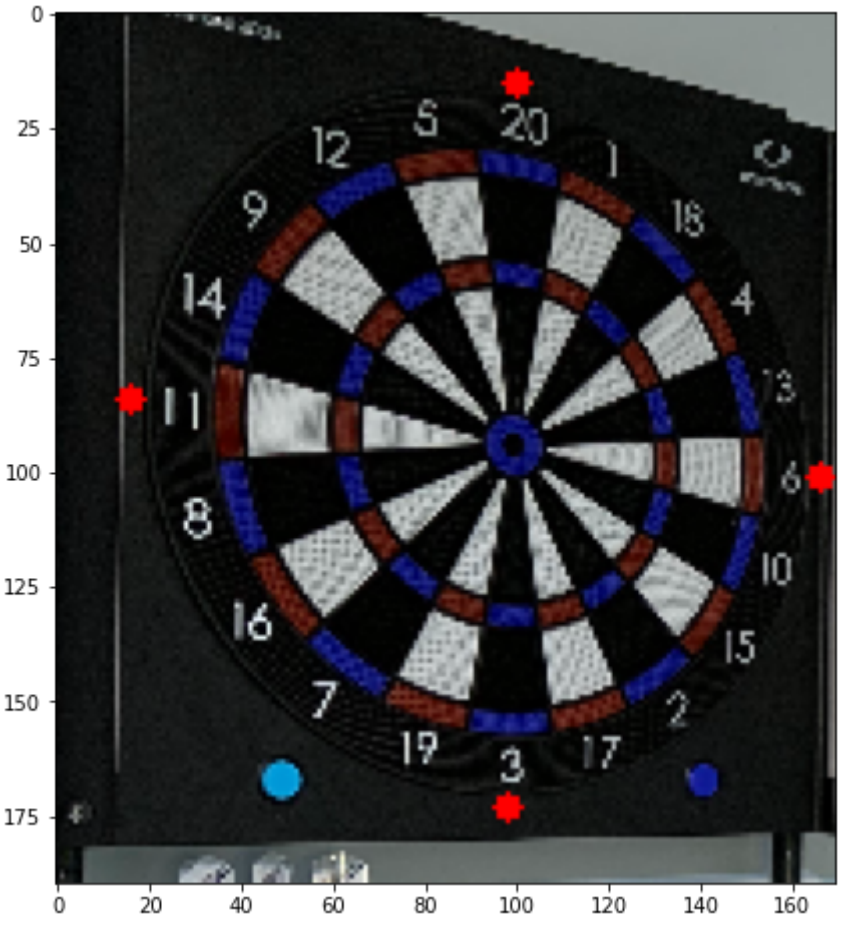



```
img_filltered_gray = cv2.cvtColor(img_filltered, cv2.COLOR_BGR2GRAY)
circles = cv2.HoughCircles(img_filltered_gray , cv2.HOUGH_GRADIENT, dp=0.3, minDist=50, param1=100,
circles = [circle for circle in circles if circle[2] < 10]

img_filltered_circle = img_filltered.copy()
for circle in circles:
  cv2.circle(img_filltered_circle, (circle[0], circle[1]), circle[2], (0, 255, 0), 1)
  cv2.circle(img_filltered_circle, (circle[0], circle[1]), 2, (0, 255, 0), 2)

plt.figure(figsize=(8,8))
plt.imshow(img_filltered_circle)
```

```
<matplotlib.image.AxesImage at 0x7f333de7d190>
```

```
img_marked = img.copy()
marked_points = [circle[:2] for circle in circles]
for mp in marked_points:
  cv2.circle(img_marked, tuple(mp), 3, (255, 0, 0), -1)

# test_points = [[180, 200], [120, 150], [60, 160], [230, 110], [140, 70]]
# for tp in test_points:
#   cv2.circle(img_marked, tuple(tp), 2, (255, 255, 0), -1)

plt.figure(figsize=(8,8))
plt.imshow(img_marked)
```

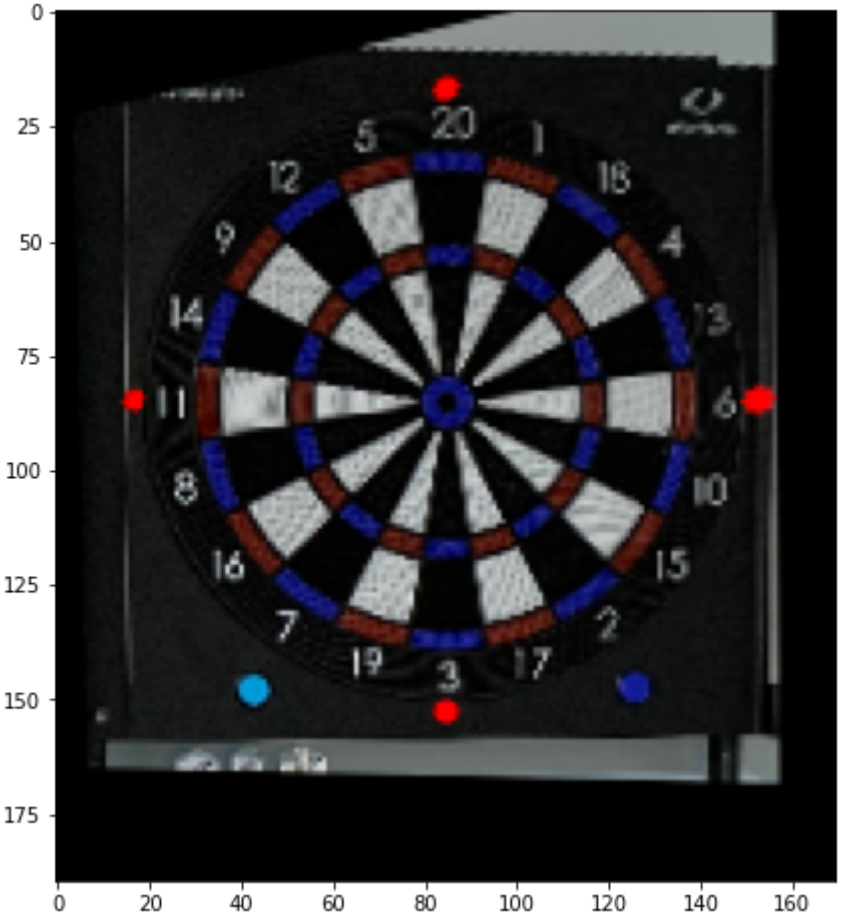
```
<matplotlib.image.AxesImage at 0x7f333dbf8090>
```




```
# 変換前後の対応点を設定
idx_mp = np.argsort([mp[0] for mp in marked_points], axis=0)
marked_points = np.array(marked_points)[idx_mp]
```

```
p_trans = np.array([[0.5,0.1], [0.5,0.9], [0.9,0.5], [0.1,0.5]])
idx = np.argsort(p_trans[:,0], axis=0)
if ((marked_points[1,1] > marked_points[2,1]) & (idx[1] < idx[2])) or ((marked_points[1,1] < marked
  tmp = idx[1]
  idx[1] = idx[2]
  idx[2] = tmp
p_trans = p_trans[idx]

p_original = np.float32(marked_points)
p_trans = p_trans.astype(np.float32)*min(img.shape[:2])

# 変換マトリクスと射影変換
M1 = cv2.getPerspectiveTransform(p_original, p_trans)
img_marked_front = cv2.warpPerspective(img_marked, M1, (img.shape[1],img.shape[0]))

plt.figure(figsize=(8,8))
plt.imshow(img_marked_front)
```

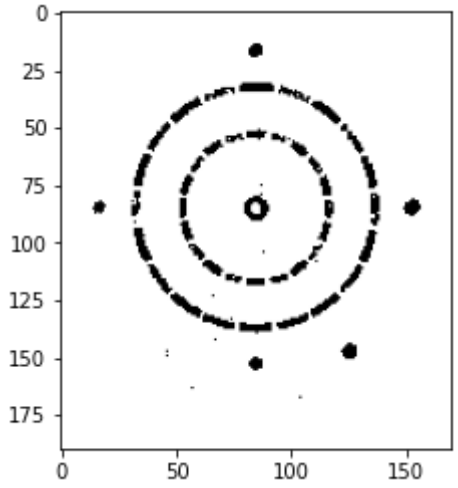
```
<matplotlib.image.AxesImage at 0x7f333db68bd0>
```




```
img_marked_front2 = img_marked_front.copy()
# img2[:,:,0] = np.where(img2[:,:,0]<100, 255, 0)
# img2[:,:,2] = np.where(img2[:,:,2]<130, 255, 0)
# img2[:,:,1] = np.ones(img2.shape[:2])*0


cond_red = (img_marked_front2[:,:,0]<70)&(img_marked_front2[:,:,1]<70)&(img_marked_front2[:,:,2]>70
cond_blue = (img_marked_front2[:,:,0]>70)&(img_marked_front2[:,:,1]<70)&(img_marked_front2[:,:,2]<7
```

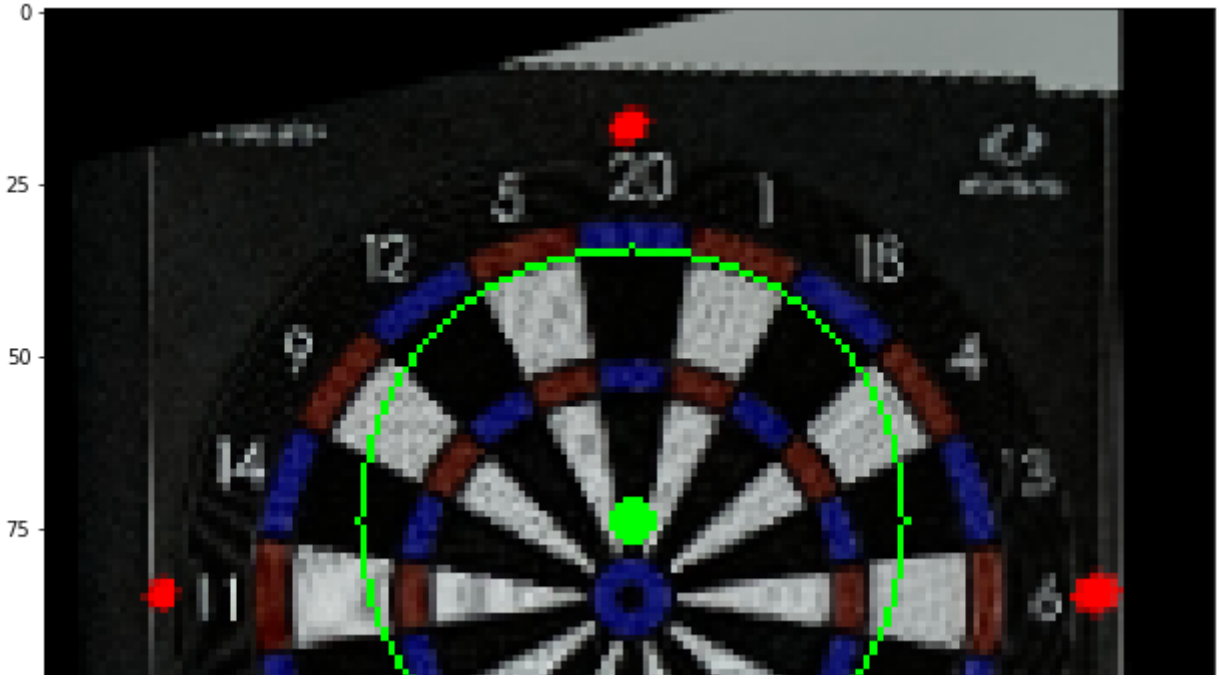
```
# cond_blue = False
cond_red_blue = cond_red | cond_blue
# cond_black = (img_filltered[:,:,0]>120)&(img_filltered[:,:,1]>120)&(img_filltered[:,:,2]>120)

img_marked_front2[:,:,0] = np.where(cond_red_blue, 0, 255)
img_marked_front2[:,:,1] = np.where(cond_red_blue, 0, 255)
img_marked_front2[:,:,2] = np.where(cond_red_blue, 0, 255)

plt.imshow(cv2.cvtColor(img_marked_front2, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f333daed7d0>
```




```
img_marked_front_gray = cv2.cvtColor(img_marked_front2, cv2.COLOR_BGR2GRAY)
circles = cv2.HoughCircles(img_marked_front_gray , cv2.HOUGH_GRADIENT, dp=0.4, minDist=400, param1=

img_marked_front2 = img_marked_front.copy()
for circle in circles:
  cv2.circle(img_marked_front2, (circle[0], circle[1]), circle[2], (0, 255, 0), 1)
  cv2.circle(img_marked_front2, (circle[0], circle[1]), 2, (0, 255, 0), 2)
plt.figure(figsize=(12,12))
plt.imshow(img_marked_front2)
```

```
<matplotlib.image.AxesImage at 0x7f333da73590>
```

