# Cel laboratorium

Celem laboratorium była poprawa zaimplementowanych 5 wybranych prostych deskryptorów kształtu przedstawionych przedstawianych prowadzącemu na poprzednich zajęciach oraz porównanie wyników..

# Przebieg laboratorium

Po przedstawieniu prowadzącemu wyników z poprzedniego zadania, stwierdził on, że wykonane zadanie należy poprawić. Prowadzący wyjaśnił dokładnie jak zadanie powinno być wykonane, a celem laboratorium było poprawienie implementacji oraz porównanie wyników z poprzednich zajęć.

Dla każdej klasy wybrano po 1 obrazie jako obraz wzorcowy (część ucząca). Dla tych obrazów były obliczane wartości deskryptorów. Następnie dla pozostałych obrazów (część testowa) były również obliczane wartości deskryptorów.

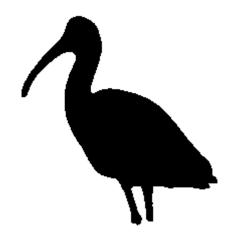
Po dokonaniu obliczeń przez deskryptory dokonywany był pomiar odległości euklidesowej dla danego deskryptora dla każdego obrazu testowego względem wszystkich obrazów wzorcowych. Klasa obrazu wzorcowego z najniższą odległością odzwierciedlała tą predykcyjną.

Wybrane obrazy dla poszczególnych klas:

Gambles quail



Glossy ibis



Greator sage grous



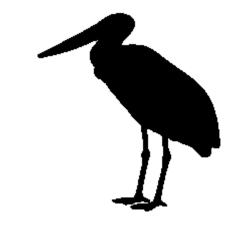
• Hooded merganser



• Indian vulture



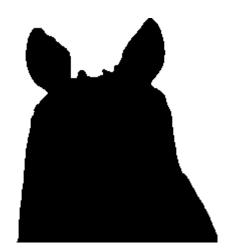
Jabiru



• King eider



• Long eared owl



• Tit mouse



Touchan



# Kod programu

## Importy oraz stałe:

```
from typing import Tuple
import numpy as np
import pandas as pd
import os
import cv2

IMAGES_DIR = './images'
```

#### Funkcje pomocnicze:

```
def read_images() -> dict:
       images['black'].append([])
def select representatives numbers(images: dict) -> list:
       'gambles-quail': 6,
       'greator-sage-grous': 5,
       'king-eider': 8,
       'long-eared-owl': 4,
  return list(map(lambda number: number - 1,
def train test split(images: dict, representative numbers: list) -> Tuple:
       for image index in range(len(images['black'])):
ex])
```

```
def get_list_of_dirs(dirs: list, sub_dir_name: str = 'color'):
def convert_image to black(image: np.array):
   image gray = cv2.cvtColor(image, cv2.COLOR RGB2GRAY)
cv2.THRESH BINARY INV)
def convert_image_to_contour(image: np.array):
  image gray = cv2.cvtColor(image, cv2.COLOR RGB2GRAY)
cv2.THRESH BINARY INV + cv2.THRESH OTSU)
  contours, hierarchy = cv2.findContours(image black and white, cv2.RETR TREE,
cv2.CHAIN APPROX SIMPLE)
color=(0, 0, 0), thickness=1)
def calc_descriptors_for_images(train_images: dict, test_images: dict) -> Tuple:
       image_type = 'black' if descriptor.__name__ != perimeter_descriptor.__name__
train_images[image_type]]
def predict based on distance(test descriptor result: float,
```

```
results = np.array([calc_distance(train_descriptor_result,
test_descriptor_result) for train_descriptor_result in train_descriptor_results])

return np.argmin(results)
```

### Deskryptory:

```
def area_descriptor(image: np.array) -> float:
   return np.array(list(zip(*np.where(image == 0)))).shape[0]
def perimeter_descriptor(contour_image: np.array) -> float:
def roundness descriptor(image: np.array) -> float:
  return perimeter_descriptor(image) ** 2 / (4 * np.pi * area_descriptor(image))
def compactness descriptor(image: np.array) -> float:
def eccentricity_descriptor(image: np.array) -> float:
  boundary = {'x_min': image.shape[0], 'x_max': 0, 'y_min': image.shape[1],
           if x < boundary['x min']:</pre>
               boundary['y_min'] = y
           if y > boundary['y_max']:
boundary['y_min']]
   return np.max(rectangle boundary) / np.min(rectangle boundary)
def calc distance(i, j):
  return np.sqrt(np.abs(i ** 2 - j ** 2))
```

