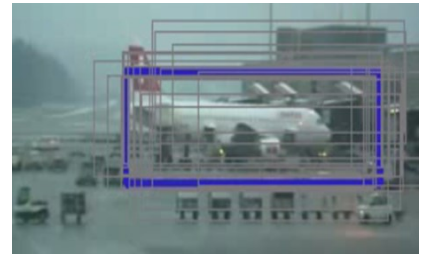
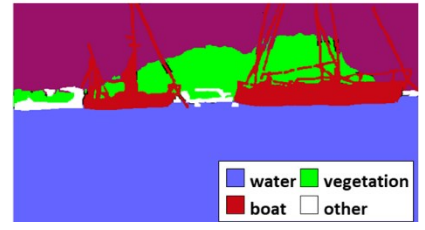








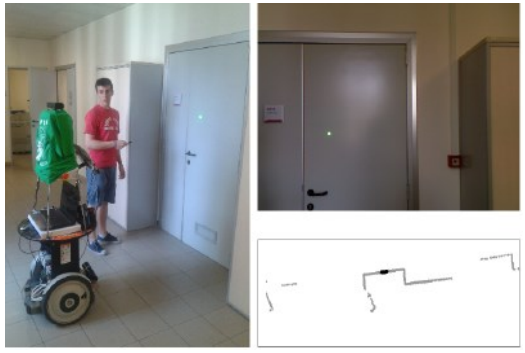
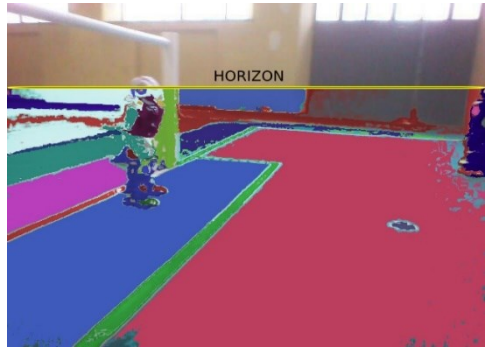
UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

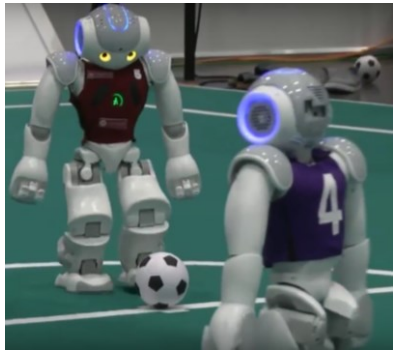
Corso di Visione e Percezione
A.A. 2019/2020

Docente
Domenico Daniele Bloisi

Esercizi Python
Parte 3

Marzo 2020




Il corso

- Home page del corso
  http://web.unibas.it/bloisi/corsi/visione-e-percezione.html

- Docente: Domenico Daniele Bloisi

- Periodo: II semestre marzo 2020 – giugno 2020

  Martedì 17:00-19:00 (Aula GUGLIELMINI)
  Mercoledì 8:30-10:30 (Aula GUGLIELMINI)

Obiettivi del corso

Il corso intende fornire agli studenti conoscenze relative alla programmazione in Python per lo sviluppo di applicazioni basate sul sistema operativo ROS, sulla libreria per la percezione OpenCV e sulla libreria per il Deep Learning Keras

https://www.youtube.com/watch?v=l9KYJlLnEbw

Esercizio 1

Scrivere del codice Python per rappresentare le due matrici seguenti

M1

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

M2

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

Esercizio 1 - Soluzione

```
class Matrix:
  def __init__(self, r, c):
    self.data = []
    self.rows = r
    self.cols = c
    for i in range(r*c):
      self.data.append(0)

  def get_elem(self, i):
    return self.data[i]

  def set_elem(self, i, v):
    self.data[i] = v

  def print(self):
    for i in range(self.rows):
      for j in range(self.cols):
        print(self.get_elem(j + i*self.rows), end = ' ')
      print()
```

Esercizio 1 - Soluzione

```
M1 = Matrix(4,4)
M1.set_elem(0,56)
M1.print()
```

```
56 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```

Esercizio 1 - Soluzione

```
M1.set_elem(1,32)
M1.set_elem(2,10)
M1.set_elem(3,18)
M1.set_elem(4,90)
M1.set_elem(5,23)
M1.set_elem(6,128)
M1.set_elem(7,133)
M1.set_elem(8,24)
M1.set_elem(9,26)
M1.set_elem(10,178)
M1.set_elem(11,200)
M1.set_elem(12,2)
M1.set_elem(13,0)
M1.set_elem(14,255)
M1.set_elem(15,220)

M1.print()
```

```
56 32 10 18
90 23 128 133
24 26 178 200
2 0 255 220
```

M1

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

La creazione e l'inizializzazione di M2 è lasciata per esercizio…

Esercizio 2

Calcolare la distanza L1 (distanza Manhattan) tra M1 e M2

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^P - I_2^P|$$

Esercizio 2 - Soluzione

$$\left| M1 - M2 \right| = \text{add}$$

M1

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

M2

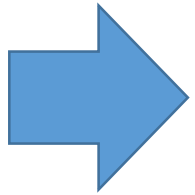
10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

$d_1(M_1, M_2) = 456$

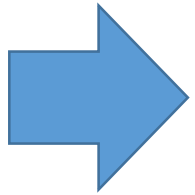
Esercizio 2 - Soluzione