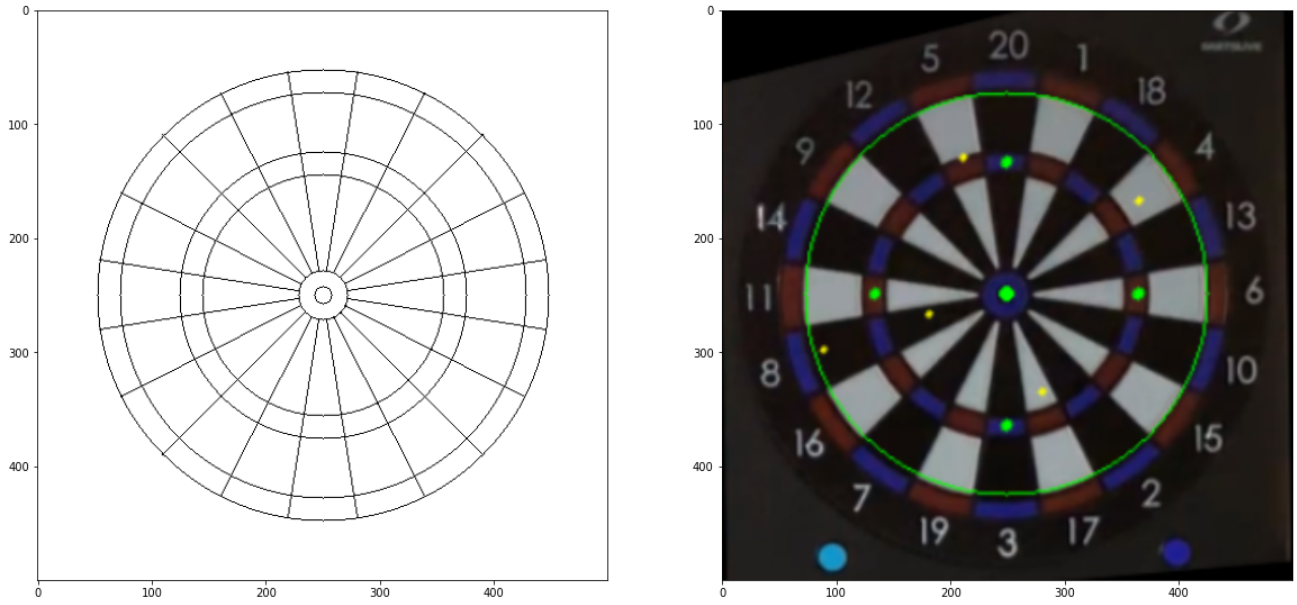


19.5：中心から店の外側 24：マークまで

既知の盤面にマッピング (Done)

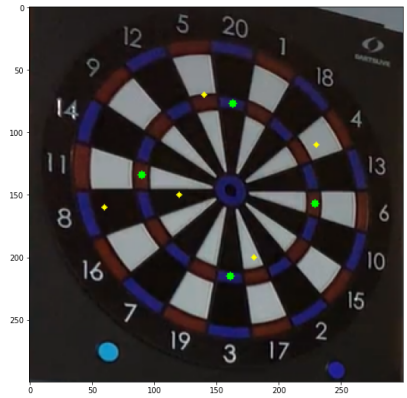
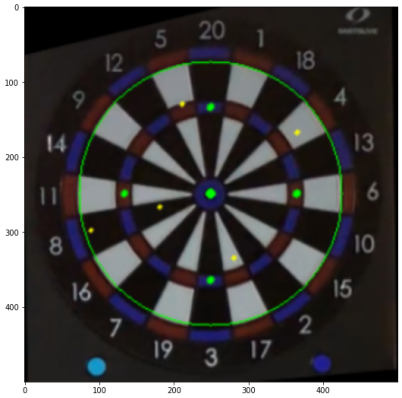
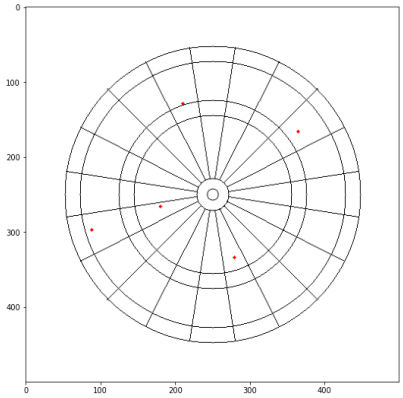
```
detected_board_circle = circles[0]
detected_center = detected_board_circle[:2]
detected_r = detected_board_circle[2]
detected_margine = detected_r/(img.shape[0]/2)

img_warped = img_marked_front2.copy()

dx = img_center[0] - detected_center[0]
dy = img_center[1] - detected_center[1]
M2 = np.float32([[1,0,dx],[0,1,dy]])
img_warped = cv2.warpAffine(img_warped, M2, img.shape[:2])

zoom_rate = coordinate_margine/detected_margine
M3 = np.float32([[zoom_rate,0,-(zoom_rate-1)*img_center[0]],[0,zoom_rate,-(zoom_rate-1)*img_center[
img_warped = cv2.warpAffine(img_warped, M3, img.shape[:2])

img_warped = cv2.resize(img_warped, img_coordinate.shape[:2], interpolation = cv2.INTER_LINEAR)

plt.figure(figsize=(20,12))
plt.subplot(1,2,1)
plt.imshow(img_coordinate)
plt.subplot(1,2,2)
plt.imshow(img_warped)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255
<matplotlib.image.AxesImage at 0x7f751b16d1d0>
```




```
img_result = img_coordinate.copy()

for tp in test_points:
  a = np.array([[tp]], dtype=np.float32)
  b = cv2.perspectiveTransform(a, M1)[0][0]
  c = M2@np.array([b[0],b[1],1], np.float32).T
  d = M3@np.array([c[0],c[1],1], np.float32).T
  res_x, res_y = d[0]/(img.shape[1]/coordinate_size), d[1]/(img.shape[0]/coordinate_size)
  print(res_x, res_y)
  cv2.circle(img_result, (int(res_x), int(res_y)), 2, (255, 0, 0), -1)

plt.figure(figsize=(30,12))
plt.subplot(1,3,1)
plt.imshow(img_marked)
plt.subplot(1,3,2)
plt.imshow(img_warped)
plt.subplot(1,3,3)
plt.imshow(img_result)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255
279.73210652669275 334.23423767089844
180.40420532226562 266.3451639811198
88.33202362060547 297.3236338297526
364.2227681477865 166.89114888509116
210.51302591959637 129.25448099772137
<matplotlib.image.AxesImage at 0x7f751b07d050>
```

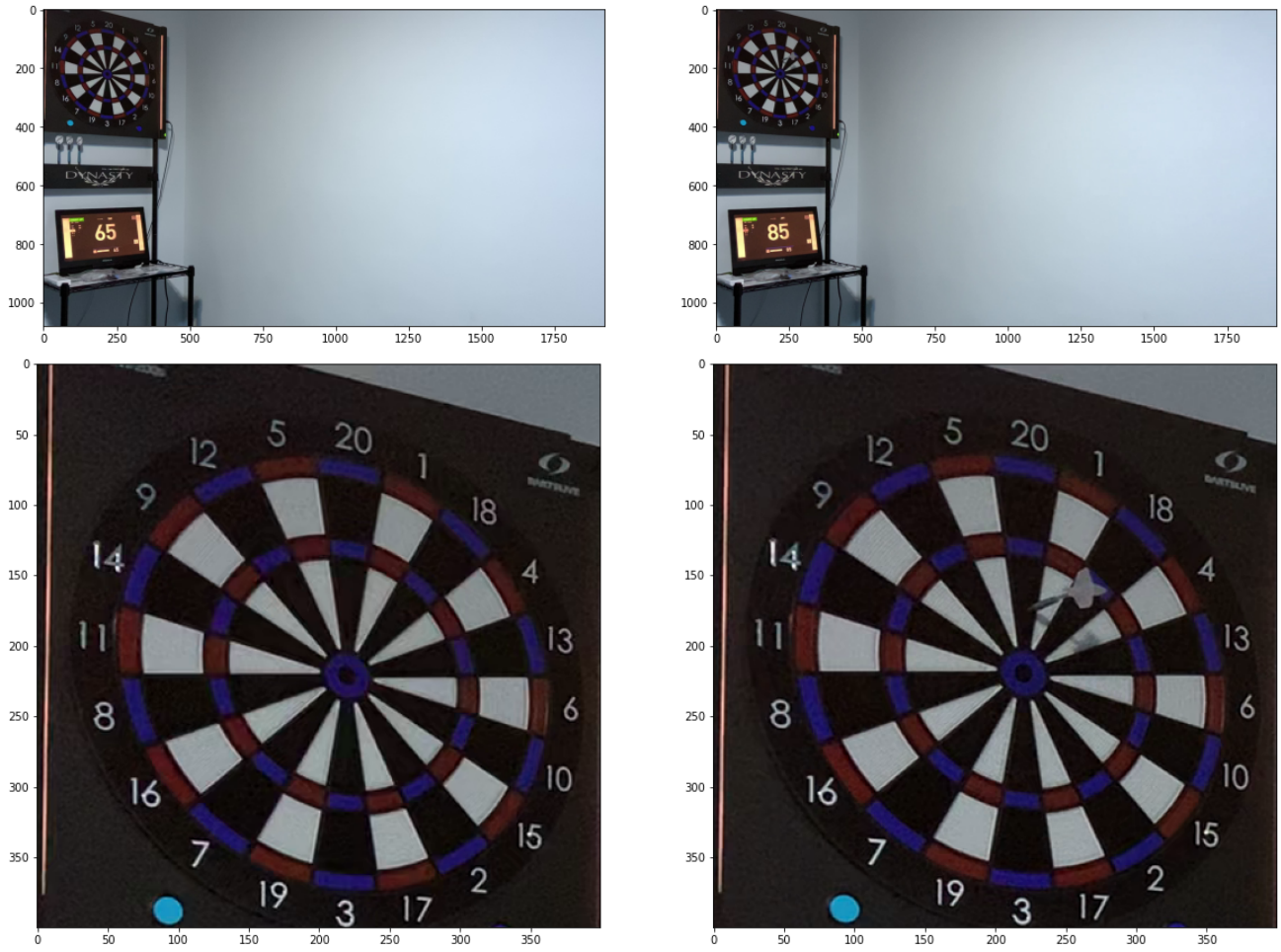
投擲位置検出

色なし案 (無理)

```
cap = cv2.VideoCapture('video/IMG_0144.MOV')
crop_width = 400
crop_height = 400
idx_frame0 = 0
idx_frame1 = 440

cap.set(cv2.CAP_PROP_POS_FRAMES, idx_frame0)
ret, frame0 = cap.read()
frame0_crop = frame0[:crop_height,:crop_width,:]
cap.set(cv2.CAP_PROP_POS_FRAMES, idx_frame1)
ret, frame1 = cap.read()
frame1_crop = frame1[:crop_height,:crop_width,:]


plt.figure(figsize=(20,12))
plt.subplot(1,2,1)
plt.imshow(frame0)
plt.subplot(1,2,2)
plt.imshow(frame1)
```

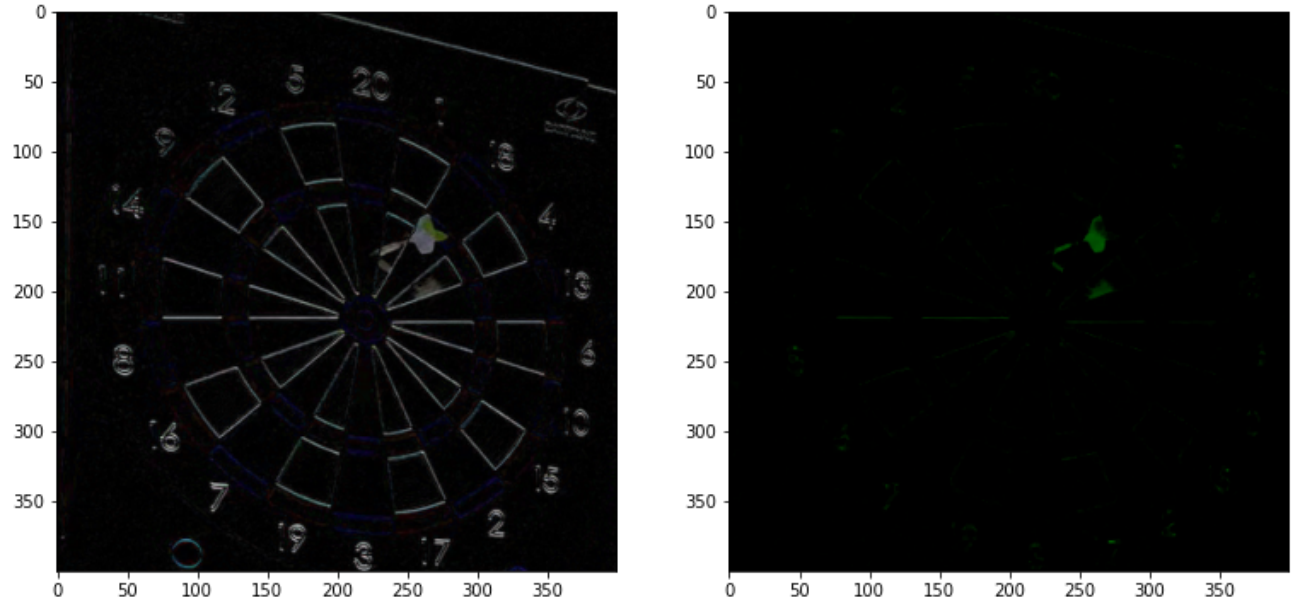
```
plt.figure(figsize=(20,12))
plt.subplot(1,2,1)
plt.imshow(frame0_crop)
plt.subplot(1,2,2)
plt.imshow(frame1_crop)
```

```
<matplotlib.image.AxesImage at 0x7f75063dc990>
```



```
def drop_color(img):
  cond_drop_red = img[:,:,2]<100
  cond_drop_blue = img[:,:,0]<50
  cond_drop_blue = False
  cond_drop_red_blue = cond_drop_red | cond_drop_blue
  img2 = frame0_crop.copy()
  img2[:,:,0] = np.where(cond_drop_red_blue,0,1)
  img2[:,:,1] = np.where(cond_drop_red_blue,0,1)
  img2[:,:,2] = np.where(cond_drop_red_blue,0,1)
  img2 = img2 * img
  return img


a = drop_color(frame0_crop)
b = drop_color(frame1_crop)
c = cv2.absdiff(a, b)
d = minimum_filter(c,size=3,mode='constant')

plt.figure(figsize=(20,8))
plt.subplot(1,3,1)
plt.imshow(c)
plt.subplot(1,3,2)
plt.imshow(d)
```

