

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

## Elektronikos fakultetas

elektroninių SISTEMŲ katedrA

### SKAITMENINIS SIGNALŲ APDOROJIMAS

#### Namų darbas – paveikslų skyros didinimo priemonė

Grupė: ISKF16

Studentas: Edgaras Raudys

Data: 2019-05-20

**Darbo tikslas**

Sukurti programinę įrangą, didinančią pasirinktą paveikslą (nuskaitomą iš \*.bmp tipo failo) naudotojo nurodomą kartų skaičių (nurodomi sveiki skaičiai). Naujai gaunamų vaizdo taškų (angl. pixel) reikšmės apskaičiuojamos vidurkinant gretimų taškų reikšmes. Numatyti galimybę išsaugoti padidintąjį paveikslą naujame \*.bmp tipo faile.

**Reikalavimai programinei įrangai:**

Programinei įrangai reikalingas betkoks kompiuteris palaikantis python programinę kalbą.

**Projektinė dalis:**

Reikalinga failų pasirinkimo sąsaja, pasirinkto paveikslo atvaizdavimas bei vartotojo įvestis kiek kartų padidinti paveikslą. Algoritmas turėtų tikrinti kiekvieną pikselį, iteruoti x bei y ašyse, skaičiuoti aplinkinių pikselių vidurkį bei įterpti į tinkamą koordinate.

**Įgyvendinimas:**

Naudojama klasė Resolution esanti faile Resolution.py. Konstruktoriuje užkraunant nuotrauką, vėliau naudojant nuotrauką change\_resolution paduodant argumentą „how\_many\_times\_to\_enlarge“. Grąžinama padidinta nuotrauka. resolution\_kivy.py faile yra vartotojo sąsajos logikos programinė dalis. Įvykdžius padidinimo funkciją failas automatiškai išsaugomas projekto pradinėje direktorijoje pavadinimu „enlarged\_picture.bmp“ bmp formatu.

class Resolution:

def \_\_init\_\_(self, picture\_location):

self.picture = Image.open(picture\_location)

self.picture\_width, self.picture\_height = self.picture.size

def change\_resolution(self, how\_many\_times\_to\_enlarge):

new\_picture\_width = self.picture\_width \* \

round(how\_many\_times\_to\_enlarge)

new\_picture\_height = self.picture\_height \* \

round(how\_many\_times\_to\_enlarge)

changed\_picture = self.\_\_make\_empty\_picture(

(new\_picture\_width, new\_picture\_height))

surounding\_pixels = [(0, 0) for value in range(9)]

for x in range(new\_picture\_width):

for y in range(new\_picture\_height):

try:

picture\_coordinate\_x = round(x/how\_many\_times\_to\_enlarge)

picture\_coordinate\_y = round(y/how\_many\_times\_to\_enlarge)

surounding\_pixels[0] = self.picture.getpixel(

(picture\_coordinate\_x - 1, picture\_coordinate\_y - 1))

surounding\_pixels[1] = self.picture.getpixel(

(picture\_coordinate\_x - 1, picture\_coordinate\_y))

surounding\_pixels[2] = self.picture.getpixel(

(picture\_coordinate\_x - 1, picture\_coordinate\_y + 1))

surounding\_pixels[3] = self.picture.getpixel(

(picture\_coordinate\_x, picture\_coordinate\_y - 1))

surounding\_pixels[4] = self.picture.getpixel(

(picture\_coordinate\_x, picture\_coordinate\_y))

surounding\_pixels[5] = self.picture.getpixel(

(picture\_coordinate\_x, picture\_coordinate\_y + 1))

surounding\_pixels[6] = self.picture.getpixel(

(picture\_coordinate\_x + 1, picture\_coordinate\_y - 1))

surounding\_pixels[7] = self.picture.getpixel(

(picture\_coordinate\_x + 1, picture\_coordinate\_y))

surounding\_pixels[8] = self.picture.getpixel(

(picture\_coordinate\_x + 1, picture\_coordinate\_y + 1))

median\_pixel = self.\_\_calculate\_median\_pixel(surounding\_pixels)

changed\_picture.putpixel((x, y), median\_pixel)

except:

pass

return changed\_picture

def \_\_calculate\_median\_pixel(self, surounding\_pixels):

red = 0

green = 0

blue = 0

for pixel in surounding\_pixels:

red += pixel[0]

green += pixel[1]

blue += pixel[2]

median\_pixel = [round(color\_value/9)

for color\_value in [red, green, blue]]

