In [1]:
```
# Nama  : Avima Haamesha
# NIM   : 10219084
```

UTS KSIC

Terdapat sumber berupa animasi bandul fisis berosilasi. Dengan menggunakan image processing akan dicari periode bandul berosilasi.

Extract Warna Tertentu

Ekstraksi warna tertentu dilakukan dengan baris program berikut.

```
def process_func(frame):
        # extract specific color range
        extracted, mask = extract_color(frame, lower=[0,127,127], upper=[179,255,255])
        ...
```

fungsi `extract_color()` didefinisikan sebagai berikut.

```
def extract_color(frame, lower=[0,0,0], upper=[0,0,0]):
        # convert BGR to HSV color space
        hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)

        # get masking frame in 1/0 intensity
        lower, upper = np.array(lower), np.array(upper)
        mask = cv.inRange(hsv, lower, upper)

        # extract frame using masking
        res = cv.bitwise_and(frame, frame, mask=mask)

        return res, mask
```

Gambar original yang diload menggunakan tipe BGR color space, dikonversi ke HSV color space. Untuk mengambil warna tertentu, ditentukan terlebih dulu batas bawah dan atasnya. Berikut ini adalah hasil screenshot yang menunjukkan perolehan `mask` dan `res` yang diperoleh. `mask` adalah citra dengan intensitas 1/0, warna dalam range yang sbelumnya ditentukan akan memiliki nilai 1. Sedangkan, `res` adalah hasil perolehan operasi `bitwwise_and()` menggunakan masking `mask` .

Operasi Erosi dan Dilasi untuk Mengurangi Noise

Dari perolehan `mask` , selanjutnya dilakukan operasi erosi untuk menghilangkan noise berukuran kecil. Setelah noise hilang, dilakukan operasi dilasi sehingga daerah berintensitas 1 dengan ukuran cukup besar akan di-dilasi ke ukuran lebih luas.

```
def process_func(frame):
        ...
        # remove noise & strengthen the region
        kernel = np.ones( (3,3), np.uint8 )
        erodila = cv.erode(mask, kernel, iterations=2)
        erodila = cv.dilate(erodila, kernel, iterations=3)
        ...
```

Di sini digunakan kernel berukuran 3x3 yang semua nilainya adalah 1. Pada operasi **erosi**, nilai citra hasil akan sama dengan nol jika terdapat nilai nol di citra asal yang overlap dengan kernel tersebut. Pada operasi **dilasi**, nilai citra hasil akan bernilai 1 jika terdapat nilai 1 pada citra asal yang overlap dengan kernel.

```
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |
```

Menemukan Kontur Objek

Untuk menemukan kontur atau tepian objek digunakan fungsi `findContours()` . Fungsi `grab_contours()` berfungsi untuk menyimpan kontur-kontur yang ada dalam suatu array.

```
def process_func(frame):
        ...
        # find & draw contour
        cnts = cv.findContours(erodila, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
        cnts = grab_contours(cnts)
        ...
```

Visualisasi Posisi Centroid

Setelah diperoleh array of contours, selanjutnya dimanfaatkan baris program berikut untuk menggambarkan persegi minimum objek dan menggambarkan titik centroid.

```
def process_func(frame):
        ...
        area = get_area(cnts)
        centroid_frame, box = draw_min_rectangle(frame, cnts)
        centroid_frame, centroid = draw_centroid(frame, cnts)
        ...
```

Perolehan persegi terkecil diperoleh dari fungsi `draw_min_rectangle()` . Variabel `box` akan menyimpan 4 titik di sudut-sudut persegi. Variabel `arr` akan menyimpan kumpulan `box` dari setiap kontur yang terdeteksi.

```
def draw_min_rectangle(frame, cnts):
        arr = []    # store arr of box for every contour detected
        for cnt in cnts:
            rect = cv.minAreaRect(cnt)
            box = cv.boxPoints(rect)
            box = np.int0(box)
            arr.append(box)

        for (a,b,c,d) in arr:
            cv.line(frame, pt1=a, pt2=b, color=(0,255,0), thickness=2)
            cv.line(frame, pt1=b, pt2=c, color=(0,255,0), thickness=2)
            cv.line(frame, pt1=c, pt2=d, color=(0,255,0), thickness=2)
            cv.line(frame, pt1=d, pt2=a, color=(0,255,0), thickness=2)

        return frame, arr
```

Menyimpan Data

Untuk menentukan periode, diperlukan beberapa data yang harus disimpan, setidaknya centroid dan area.

```
def process_func(frame):
        ...
        myData.store_centroid(centroid[0])  # centroid[0] because the array store centroid of many objects
        myData.store_area(area)
        myData.store_period()
        ...
```

Menentukan Periode

Data-data disimpan pada suatu class dengan format sebagai berikut.

```
class Data():
        def __init__(self):
            self.fps = None
            self.time = [None,None] # store time start & end when in oscilataion
            self.period = None
            self.pos = [(None,None)]    # storing centroid (cx, cy)
            self.area = None
        ...
```

Tiga fungsi di atas didefinisikan dalam suatu `Class Data` . Akan disimpan posisi awal objek terlebih dahulu, yaitu ketika data `self.pos = [(None,None)]` . Setelah data centroid awal disimpan. Array `self.pos` di-append tuple `(None,None)` . Tuple ini berfungsi menyimpan centroid yang terdeteksi di setiap frame. Kondisi if pertama hanya bekerja satu kali di awal program. Setelah itu, di frame selanjutnya, program di lingkup else yang akan dijalankan. Baris ini akan menyimpan data centroid di setiap frame.

```
class Data:
        ...
        def store_centroid(self, centroid):
            if self.pos[0] == (None,None):  # store initial value
                self.pos[0] = centroid
                (self.pos).append( (None,None) )    # i only need to store 2 centroid data
            else:
                self.pos[1] = centroid
            print("centroid = [%s,%s]" %(self.pos[0], self.pos[1]))
        ...
```

Kemudian untuk menentukan periode, digunakan konsep selisih waktu ketika benda mencapai posisi awal. Posisi awal adalah kondisi setelah data centroid pertama disimpan, atau sama saja ketika `self.time[0] == None` . Kondisi `self.pos[1] == self.pos[0]` terpenuhi ketika centroid frame terkini kembali ke posisi awal (centroid awal).

Apabila benda kembali ke centroid awal, maka simpan waktu saat itu. Dengan demikian, periode dapat dihitung dengan selisih waktu. Di akhir, `self.time[0]` diatur menjadi `None` kembali. Agar di iterasi saat frame selanjutnya, kodisi if pertama dipenuhi kembali untuk memperbarui nilai `self.time[0]` . Dengan demikian, nilai periode akan selalu diperbarui setiap satu kali ayunan.

```
class Data:
        ...
        def store_period(self):
            if self.time[0] == None:
                self.time[0] = getTickCount()
            # store period when in peak
            elif self.pos[1] == self.pos[0]:
                self.time[1] = getTickCount()
                self.period = (self.time[1] - self.time[0]) / getTickFrequency()
                print("T = %s sekon" %self.period)
                self.time[0] = None     # so that make it go back to first condition
            print("time     = [%s,%s]" %(self.time[0], self.time[1]))
        ...
```

Hasil Program

Screeshot

Video