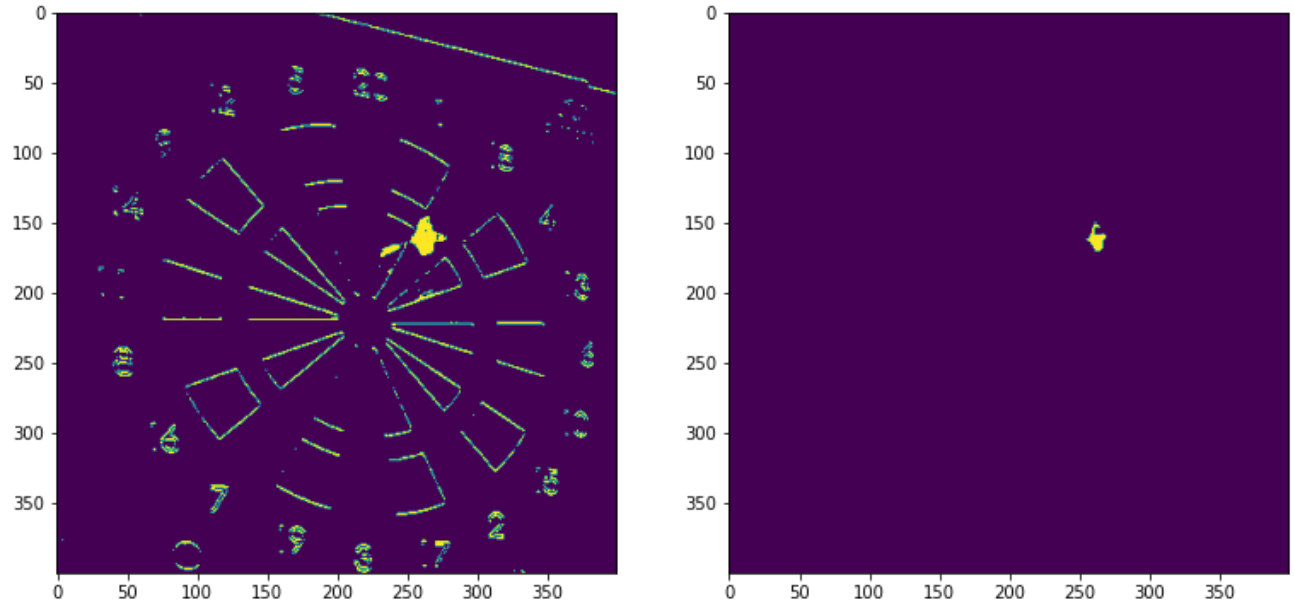
```
<matplotlib.image.AxesImage at 0x7f750b496350>
```



```
c2 = c.copy()
color_drop = (c2[:,:,0]<70) & (c2[:,:,1]<70) & (c2[:,:,2]<70)
c2[:,:,0] = np.where(color_drop,0,255)
c2[:,:,1] = np.where(color_drop,0,255)
c2[:,:,2] = np.where(color_drop,0,255)

c2 = cv2.cvtColor(c2, cv2.COLOR_BGR2GRAY)
d2 = minimum_filter(c2,size=5,mode='constant')
```

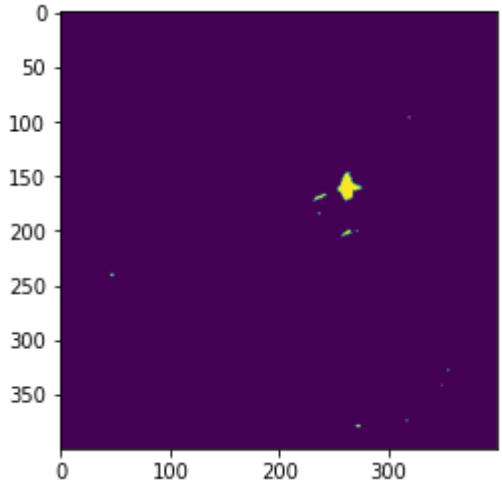
```
plt.figure(figsize=(20,8))
plt.subplot(1,3,1)
plt.imshow(c2)
plt.subplot(1,3,2)
plt.imshow(d2)
```

```
<matplotlib.image.AxesImage at 0x7f75085a24d0>
```



```
lines = cv2.HoughLinesP(minimum_filter(d2,size=3,mode='constant'), rho=1, theta=np.pi/360, threshol

img_line = d2.copy()
# for line in lines:
#   line = line[0]
#   cv2.line(img_line, tuple(line[:2]), tuple(line[2:]), (255, 255, 255), thickness=1, lineType=cv2

# plt.figure(figsize=(8,8))
plt.imshow(img_line)
```

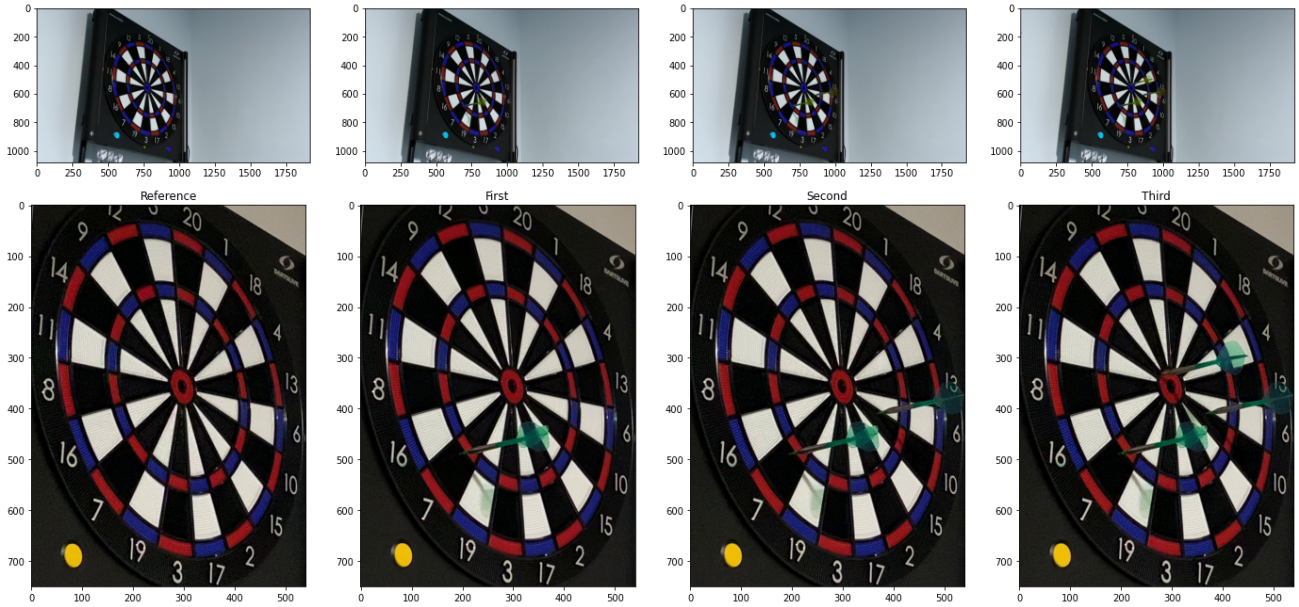
```
<matplotlib.image.AxesImage at 0x7f7508c234d0>
```



色あり案

```
cap = cv2.VideoCapture('video/IMG_0164.MOV')
img_arrow = cv2.imread('video/arrow.png')

idx_frame0 = 0
idx_frame1 = 550
idx_frame2 = 800
idx_frame3 = 1000

cap.set(cv2.CAP_PROP_POS_FRAMES, idx_frame0)
ret, frame0 = cap.read()
frame0_crop = frame0[200:950,480:1020,:]
cap.set(cv2.CAP_PROP_POS_FRAMES, idx_frame1)
ret, frame1 = cap.read()
frame1_crop = frame1[200:950,480:1020,:]
cap.set(cv2.CAP_PROP_POS_FRAMES, idx_frame2)
ret, frame2 = cap.read()
frame2_crop = frame2[200:950,480:1020,:]
cap.set(cv2.CAP_PROP_POS_FRAMES, idx_frame3)
ret, frame3 = cap.read()
frame3_crop = frame3[200:950,480:1020,:]


plt.figure(figsize=(24,12))
plt.subplot(1,4,1)
plt.imshow(frame0)
plt.subplot(1,4,2)
plt.imshow(frame1)
plt.subplot(1,4,3)
plt.imshow(frame2)
plt.subplot(1,4,4)
plt.imshow(frame3)

plt.figure(figsize=(24,8))
plt.subplot(1,4,1)
plt.title("Reference")
plt.imshow(cv2.cvtColor(frame0_crop, cv2.COLOR_BGR2RGB))
plt.subplot(1,4,2)
plt.title("First")
plt.imshow(cv2.cvtColor(frame1_crop, cv2.COLOR_BGR2RGB))
plt.subplot(1,4,3)
plt.title("Second")
plt.imshow(cv2.cvtColor(frame2_crop, cv2.COLOR_BGR2RGB))
plt.subplot(1,4,4)
plt.title("Third")
plt.imshow(cv2.cvtColor(frame3_crop, cv2.COLOR_BGR2RGB))
```

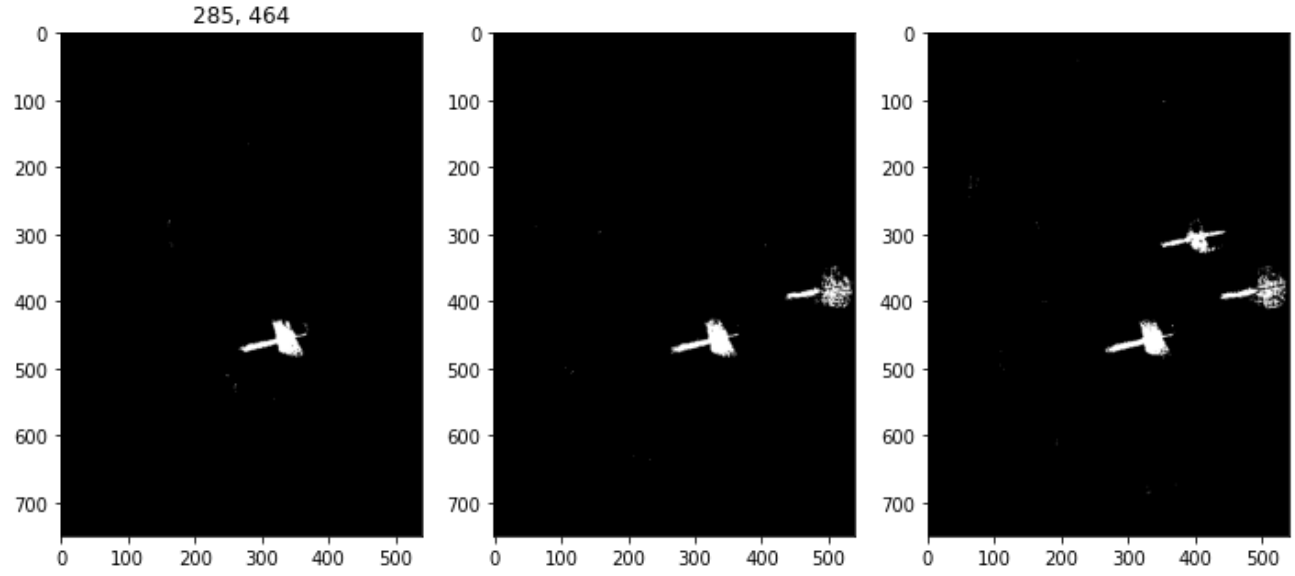
```
<matplotlib.image.AxesImage at 0x7fb1e473e050>
```




```
def is_ignore_pixel(img, idx, column, kernel_size, p):
  img_crop = img[max(0,idx-int(kernel_size/2)):idx+int(kernel_size/2), max(0,column-int(kernel_size
  if np.percentile(img_crop, p) == 255:
    # print(idx,column)
    # print(img_crop)
    return False
  else:
    return True


def extract_arrow(img, diff_image = False, ignore_kernel_size = 3, p = 50):
  img_arrow = img.copy()

  if not diff_image:
    cond_green = (img[:,:,0]>40) & (img[:,:,1]>40) & (img[:,:,2]<30)
    img_arrow[:,:,0] = np.where(cond_green, 255, 0)
    img_arrow[:,:,1] = np.where(cond_green, 255, 0)
    img_arrow[:,:,2] = np.where(cond_green, 255, 0)

  for c in range(0,img.shape[1]):
    i = np.argmax(img_arrow[:,c,1])
    if is_ignore_pixel(img_arrow,i,c,ignore_kernel_size,p):
      continue
    est_xtip = c
    est_ytip = i
```

```
        break

    return img_arrow, est_xtip, est_ytip


img_arrow1, est_xtip1, est_ytip1 = extract_arrow(frame1_crop, ignore_kernel_size = 3)
img_arrow2, _, _ = extract_arrow(frame2_crop)
img_arrow3, _, _ = extract_arrow(frame3_crop)

plt.figure(figsize=(12,6))
plt.subplot(1,3,1)
plt.title(f'{est_xtip1}, {est_ytip1}')
plt.imshow(img_arrow1)
plt.subplot(1,3,2)
plt.imshow(img_arrow2)
plt.subplot(1,3,3)
plt.imshow(img_arrow3)
```

