OpenCV

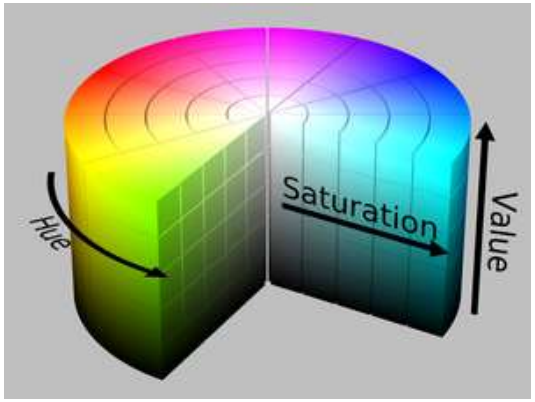
カラー認識

1.HSVの解説
2.Mask画像の解説
3.コード見ながら解説

HSVとは

RGBは、赤（Red)、緑(Green)、青(Blue)の要素の組み合わせに対し、HSVは、色相(Hue)、彩度(Saturation)、明度(Value)の組み合わせで表現します。

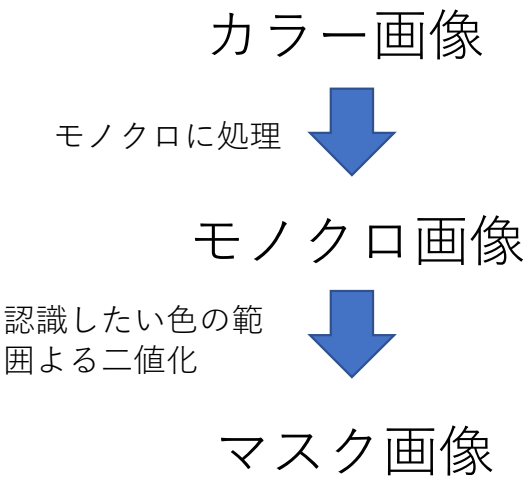
- OpenCVだと、色相、彩度、明度はそれぞれ0～179,0～255, 0～255の範囲で設定される



なぜHSVを用いる？

簡単に言えばカラー画像を処理を軽くするために、白と黒であらわす必要があるとき、カラー画像から色をHSVのほうがうまく抽出できることや、HSVの表す色が人の目が認識する色と近いなど理由があります。

マスク処理とマスク画像



マスク処理とは…画像中の特定の領域だけ抜き出す処理。

閾値　…境界となる値。画像処理ではモノクロ画像を真っ白と真っ黒に分けるのに閾値を設ける。
閾値によって、値を2つ(白と黒)に分けることを二値化という

マスク画像　…二値化処理を施した画像

つまり、カラー画像の中でほしい色の部分だけを白色にし、余計な色を黒で塗りつぶす

- OpenCVだとモノクロにせずとも、色の範囲を決めてマスク画像に簡単にできる

コードの解説

```
def red_detect(img):
    # HSV色空間に変換
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)          #OpenCVの画像（映像）はRGBではなくBGRの順番で格納される

    # 赤色のHSVの値域1
    hsv_min = np.array([0, 180, 50])
    hsv_max = np.array([15, 255, 255])
    mask1 = cv2.inRange(hsv, hsv_min, hsv_max)          #inRange関数は閾値とするHSVの最小と最大の範囲を決めて二値化してくれる

    # 赤色のHSVの値域2
    hsv_min = np.array([165, 180, 50])
    hsv_max = np.array([179, 255, 255])
    mask2 = cv2.inRange(hsv, hsv_min, hsv_max)          #HSVの赤は0~30と150~179をとるので、二つ値域を設けている。
                                                            つまり、複数色の検知も可能
    return mask1 + mask2                                #二値化した画像を返してあげる
```

続き

```
# ブロブ解析
def analysis_blob(binary_img):
    global label
    global data
    global max_index
    global center
    # 2値画像のラベリング処理
    label = cv2.connectedComponentsWithStats(binary_img)   #このメソッドは返り値がめっちゃ多いので、
```

https://axa.biopapyrus.jp/ia/opencv/object-detection.htmlを参考にするとわかりやすい

外部の関数で値を使いたいから、グローバルに

```
    # ブロブ情報を項目別に抽出
    n = label[0] - 1
    data = np.delete(label[2], 0, 0)
    center = np.delete(label[3], 0, 0)
```