### Główny kod programu:

```
if __name__ == '__main__':
    descriptors = [area_descriptor, perimeter_descriptor, roundness_descriptor,
compactness_descriptor, eccentricity_descriptor]
```

```
descriptor results = pd.DataFrame(columns=['predicted', 'real', 'score'])
enumerate(test results[descriptor name]):
               row = pd.Series({'predicted': predicted_class, 'real': class_iter,
               descriptor_results = pd.concat([descriptor_results,
          scores_df = pd.DataFrame(np.array([[*score_by_class.keys()],
100):0.4f}%')
descriptor results.shape[0] * 100
```

# Rezultaty

## Poprzednie laboratoria

Poniżej wyniki z wielokrotnego uruchamiania programu dla całego zbioru z poprzedniego programu:

```
{'area_descriptor': 0.4, 'perimeter_descriptor': 0.1, 'roundness_descriptor': 1.0,
'compactness_descriptor': 1.0, 'eccentricity_descriptor': 1.0}

{'area_descriptor': 0.4, 'perimeter_descriptor': 1.0, 'roundness_descriptor': 1.0,
'compactness_descriptor': 1.0, 'eccentricity_descriptor': 1.0}

{'area_descriptor': 0.0, 'perimeter_descriptor': 0.2, 'roundness_descriptor': 1.0,
'compactness_descriptor': 1.0, 'eccentricity_descriptor': 1.0}

{'area_descriptor': 0.9, 'perimeter_descriptor': 0.2, 'roundness_descriptor': 1.0,
'compactness_descriptor': 1.0, 'eccentricity_descriptor': 1.0}

{'area_descriptor': 0.2, 'perimeter_descriptor': 1.0, 'roundness_descriptor': 1.0,
'compactness_descriptor': 0.2, 'perimeter_descriptor': 0.1, 'roundness_descriptor': 1.0,
'compactness_descriptor': 0.2, 'perimeter_descriptor': 0.1, 'roundness_descriptor': 1.0,
'compactness_descriptor': 0.7, 'perimeter_descriptor': 1.0, 'roundness_descriptor': 1.0,
'compactness_descriptor': 0.7, 'perimeter_descriptor': 1.0, 'roundness_descriptor': 1.0,
'compactness_descriptor': 1.0, 'eccentricity_descriptor': 1.0}
```

Wyniki dla poszczególnych klas wraz z deskryptorami:

| area\_descriptor | perimeter\_descriptor | roundness\_descriptor | compactness\_descriptor | eccentricity descriptor |

:	:	:	:	: -	
:					
0	0.1	0.3	1	1	1
1	0.3	1	1	1	1
2	0.2	0.1	1	1	1
3	0.6	0.1	1	1	1
4	1	0.1	1	1	1
5	0.2	0	1	1	1
6	0.2	1	1	1	1
7	0.1	0.1	1	1	1
8	0.4	1	1	1	1
9	0.1	0.7	1	1	1

# Obecne wyniki

Descriptor: area\_descriptor

```
| class | score | |
|---|---|---|
| 0 | 1 | 55.5556% |
| 1 | 2 | 22.2222% |
| 2 | 3 | 44.4444% |
| 3 | 4 | 11.1111% |
```

```
| 4 | 5 | 44.4444% |
| 5 | 6 | 33.3333% |
| 6 | 7 | 11.1111% |
| 7 | 8 | 55.5556% |
| 8 | 9 | 11.1111% |
| 9 | 10 | 11.1111% |
```

Score for whole dataset: 30.0000%

### Descriptor: perimeter\_descriptor

```
| | class | score |
|---:|:------|
| 0 |
        1 | 22.2222% |
| 1|
        2 | 22.2222% |
| 2 |
        3 | 88.8889% |
| 3 |
        4 | 11.1111% |
| 4 |
        5 | 44.4444% |
| 5 |
        6 | 11.1111% |
| 6 |
        7 | 11.1111% |
| 7 |
        8 | 0.0000% |
| 8 |
        9 | 0.0000% |
| 9 |
        10 | 0.0000% |
```