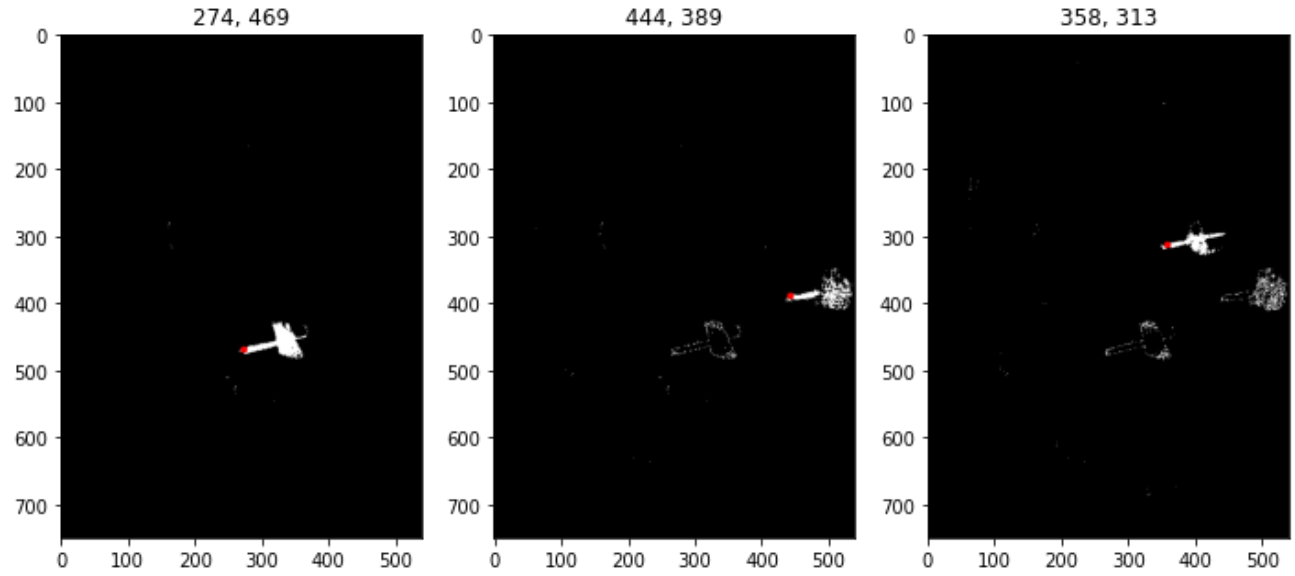
```
<matplotlib.image.AxesImage at 0x7fb1e459c790>
```




```
img_arrow1, est_xtip1, est_ytip1 = extract_arrow(frame1_crop, ignore_kernel_size = 8, p = 50)

img_arrow2_diff = cv2.absdiff(img_arrow2, img_arrow1)
img_arrow2_diff, est_xtip2, est_ytip2 = extract_arrow(img_arrow2_diff, diff_image = True, ignore_ke

img_arrow3_diff = cv2.absdiff(img_arrow3, img_arrow2)
img_arrow3_diff, est_xtip3, est_ytip3 = extract_arrow(img_arrow3_diff, diff_image = True, ignore_ke


plt.figure(figsize=(12,6))
plt.subplot(1,3,1)
plt.title(f'{est_xtip1}, {est_ytip1}')
cv2.circle(img_arrow1, (est_xtip1, est_ytip1), 5, (255, 0, 0), -1)
plt.imshow(img_arrow1)
plt.subplot(1,3,2)
plt.title(f'{est_xtip2}, {est_ytip2}')
cv2.circle(img_arrow2_diff, (est_xtip2, est_ytip2), 5, (255, 0, 0), -1)
```

```
plt.imshow(img_arrow2_diff)
plt.subplot(1,3,3)
plt.title(f'{est_xtip3}, {est_ytip3}')
cv2.circle(img_arrow3_diff, (est_xtip3, est_ytip3), 5, (255, 0, 0), -1)
plt.imshow(img_arrow3_diff)
```

```
<matplotlib.image.AxesImage at 0x7fb1e4a68e10>
```




```
given_xtip, given_ytip = (199, 488)

tuned_dx = given_xtip - est_xtip1
tuned_dy = given_ytip - est_ytip1


plt.figure(figsize=(16,8))
plt.subplot(1,3,1)
plt.title("First")
img_marked = frame1_crop.copy()
cv2.circle(img_marked, (given_xtip, given_ytip), 4, (0, 255, 0), -1)
cv2.circle(img_marked, (est_xtip1, est_ytip1), 4, (255, 0, 0), -1)
plt.imshow(img_marked)

plt.subplot(1,3,2)
plt.title("Second")
img_marked = frame2_crop.copy()
cv2.circle(img_marked, (est_xtip2 + tuned_dx, est_ytip2 + tuned_dy), 4, (255, 255, 0), -1)
cv2.circle(img_marked, (est_xtip2, est_ytip2), 4, (255, 0, 0), -1)
plt.imshow(img_marked)

plt.subplot(1,3,3)
plt.title("Third")
img_marked = frame3_crop.copy()
cv2.circle(img_marked, (est_xtip3 + tuned_dx, est_ytip3 + tuned_dy), 4, (255, 255, 0), -1)
cv2.circle(img_marked, (est_xtip3, est_ytip3), 4, (255, 0, 0), -1)
plt.imshow(img_marked)
```

```
<matplotlib.image.AxesImage at 0x7fe736063190>
```

投擲軌道検出

セットアップ

```
def is_ignore_pixel(img, idx, column, kernel_size, p):
  img_crop = img[max(0,idx-int(kernel_size/2)):idx+int(kernel_size/2), max(0,column-int(kernel_size
  # print(img_crop.shape)
  if np.percentile(img_crop, p) == 255:
    # print(idx,column)
    # print(img_crop)
    return False, img_crop
  else:
    return True, img_crop


def extract_arrow(img, condition, diff_image = False, ignore_kernel_size = 3, p = 50):
  img_arrow = img.copy()

  if not diff_image:
    img_arrow[:,:,0] = np.where(condition, 255, 0)
    img_arrow[:,:,1] = np.where(condition, 255, 0)
    img_arrow[:,:,2] = np.where(condition, 255, 0)


  est_xtip, est_ytip = np.nan, np.nan
  for c in range(0,img.shape[1]):
    i = np.argmax(img_arrow[:,c,1])
```

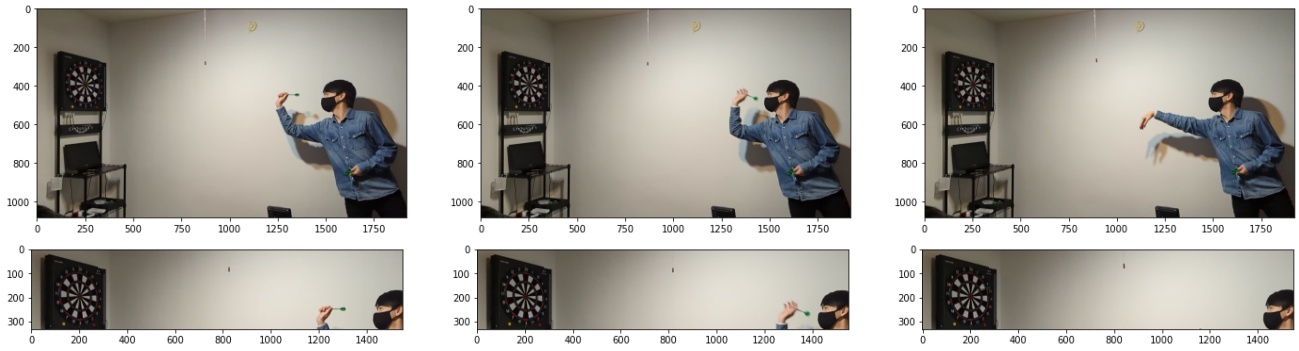
```
        state, img_crop = is_ignore_pixel(img_arrow,i,c,ignore_kernel_size,p)
        if state:
          continue
        est_xtip = c
        est_ytip = i
        # print(img_crop)
        break

      return img_arrow, est_xtip, est_ytip

    from sklearn import preprocessing

    def estimate_axis(img, est_xtip, est_ytip, kernel, rate=20):
      # print(img)
      # print(max(0,est_ytip-kernel),est_ytip+kernel, max(0,est_xtip-kernel),est_xtip+kernel)
      img = img[max(0,est_ytip-kernel):est_ytip+kernel, max(0,est_xtip-kernel):est_xtip+kernel, 1].copy
      # print(img)
      x, y = np.where(img > 10)
      samples = np.array([x,y]).T
      mm = preprocessing.MinMaxScaler()
      samples = mm.fit_transform(samples)
      # samples = scipy.stats.zscore(samples)

      pcamodel = PCA()
      pcamodel.fit(samples)
      pca_cor = pcamodel.transform(samples)
      # print(pca_cor)
      dx, dy = pcamodel.components_[0]
      if dx > 0:
        axis = np.array([[est_xtip-rate*dx, est_ytip-rate*dy], [est_xtip, est_ytip]], dtype=int)
      else:
        axis = np.array([[est_xtip, est_ytip], [est_xtip+rate*dx, est_ytip+rate*dy]], dtype=int)

      return axis


    def lineList(x1, y1, x2, y2):
        line_lst = []
        step = 0
        dx = abs(x2 - x1)
        dy = abs(y2 - y1)
        if dx > dy:
            if x1 > x2 :
                step = 0
                if y1 > y2:
                    step = 1
                else:
                    step = -1
                x1, x2 = x2, x1 # swap
                y1 = y2;
            else:
                if y1 < y2:
                    step = 1
                else:
                    step = -1
```

```
            line_lst.append((x1, y1))
            s = dx >> 1
            x1 += 1
            while (x1 <= x2):
                s -= dy
                if s < 0:
                    s += dx
                    y1 += step
                line_lst.append((x1, y1))
                x1 += 1
        else:
            if y1 > y2:
                if x1 > x2:
                    step = 1
                else:
                    step = -1

                y1, y2 = y2, y1 # swap
                x1 = x2
            else:
                if x1 < x2:
                    step = 1
                else:
                    step = -1
            line_lst.append((x1, y1))
            s = dy >> 1
            y1 += 1
            while y1 <= y2:
                s -= dx
                if s < 0:
                    s += dy
                    x1 += step
                line_lst.append((x1, y1))
                y1 += 1
        return  line_lst

    #   img         イメージ
    #   start_point 始点
    #   end_point   終点
    #   gap         点線ギャップ（間隔）
    #   linewidth   線幅
    #   color       色
    #
    def drawDashedLine(img, start_point, end_point, gap, linewidth, color):
        li = lineList(start_point[0], start_point[1], end_point[0], end_point[1])
        fwd = start_point
        bwd = start_point
        j = 0
        for i, pt in enumerate(li):
            if i % gap == 0:
                bwd = pt

                if(j % 2):
                    cv2.line(img, fwd, bwd, color, linewidth, lineType=cv2.LINE_AA)
                fwd = bwd
```

```
            j += 1
    return img



def detect_release(img_arrow, img_hand, axis, kernel=5):
  p_grasp = axis[np.argmin(axis[:,0])]
  pg_minw, pg_maxw = p_grasp[0]-kernel, p_grasp[0]+kernel
  pg_minh, pg_maxh = p_grasp[1]-kernel, p_grasp[1]+kernel
  grasp_area = np.sum(img_hand[pg_minh:pg_maxh,pg_minw:pg_maxw,:])
  if grasp_area == 0:
    return True
  else:
    return False


# movie_path = 'video/u4-t.mp4'
# movie_path = 'IMG_5308.mov'
movie_path = 'video/u5-1.mp4'

cap = cv2.VideoCapture(movie_path)

# start_idx = 1910
# end_idx = 1990
start_idx = 315
end_idx = 450

hmin = 200
hmax = 530
wmin = 50
wmax = 1600

cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
ret, frame0 = cap.read()
frame0_crop = frame0[hmin:hmax,wmin:wmax,:]
cap.set(cv2.CAP_PROP_POS_FRAMES, start_idx)
ret, frame1 = cap.read()
frame1_crop = frame1[hmin:hmax,wmin:wmax,:]
cap.set(cv2.CAP_PROP_POS_FRAMES, end_idx)
ret, frame2 = cap.read()
frame2_crop = frame2[hmin:hmax,wmin:wmax,:]


plt.figure(figsize=(24,12))
plt.subplot(1,3,1)
plt.imshow(cv2.cvtColor(frame0, cv2.COLOR_BGR2RGB))
plt.subplot(1,3,2)
plt.imshow(cv2.cvtColor(frame1, cv2.COLOR_BGR2RGB))
plt.subplot(1,3,3)
plt.imshow(cv2.cvtColor(frame2, cv2.COLOR_BGR2RGB))

plt.figure(figsize=(24,24))
plt.subplot(1,3,1)
plt.imshow(cv2.cvtColor(frame0_crop, cv2.COLOR_BGR2RGB))
plt.subplot(1,3,2)
```

```
plt.imshow(cv2.cvtColor(frame1_crop, cv2.COLOR_BGR2RGB))
plt.subplot(1,3,3)
plt.imshow(cv2.cvtColor(frame2_crop, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7fbda93409d0>
```



```
cap = cv2.VideoCapture(movie_path)

frames = []
for idx in range(start_idx,end_idx,1):
  cap.set(cv2.CAP_PROP_POS_FRAMES, idx)
  _, frame = cap.read()
  frames.append(frame)
```