```
class Error(Exception):
  """Base class for exceptions in this module."""
  pass

class DiffDimsError(Error):
  """Raised when the dims of the matrices are different"""
  pass

class Matrix:
  def __init__(self, r, c):
    self.data = []
    self.rows = r
    self.cols = c
    for i in range(r*c):
      self.data.append(0)

  def get_elem(self, i):
    return self.data[i]

  def set_elem(self, i, v):
    self.data[i] = v

  def print(self):
    for i in range(self.rows):
      for j in range(self.cols):
        print(self.get_elem(j + i*self.rows), end = ' ')
      print()
```

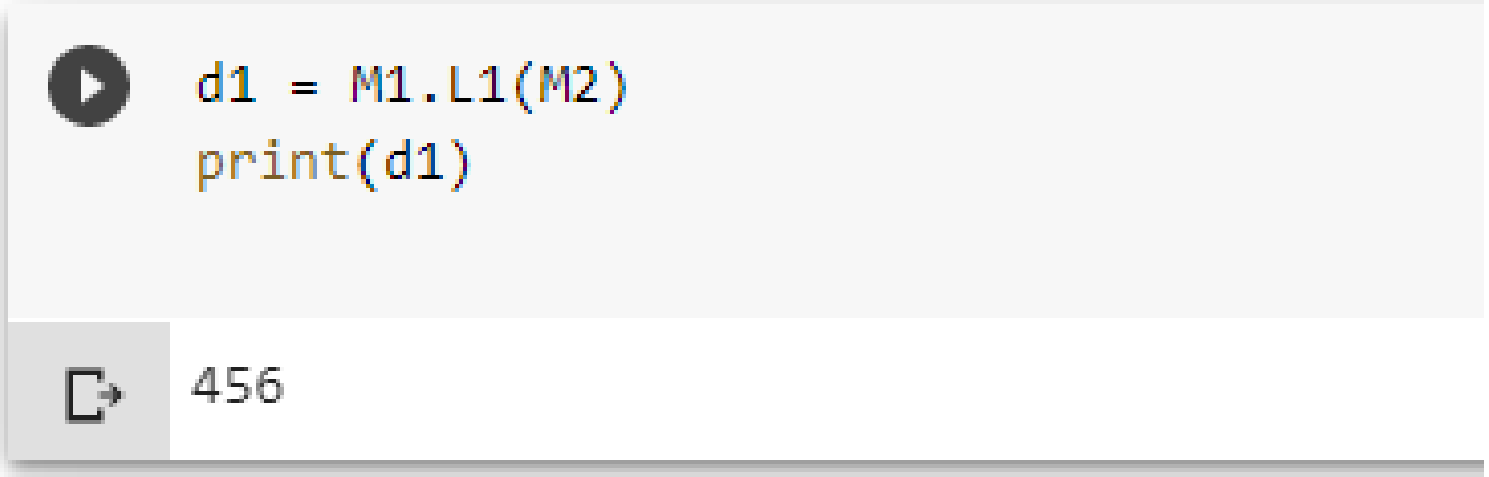


Esercizio 2 - Soluzione

```
def L1(self, mat):
  if mat.rows != self.rows or mat.cols != self.cols:
    raise DiffDimsError
  abs_diff = Matrix(self.rows, self.cols)
  for i in range(self.rows):
    for j in range(self.cols):
      v = abs(self.get_elem(j + i*self.rows) - mat.get_elem(j + i*mat.rows))
      abs_diff.set_elem(j + i*self.rows, v)
  res = 0
  for e in abs_diff.data:
    res = res + e
  return res
```



Esercizio 2 - Soluzione



$d_1(M_1, M_2) = 456$

Esercizio 3

Calcolare la distanza L2 (distanza Euclidea) tra M1 e M2

L2 (Euclidean) distance

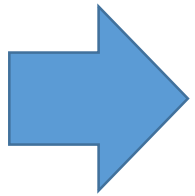
$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^P - I_2^P)^2}$$

Esercizio 3 - soluzione

```
from math import sqrt
```

.
.
.

```
def L2(self, mat):
  if mat.rows != self.rows or mat.cols != self.cols:
    raise DiffDimsError
  pow_diff = Matrix(self.rows, self.cols)
  for i in range(self.rows):
    for j in range(self.cols):
      v = pow(self.get_elem(j + i*self.rows) -
                  mat.get_elem(j + i*mat.rows), 2)
      pow_diff.set_elem(j + i*self.rows, v)
  res = 0
  for e in pow_diff.data:
    res = res + e
  return sqrt(res)
```



Esercizio 3 - soluzione

```
d2 = M1.L2(M2)

print(d2)
```

```
162.11107303327555
```

Esercizio 4

Aggiungere alla classe `Matrix` un metodo per calcolare la matrice trasposta dell'oggetto `Matrix` su cui viene invocato

Esercizio 5

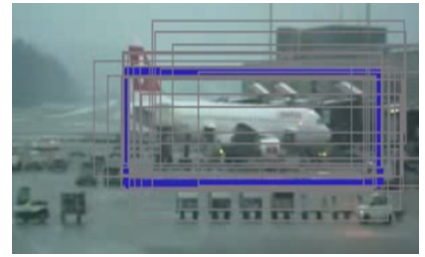
Aggiungere alla classe `Matrix` un metodo per calcolare il rango dell'oggetto `Matrix` su cui viene invocato

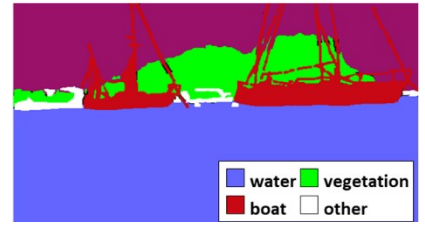
Esercizio 6

Aggiungere alla classe `Matrix` un metodo per calcolare la diagonale principale dell'oggetto `Matrix` su cui viene invocato









UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Corso di Visione e Percezione
A.A. 2019/2020

Docente
Domenico Daniele Bloisi

Esercizi Python
Parte 3

Marzo 2020