Score for whole dataset: 21.1111%

### Descriptor: roundness\_descriptor

```
| | class | score |
|---:|:------|
        1 | 55.5556% |
| 0 |
        2 | 22.2222% |
| 1 |
| 2 |
        3 | 44.4444% |
| 3 |
        4 | 11.1111% |
        5 | 44.4444% |
| 4 |
| 5 |
        6 | 33.3333% |
        7 | 11.1111% |
| 6 |
| 7 |
        8 | 55.5556% |
| 8 |
        9 | 11.1111% |
        10 | 11.1111% |
| 9 |
```

Score for whole dataset: 30.0000%

### Descriptor: compactness\_descriptor

```
| | class | score |
|---:|------|
| 0 |
        1 | 55.5556% |
| 1 |
        2 | 22.2222% |
| 2 |
        3 | 44.4444% |
| 3 |
        4 | 11.1111% |
        5 | 44.4444% |
| 4 |
| 5 |
        6 | 33.3333% |
        7 | 11.1111% |
| 6 |
```

```
| 7 | 8 | 55.5556% |
| 8 | 9 | 11.1111% |
| 9 | 10 | 11.1111% |
```

Score for whole dataset: 30.0000%

Descriptor: eccentricity\_descriptor

```
| | class | score |
|---:|:------|
        1 | 22.222% |
| 0 |
        2 | 0.0000% |
| 1 |
| 2 |
        3 | 0.0000% |
        4 | 0.0000% |
| 3 |
| 4 |
        5 | 33.3333% |
| 5 |
        6 | 11.1111% |
        7 | 0.0000% |
| 6 |
| 7 |
        8 | 55.5556% |
| 8 |
        9 | 11.1111% |
       10 | 22.2222% |
| 9 |
```

Score for whole dataset: 15.5556%

I	class   area_descriptor	perimeter_descriptor	roundness_descriptor	compactness_descriptor	eccentricity_descriptor	score
I:	: :	:	:	:	:	:
ΙO	1   55.5556%	22.2222%	55.5556%	55.5556%	22.2222%	42.2222
1 1	2   22.2222%	22.2222%	22.2222%	22.2222%	0.0000%	17.7778
1 2	3   44.4444%	88.8889%	44.4444%	44.4444%	0.0000%	44.4444
1 3	4   11.1111%	11.1111%	11.1111%	11.1111%	0.0000%	8.88888
1 4	5   44.4444%	44.4444%	44.4444%	44.4444%	33.3333%	42.2222
1 5	6   33.3333%	11.1111%	33.3333%	33.3333%	11.1111%	24.4444
1 6	7   11.1111%	11.1111%	11.1111%	11.1111%	0.0000%	8.88888
1 7	8   55.5556%	0.0000%	55.5556%	55.5556%	55.5556%	44.4445
1 8	9   11.1111%	0.0000%	11.1111%	11.1111%	11.1111%	8.88888
1 9	10   11.1111%	0.0000%	11.1111%	11.1111%	22.2222%	11.1111

Scores for all classes by descriptor

29.99999
21.11110
29.99999
29.99999
15.55555

### Porównanie

W poprzednich laboratoriach wszystkie obrazy były traktowane jako uczące i testowe. Te laboratoria wymagały wybrania jednego obrazu wzorcowego z każdej klasy, a następnie testowanie na pozostałch.

Wyniki nie są już 100% tak jak w poprzednim przypadku, ponieważ nie testujemy na danych uczących.

Najsłabiej poradziła sobie klasa numer 9 (Tit mouse), a najlepiej klasa 3 (Greator sage grous) oraz 8 (Long eared owl). Pozostałe klasy poradziły sobie różnie: niektóre lepiej, niektóre gorzej.

Patrząc się na łączną dokładność deskryptorów dla wszystkich klas, poradziły sobie one podobnie i żaden z nich nie odstawał znacznie od pozostałych.