射出検出

```
for i in range(0,20):
  tmp_img = frames[i][hmin:hmax,wmin:wmax,:].copy()

  # 手の検出
  cond_blue = (130<tmp_img[:,:,0]) & (tmp_img[:,:,0]<180)
  cond_green = (150<tmp_img[:,:,1]) & (tmp_img[:,:,1]<190)
  cond_red = (200<tmp_img[:,:,2]) & (tmp_img[:,:,2]<230)
  condition = cond_blue & cond_green & cond_red
  img_hand, _, _ = extract_arrow(tmp_img, condition, ignore_kernel_size = 8, p = 60)

  # 矢の検出
  cond_blue = (30<tmp_img[:,:,0]) & (tmp_img[:,:,0]<90)
  cond_green = 70<tmp_img[:,:,1]
  cond_red = (30<tmp_img[:,:,2]) & (tmp_img[:,:,2]<90)
  condition = cond_blue & cond_green & cond_red
```

```
  cond_blue = (100<tmp_img[:,:,0]) & (tmp_img[:,:,0]<150)
  cond_green = 150<tmp_img[:,:,1]
  cond_red = (100<tmp_img[:,:,2]) & (tmp_img[:,:,2]<150)
  condition = (cond_blue & cond_green & cond_red) | condition

  img_arrow, est_xtip, est_ytip = extract_arrow(tmp_img, condition, ignore_kernel_size = 16, p = 70

  if est_xtip == est_xtip:
    # 先端の検出
    axis = estimate_axis(img_arrow, est_xtip, est_ytip, kernel=30, rate=55)
    img_axis = np.zeros(tmp_img.shape)
    cv2.line(img_axis,tuple(axis[0]),tuple(axis[1]),(255,0,0),6)

    # 把持位置の検出
    print(detect_release(img_arrow, img_hand, axis, kernel=5))

    plt.figure(figsize=(24,24))
    plt.subplot(1,2,1)
    plt.imshow(cv2.cvtColor(img_arrow, cv2.COLOR_BGR2RGB))
    plt.subplot(1,2,2)
    cv2.line(tmp_img,tuple(axis[0]),tuple(axis[1]),(255,0,0),6)
    cv2.circle(tmp_img, (int(est_xtip), int(est_ytip)), 3, (0, 0, 255), -1)
    plt.imshow(cv2.cvtColor(tmp_img, cv2.COLOR_BGR2RGB))
```

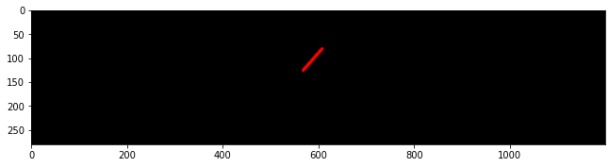
```
    False
    False
    False
    False
    False
    False
    False
    False
    False
    True
    True
    True
    True
    True
    True
    True
    True
    True
    True
```

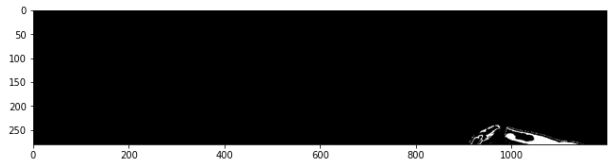
```
axis
```

```
    array([[1090,  163],
           [1129,  118]])
```

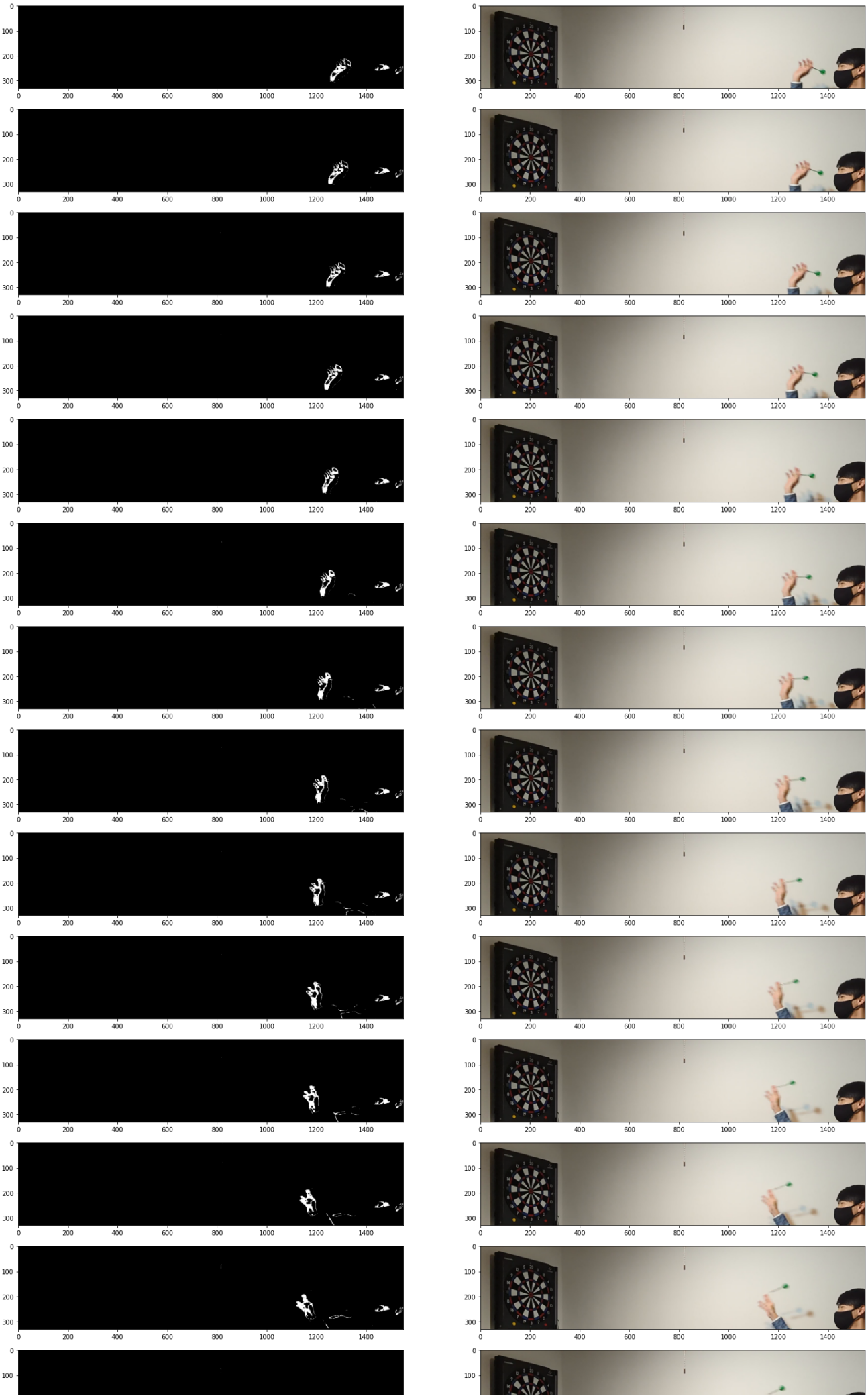
```
plt.figure(figsize=(24,24))
plt.subplot(1,2,1)
plt.imshow(img_axis)
plt.subplot(1,2,2)
plt.imshow(img_hand)
```

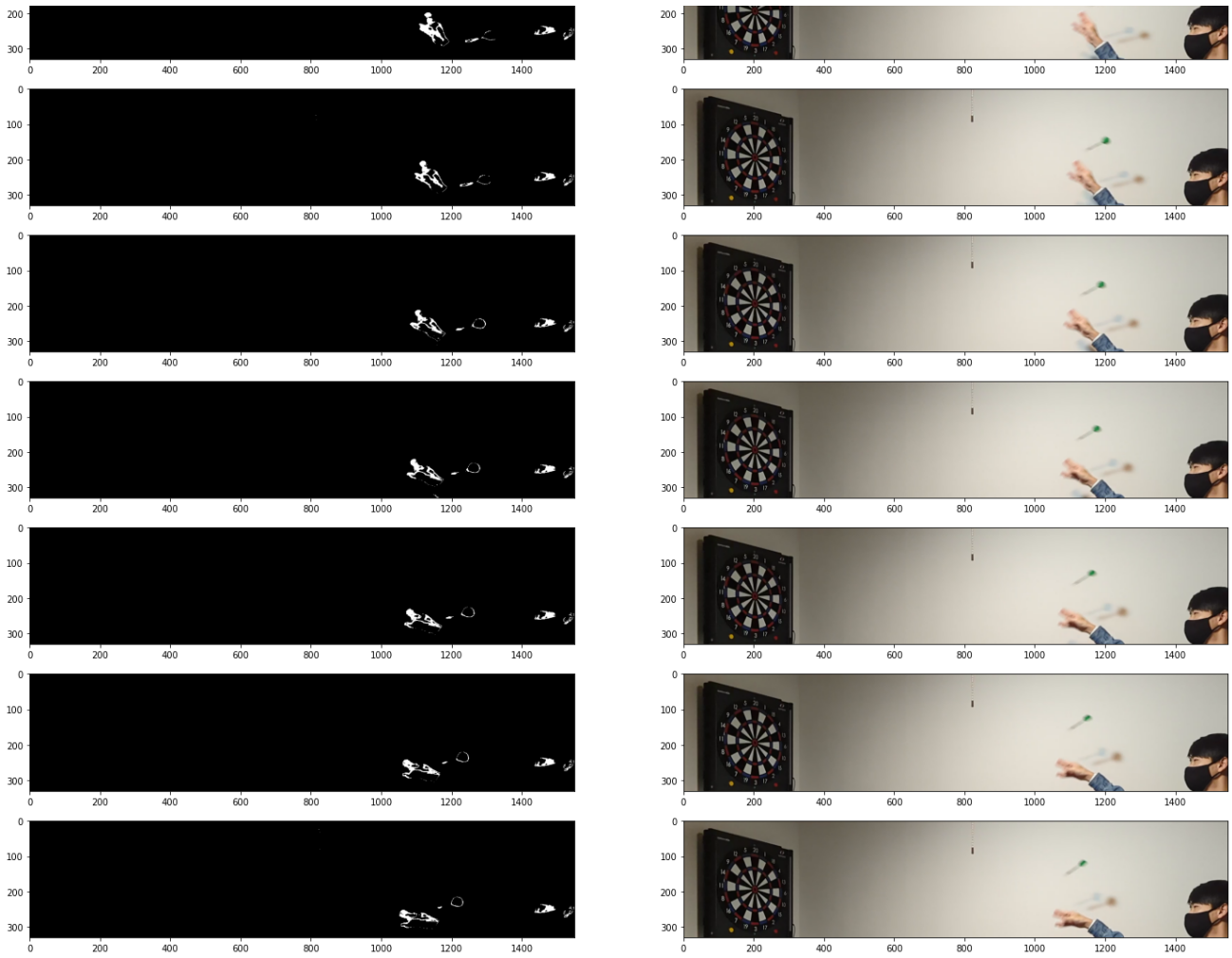
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255
<matplotlib.image.AxesImage at 0x7f1480155550>
```