- ブロブ解析とは、二値化した画像に対して、物体の形や位置など分析する手法のこと
- ラベリング処理とは二値画像中の白い地域に番号をつけることで、物体が複数あっても区別できるようにする処理

dataには行列の形式で、
ブロブ（和訳…塊　つまり映像中の物体のこと）の面積と
外接矩形（がいせつくけい…物体に接するように引いた四角）
の幅や高さ、左上頂点の座標が入れられている。

centerにはブロブの中心座標が入れられている。

詳しくはhttps://algorithm.joho.info/programming/python/opencv-labeling-blob-py/

エラー対策

配列の次元数を取得

dimensions = data.shape　　numpyのshapeメソッドにより、dataに情報が何行何列で格納してるか調べる

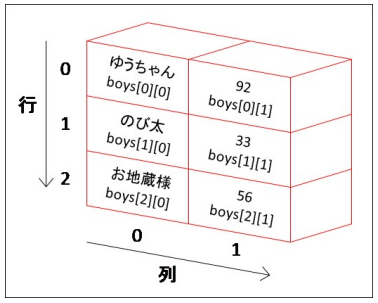
if dimensions:　dimensionsが空ではない

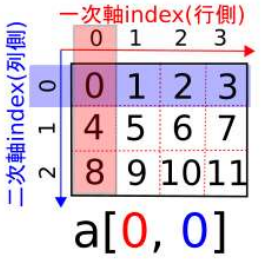
　if len(dimensions) >= 2:　2次元以上であること。
※dataには行列(二次元形式)で情報が入っているから

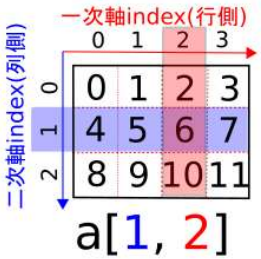
dim2nd = dimensions[1]　2次元目の要素数を確認
[1]で行列の2行目（2次元目）の情報だけを取得できる

if dim2nd >= 5:　2次元目の要素数5以上を確認
Dataにはブロブの面積と外接矩形に対する合計5つの情報が入れられているため

＊イメージ







★ [:,4]は全ての行の５列を表す。

なぜ行と列を数える際に０から始まるのか？
(プログラム全般、0から始まる)

(ネットにあったわかりやすい例)

例えば「1m間隔で5本の木を植えます。最初の木と最後の木の距離は幾らですか？」

- 答えは4。

1□2□3□4□5　　木の数は５本なのに間の□の数は4つ。
これが多くのバグを生む。

これを0から始めると

- 0□1□2□3□4　　□の数は4と一致する。

つまり、数え間違いによるバグ対策

面積最大の物体の情報取得

```
# ブロブ面積最大のインデックス
        max_index = np.argmax(data[:, 4])

        # 面積最大ブロブの情報格納用
        maxblob = {}

    # 面積最大ブロブの各種情報を取得
        maxblob["upper_left"] = (data[:, 0][max_index],
                                 data[:, 1][max_index]) # 左上座標
        maxblob["width"] = data[:, 2][max_index]  # 幅
        maxblob["height"] = data[:, 3][max_index]  # 高さ
        maxblob["area"] = data[:, 4][max_index]   # 面積
        maxblob["center"] = center[max_index]  # 中心座標

        return maxblob
```

Numpyのメソッド「argmax」により、面積が最大のところが何行何列目かだけ格納する

各情報を辞書型で入れることにより、参照しやすくする

情報を格納する

```
def main():

    # カメラのキャプチャ
    cap = cv2.VideoCapture(0)
```

カメラの映像の情報を取得
(0としているのは何番目のカメラの情報を取得するかを表している)

```
while(cap.isOpened()):
```

.isOpened()は、capに映像の情報が格納されているとき、Trueとなる。
つまり、while Trueでループ処理となる。

```
# フレームを取得
        ret, frame = cap.read()
```

retには情報を読み込めたかどうか,
False または Trueが入る。
Frameには読み込んだ映像の情報が入る。

```
 # 赤色検出
        mask = red_detect(frame)
```

映像の情報を関数で処理し、二値画像にする。

```
try:
    # マスク画像をブロブ解析（面積最大のブロブ情報を取得）
    target = analysis_blob(mask)

    # 面積最大ブロブの中心座標を取得
    center_x = int(target["center"][0])
    center_y = int(target["center"][1])

    # フレームに面積最大ブロブの中心周囲を円で描く
    cv2.circle(frame, (center_x, center_y), 50, (0, 200, 0),
               thickness=3, lineType=cv2.LINE_AA)

    # 結果表示
    cv2.imshow("Frame", frame)
    cv2.imshow("Mask", mask)

except ValueError:
    continue
```

二値画像をブロブ解析にかける

引数…　(映像のデータ,円の中心,円の半径,円の色(青,緑,赤),円の厚さ, 線を描写するアルゴリズム)

カメラの映像とマスク処理した映像を表示

ValueErrorというエラーが発生したら、処理を飛ばす

tryには、何かあったときエラーが発生するコードを
exceptには、tryに入っているコードがエラーが発生したとき、どのような処理を行うかを記述する