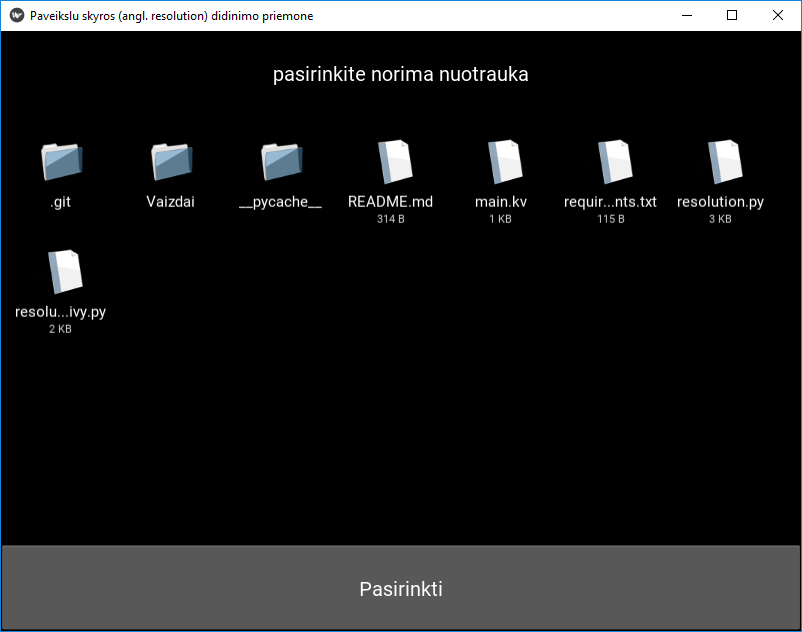
median\_pixel = (median\_pixel[0], median\_pixel[1], median\_pixel[2])

return median\_pixel

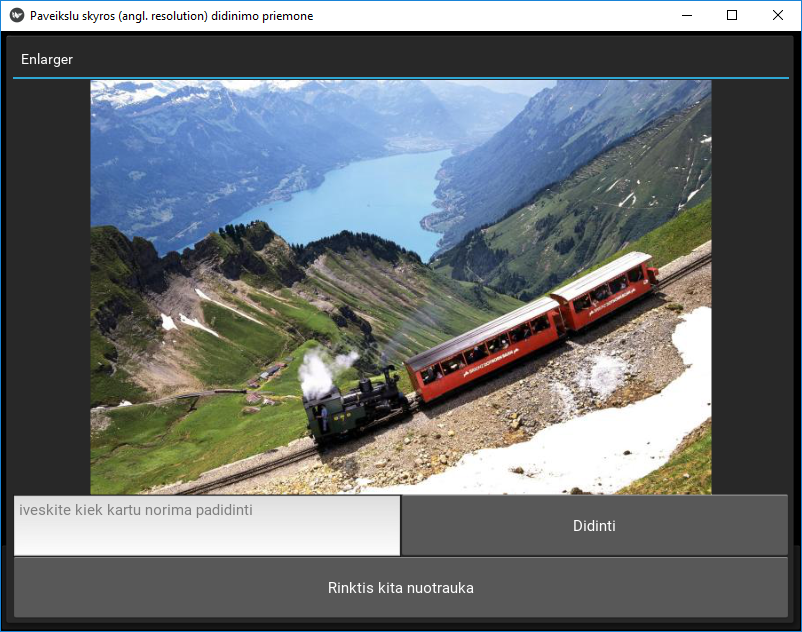
def \_\_make\_empty\_picture(self, size):

return Image.new("RGB", (size), 'white')

**Programos veikimas**



pav. 1 Failo pasirinkimo langas



pav. 2 Paveikslo skyros padidinimo langas

**Darbo rezultatų apibendrinimas:**

Sėkmingai įgyvendintas rezoliucijos padidinimas. Transformacijos laikas lėtas dėl sprendimo, kuris buvo naudotas dėl reikalavimo naudoti žemo lygio programavimą. Naudojant gijas, cython, dataklases bei subprocesus būtų galima pagreitinti įvykdymo laiką dešimtis kartų. Grafinėje sąsaja nepalaikė 8 bei 16 bitų bmp paveikslų, dėl to buvo konvertuota į jpg atvaizdavimui grafinėje sąsajoje, tačiau visi skaičiavimai atlikti tiesiogiai su bmp failu.