```
gif_imgs = []
cv2_imgs = []
diff_imgs = []
for idx in range(0, 20, 1):
  # print(idx)
  tmp_img = frames[idx][hmin:hmax,wmin:wmax,:].copy()

  cond_blue = (130<tmp_img[:,:,0]) & (tmp_img[:,:,0]<180)
  cond_green = (150<tmp_img[:,:,1]) & (tmp_img[:,:,1]<200)
  cond_red = (200<tmp_img[:,:,2]) & (tmp_img[:,:,2]<230)
  condition = cond_blue & cond_green & cond_red

  img_arrow, est_xtip, est_ytip = extract_arrow(tmp_img, condition, ignore_kernel_size = 8, p = 60)
  gif_imgs.append(img_arrow)

  plt.figure(figsize=(24,24))
  plt.subplot(1,2,1)
  plt.imshow(cv2.cvtColor(img_arrow, cv2.COLOR_BGR2RGB))
  plt.subplot(1,2,2)
  plt.imshow(cv2.cvtColor(tmp_img, cv2.COLOR_BGR2RGB))
```



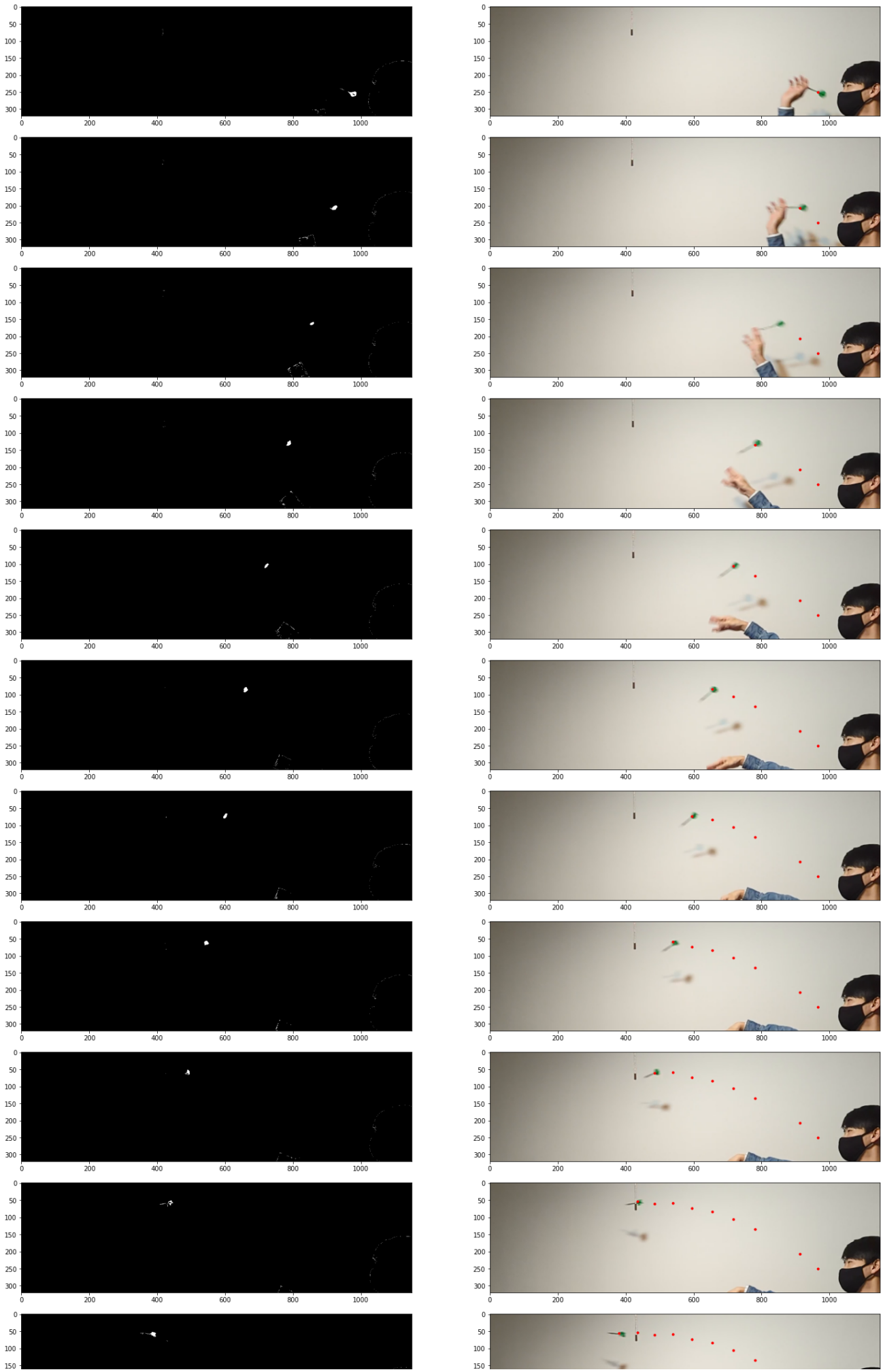


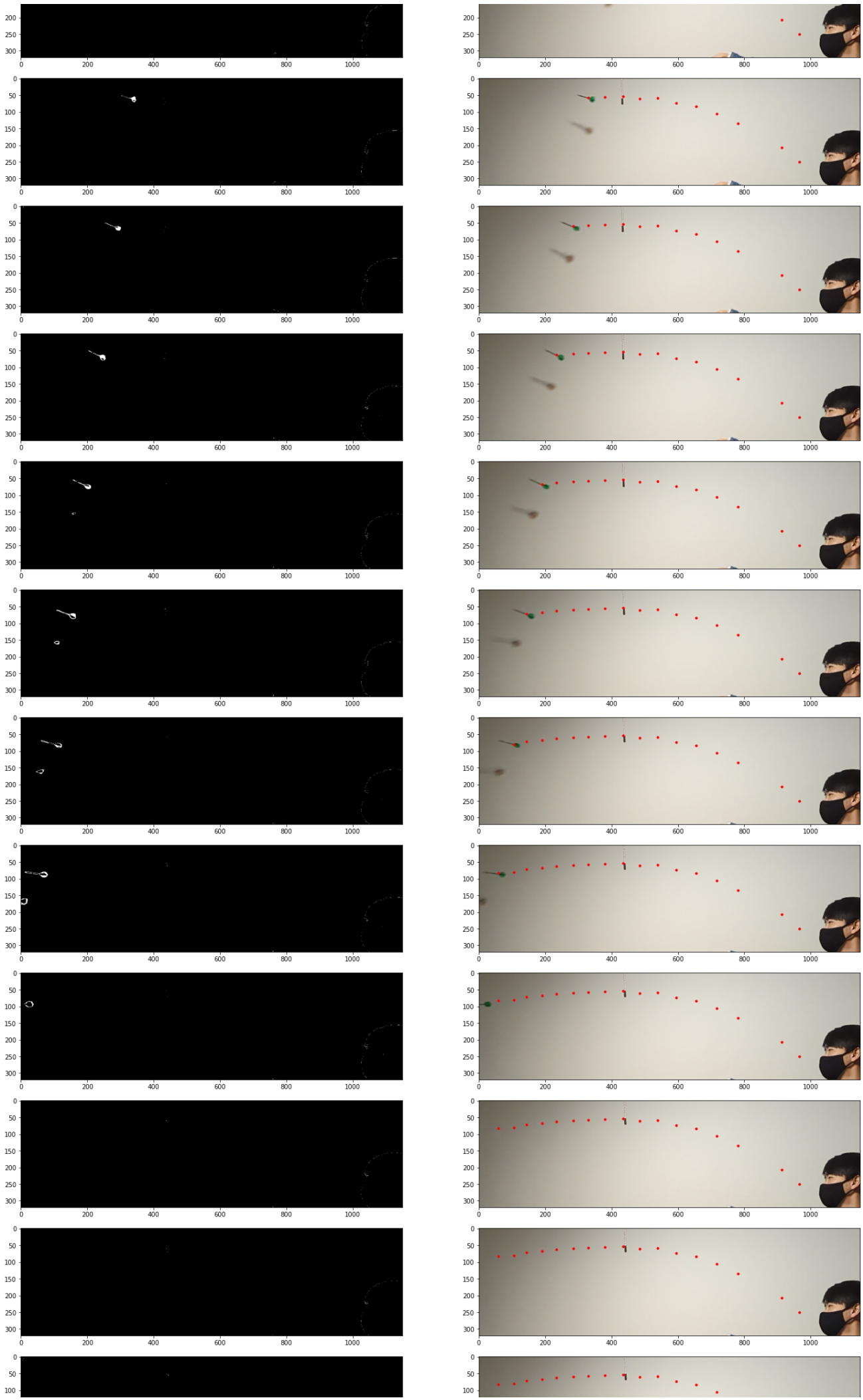
羽検出

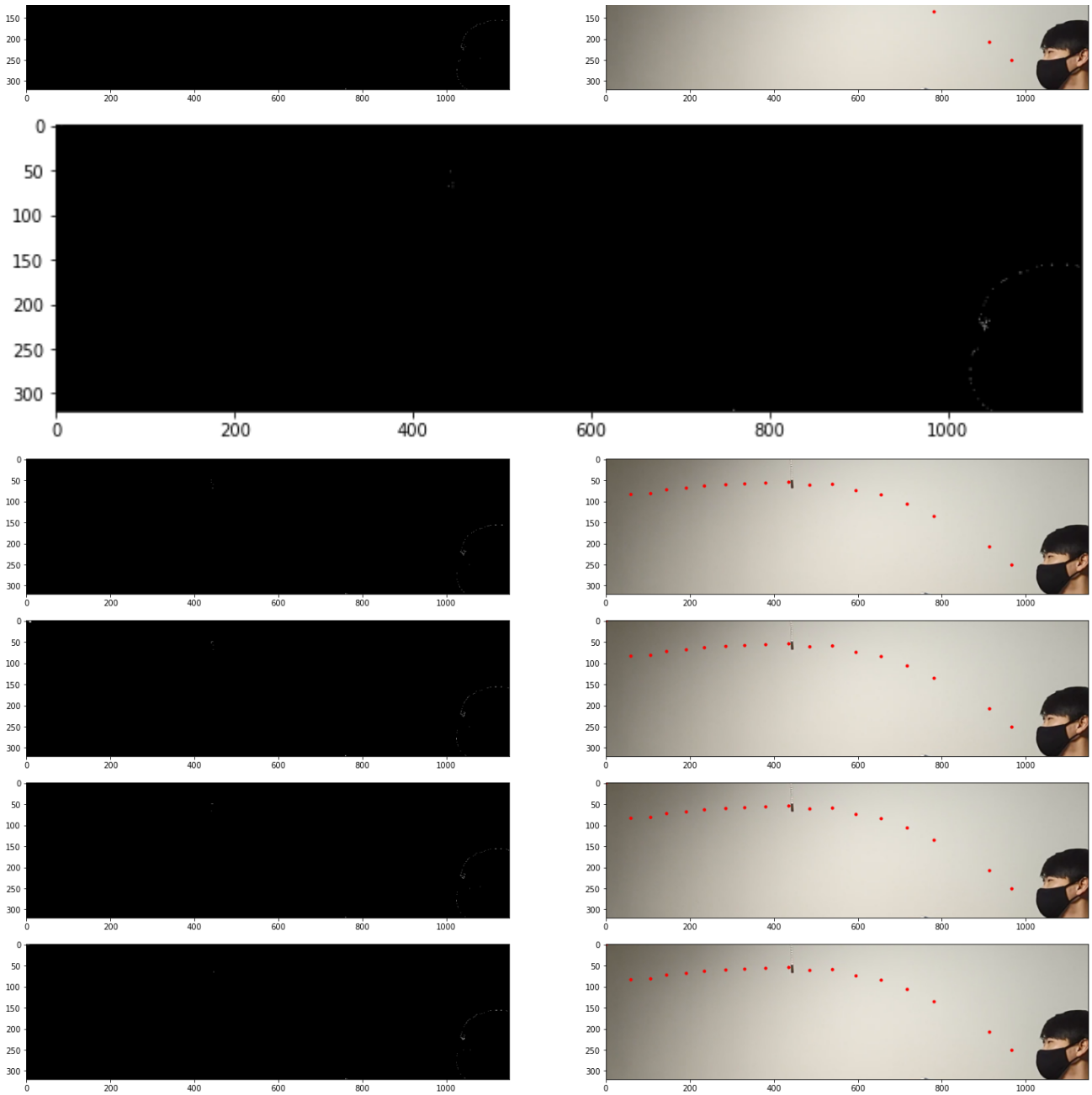
```
trajectory = []
cv2_imgs = []
for idx in range(0, len(frames), 5):
  # print(idx)
  tmp_img = frames[idx][hmin:hmax,wmin:wmax,:][10:,400:,:].copy()

  cond_blue = (0<tmp_img[:,:,0]) & (tmp_img[:,:,0]<100)
  cond_green = 80<tmp_img[:,:,1]
```

```
    cond_red = (0<tmp_img[:,:,2]) & (tmp_img[:,:,2]<100)
    condition = cond_blue & cond_green & cond_red

    img_arrow, est_xtip, est_ytip = extract_arrow(tmp_img, condition, ignore_kernel_size = 20, p = 80
    trajectory.append([est_xtip, est_ytip])

    plt.figure(figsize=(24,24))
    plt.subplot(1,2,1)
    plt.imshow(cv2.cvtColor(img_arrow, cv2.COLOR_BGR2RGB))
    plt.subplot(1,2,2)

    for est_xtip, est_ytip in trajectory:
      if est_xtip==est_xtip:
        cv2.circle(tmp_img, (int(est_xtip), int(est_ytip)), 4, (0, 0, 255), -1)
    cv2_imgs.append(tmp_img)
    tmp_img = cv2.cvtColor(tmp_img, cv2.COLOR_BGR2RGB)
    plt.imshow(tmp_img)

    gif_imgs.append(Image.fromarray(tmp_img))

# gif_imgs[0].save('./video/output.gif', save_all=True, append_images=gif_imgs[1:], optimize=False,
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: RuntimeWarning: More than 20
  from ipykernel import kernelapp as app
```







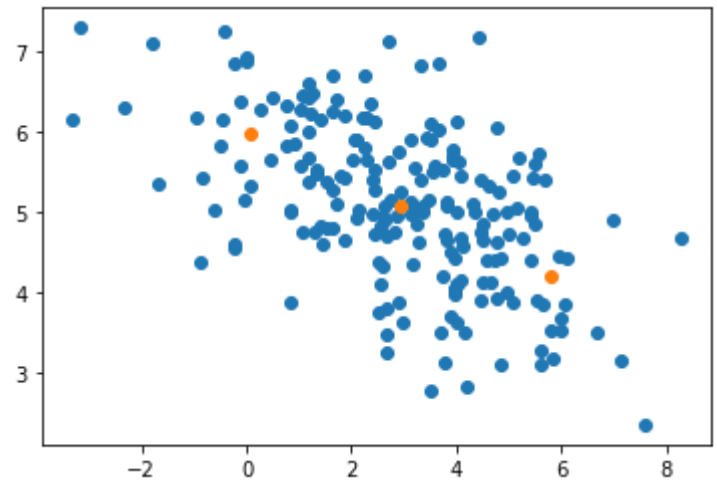
```
# max_idx = np.argmin(np.nan_to_num(np.array(trajectory)[:,1]/5,nan = 1000))
# max_height_x = int(trajectory[max_idx][0])
# max_height_y = int(trajectory[max_idx][1])

# gif_imgs = []
# # for i, img in enumerate(cv2_imgs):
# for i in range(0,len(frames),5):
#   img = cv2_imgs[i].copy()
#   if i >= max_idx:
#     cv2.circle(img, (max_height_x, max_height_y), 36, (255, 255, 255), 1)
#     cv2.circle(img, (max_height_x, max_height_y), 32, (0, 0, 0), 2)
#     cv2.circle(img, (max_height_x, max_height_y), 28, (255, 255, 255), 1)
#     cv2.putText(img,
```

```
#             text='Max Height',
#             org=(max_height_x+40, max_height_y-20),
#             fontFace=cv2.FONT_HERSHEY_SIMPLEX,
#             fontScale=1.0,
#             color=(0, 0, 0),
#             thickness=2,
#             lineType=cv2.LINE_4)
#     drawDashedLine(img, (max_height_x, max_height_y), (max_height_x, img.shape[0]), 8, 2, (255,0,
#   img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
#   plt.figure(figsize=(24,6))
#   plt.imshow(img)
#   gif_imgs.append(Image.fromarray(img))
# # gif_imgs[0].save('./video/output.gif', save_all=True, append_images=gif_imgs[1:], optimize=Fals
```

先端検出

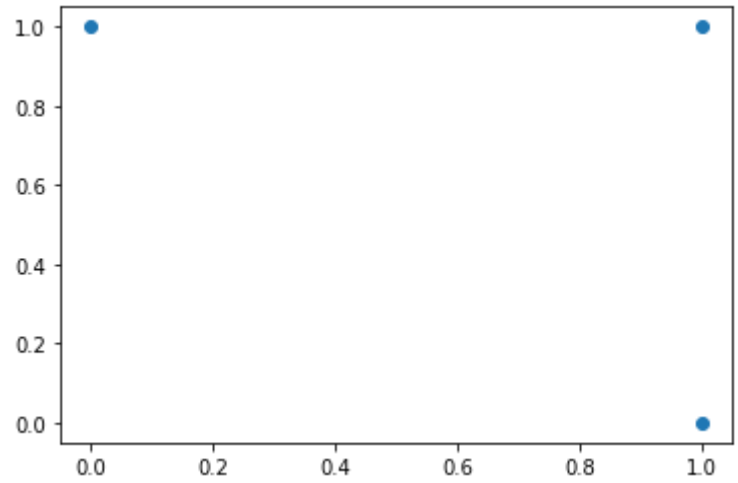
```
from scipy.stats import multivariate_normal
import scipy

mean = np.array([3, 5])
cov = np.array([[4, -1.2], [-1.2, 1]])

# numpy を用いた生成
data = np.random.multivariate_normal(mean, cov, size=200)

plt.scatter(data[:,0], data[:,1])

pcamodel = PCA()
pcamodel.fit(data)
pca_cor = pcamodel.transform(data)
dx, dy = pcamodel.components_[0]

x_mean = np.mean(data[:,0])
y_mean = np.mean(data[:,1])

plt.scatter(x=[x_mean-3*dx, x_mean, x_mean+3*dx], y = [y_mean-3*dy, y_mean, y_mean+3*dy])
```

```
<matplotlib.collections.PathCollection at 0x7f1486086fd0>
```



```
data = np.array([[0,255],[255,255]])
x, y = np.where(data>10)
plt.scatter(x, y)
```

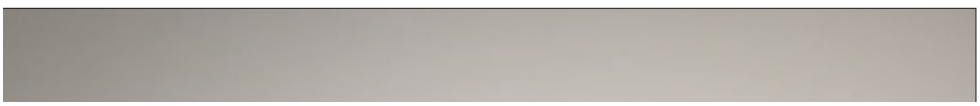
```
<matplotlib.collections.PathCollection at 0x7f199ccf4950>
```

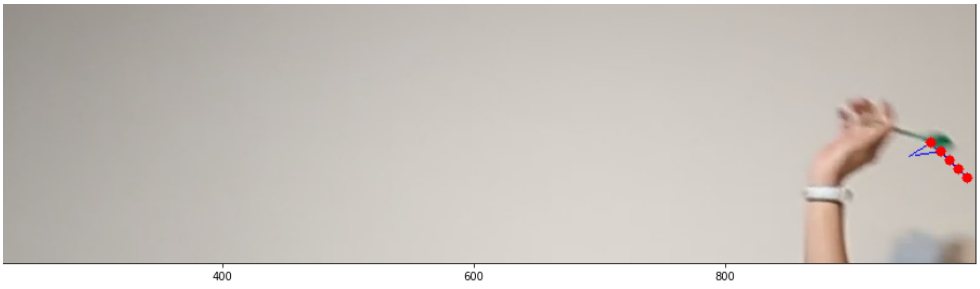


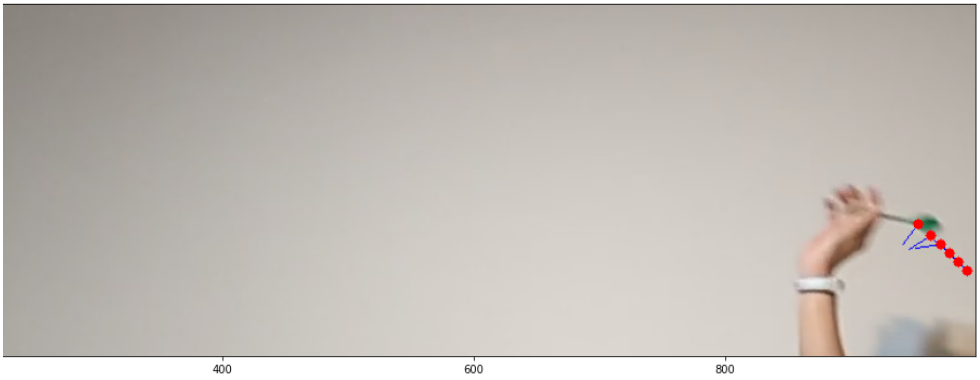

```
x_offset = 200


# cv2_imgs = []
# trajectory = []
# axis_list = []
# gif_imgs = []
# for idx in range(0, len(frames), 1):
#   # print(idx)
#   tmp_img = frames[idx][hmin:hmax,wmin+x_offset:wmax,:].copy()

#   cond_blue = (0<tmp_img[:,:,0]) & (tmp_img[:,:,0]<90)
#   cond_green = 70<tmp_img[:,:,1]
#   cond_red = (0<tmp_img[:,:,2]) & (tmp_img[:,:,2]<90)
#   condition = cond_blue & cond_green & cond_red

#   img_arrow, est_xtip, est_ytip = extract_arrow(tmp_img, condition, ignore_kernel_size = 8, p = 8
#   trajectory.append([est_xtip, est_ytip])

#   if est_xtip == est_xtip:
#     axis = estimate_axis(img_arrow, est_xtip, est_ytip, kernel=10)
#     axis_list.append(axis)
#   else:
#     axis_list.append([[None, None], [None,None]])

#   plt.figure(figsize=(24,24))
#   plt.subplot(1,2,1)
#   plt.imshow(cv2.cvtColor(img_arrow, cv2.COLOR_BGR2RGB))
#   plt.subplot(1,2,2)

#   for (est_xtip, est_ytip), axis in zip(trajectory, axis_list):
#     if est_xtip==est_xtip:
#       cv2.line(tmp_img,tuple(axis[0]),tuple(axis[1]),(255,0,0),1)
#       cv2.circle(tmp_img, (int(est_xtip), int(est_ytip)), 4, (0, 0, 255), -1)

#   cv2_imgs.append(tmp_img)
```

```
#   tmp_img = cv2.cvtColor(tmp_img, cv2.COLOR_BGR2RGB)
#   plt.imshow(tmp_img)

#   gif_imgs.append(Image.fromarray(tmp_img))

# gif_imgs[0].save('./video/output2.gif', save_all=True, append_images=gif_imgs[1:], optimize=False


max_idx = np.argmin(np.nan_to_num(np.array(trajectory)[:,1],nan = 1000))
max_height_x = int(trajectory[max_idx][0])
max_height_y = int(trajectory[max_idx][1])

gif_imgs = []
for i, img in enumerate(cv2_imgs):
  tmp_img = img.copy()
  if i >= max_idx:
    cv2.circle(img, (max_height_x, max_height_y), 36, (255, 255, 255), 1)
    cv2.circle(img, (max_height_x, max_height_y), 32, (0, 0, 0), 2)
    cv2.circle(img, (max_height_x, max_height_y), 28, (255, 255, 255), 1)
    cv2.putText(img,
            text='Max Height',
            org=(max_height_x+40, max_height_y-20),
            fontFace=cv2.FONT_HERSHEY_SIMPLEX,
            fontScale=1.0,
            color=(0, 0, 0),
            thickness=2,
            lineType=cv2.LINE_4)
    drawDashedLine(img, (max_height_x, max_height_y), (max_height_x, img.shape[0]), 8, 2, (255,0,25
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  plt.figure(figsize=(24,6))
  plt.imshow(img)
  gif_imgs.append(Image.fromarray(img))
gif_imgs[0].save('./video/output2.gif', save_all=True, append_images=gif_imgs[1:], optimize=False,
```

```
7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning: More than 20 figures have been opene
```



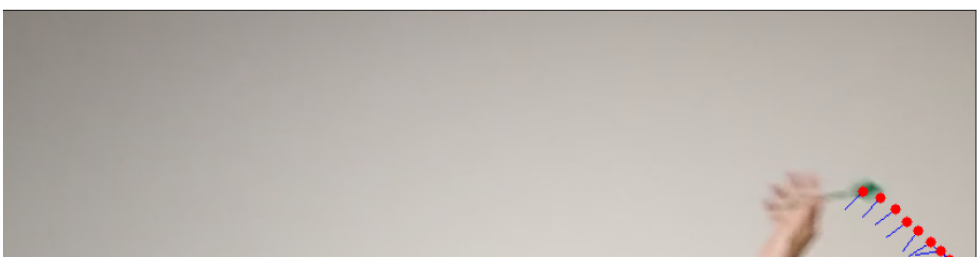

400
600
800


400
600
800


400
600
800


400
600
800


400
600
800




400
600
800


400
600
800


400
600
800

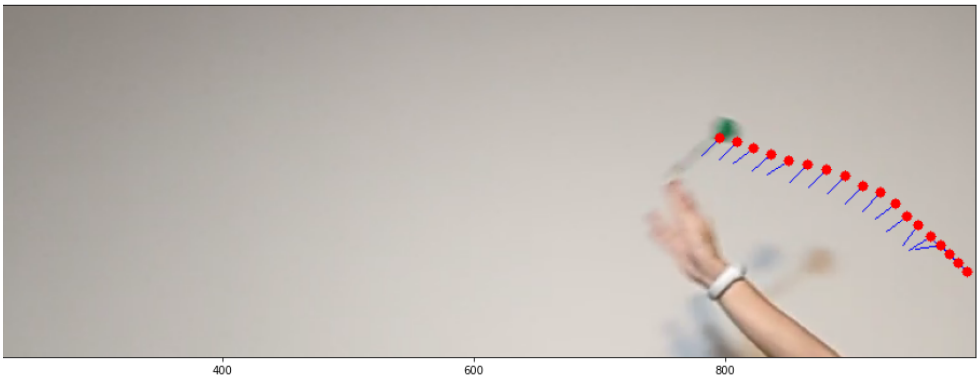

400
600
800


400
600
800


400
600
800

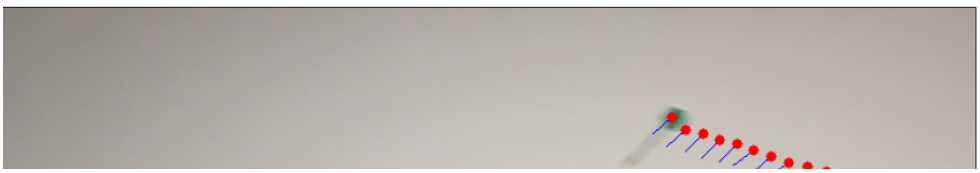

400
600
800

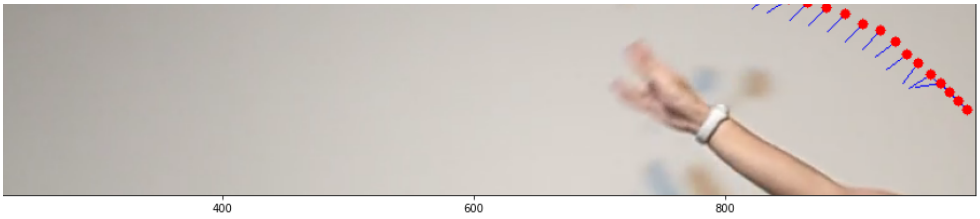

400
600
800


400
600
800


400
600
800

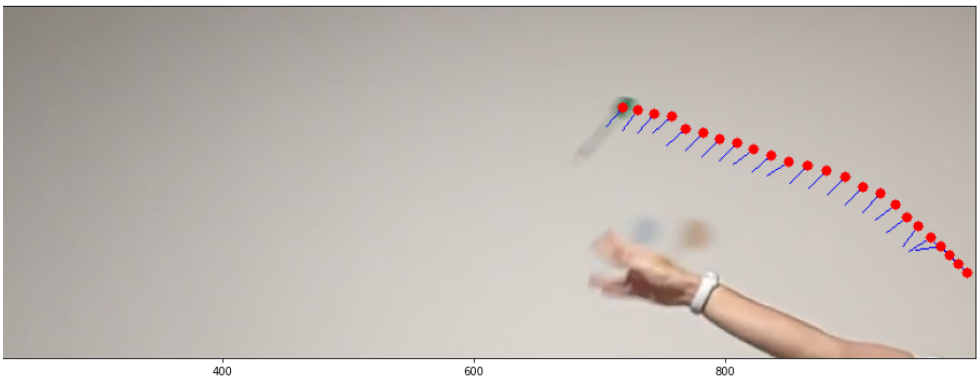

400
600
800

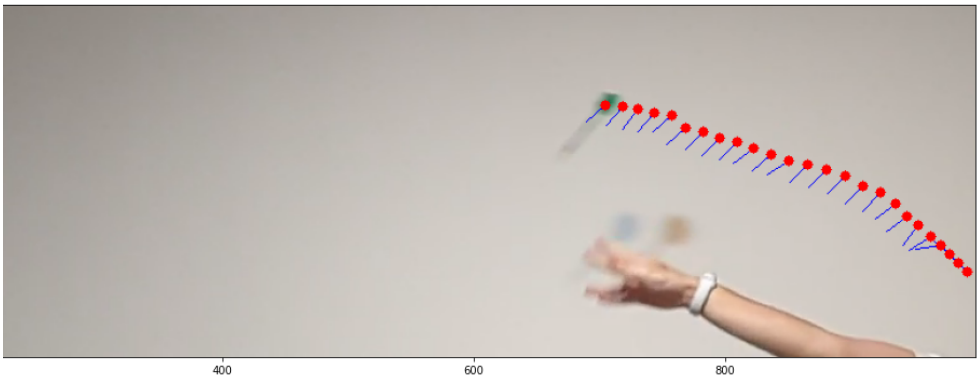


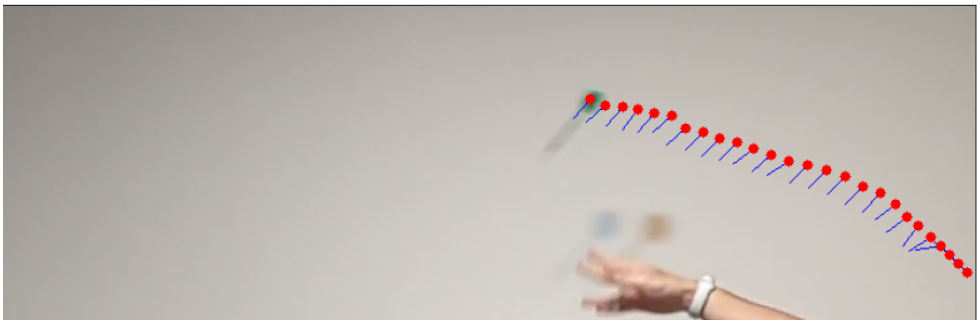

400
600
800

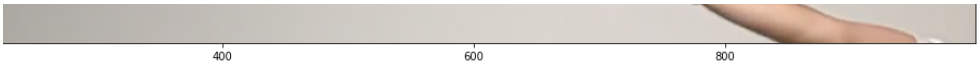

400
600
800

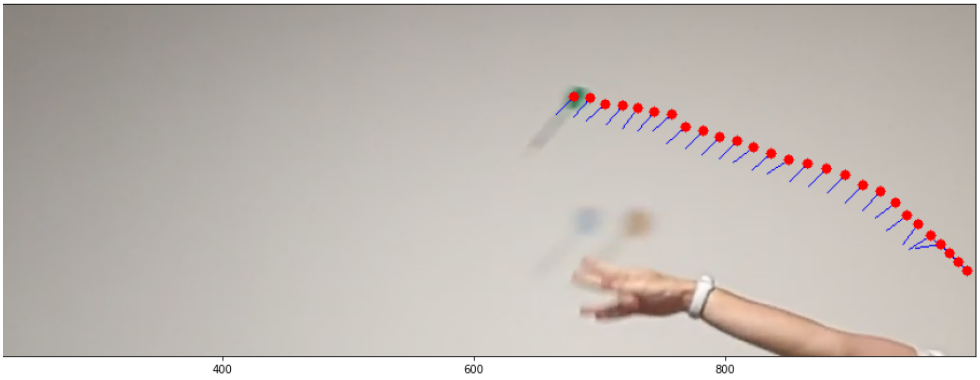

400
600
800

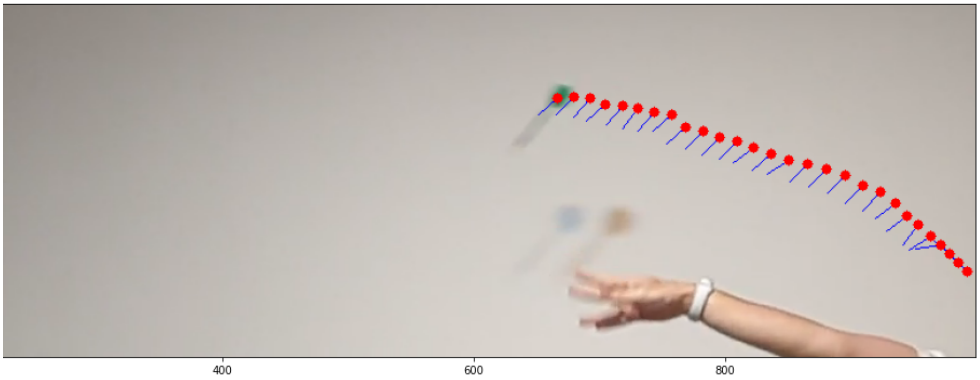

400
600
800

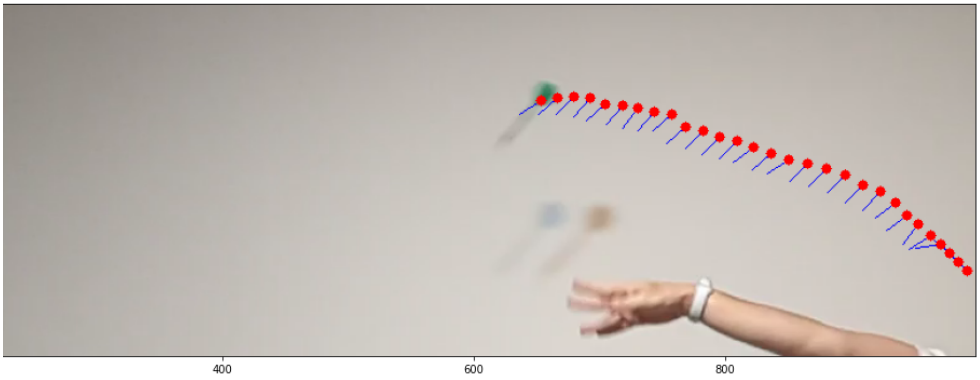

400
600
800

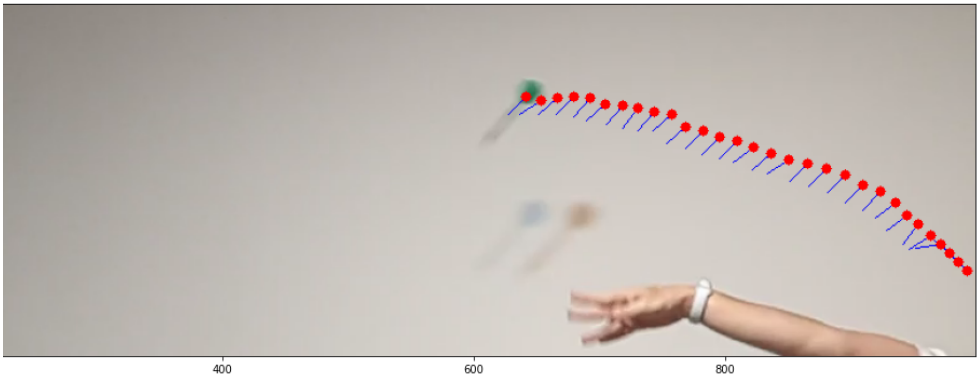


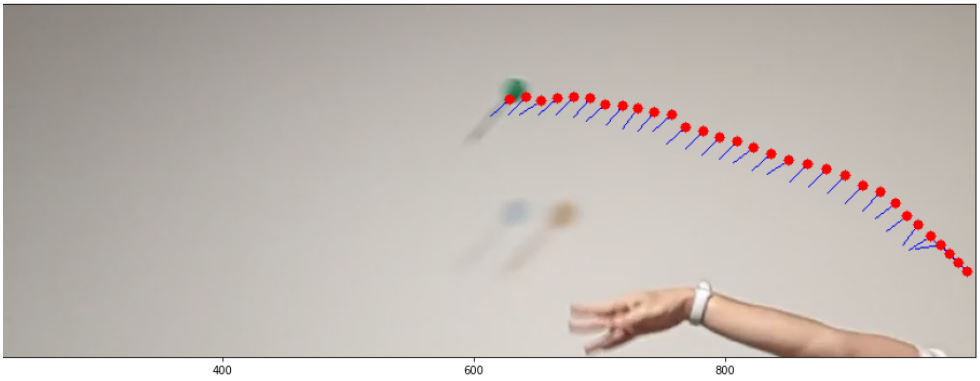

400
600
800


400
600
800


400
600
800


400
600
800


400
600
800


400
600
800