**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Руководитель департамента программной инженерии факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_ Р. А. Родригес Залепинос  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

|  |  |
| --- | --- |
| *Подп. и дата* |  |
| *Инв. № дубл.* |  |
| *Взам. инв. №* |  |
| *Подп. и дата* |  |
| *Инв. № подл* | *RU.17701729.04.01-01 12 01-1* |

**Программа для обнаружения облаков на спутниковых снимках**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.01-01 12 01-1-ЛУ**

|  |  |
| --- | --- |
|  | Исполнитель:  студент группы БПИ182  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /М. А. Шакура /  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

**2020**

|  |
| --- |
| УТВЕРЖДЕН  RU.17701729.04.01-01 12 01-1-ЛУ |

|  |  |
| --- | --- |
| *Подп. и дата* |  |
| *Инв. № дубл.* |  |
| *Взам. инв. №* |  |
| *Подп. и дата* |  |
| *Инв. № подл* | *RU.17701729.04.01-01 12 01-1* |

**Программа для обнаружения облаков на спутниковых снимках**

**Текст программы**

**RU.17701729.04.01-01 12 01-1**

**Листов 12**

**2020**

# АННОТАЦИЯ

В данном программном документе представлен текст программы «Программа для обнаружения облаков на спутниковых снимках». В разделе «Текст программы» находится текст программы, распределённый по файлам.

Так же текст программы можно найти на ресурсе: <https://github.com/M2000h/CloudDetect>

**СОДЕРЖАНИЕ**

[АННОТАЦИЯ 2](#_Toc40368206)

[1. ТЕКСТ ПРОГРАММЫ 4](#_Toc40368207)

[1.1. Cloud.py 4](#_Toc40368208)

[1.2. \_\_init\_\_.py 8](#_Toc40368209)

[1.3. return send\_file('/var/www/FlaskApp/FlaskApp/output1.jpg', 'output1.jpg')Servises\IDataStore.cs 9](#_Toc40368210)

# ТЕКСТ ПРОГРАММЫ

## Cloud.py

# Python 3.6

import cv2

from netCDF4 import Dataset

from zipfile import ZipFile

import numpy as np

class Channel:

def \_\_init\_\_(self, name, longname, picture, flags="", wavelength=-1):

"""

Класс хранения одного кадра спутниковой сцены

:param name: Короткое название канала

:param longname: Полное название канала

:param picture: Изображение с канала

:param flags: Флаги, если канал размечен

:param wavelength: Длина волны, если имеется

"""

self.name = name

self.longname = longname

self.picture = picture

self.flags = flags

self.wavelength = wavelength

self.updateMeta()

def updateMeta(self):

"""

Обновляет метаинформацию о канале

min - минимаельное значение пикселя канала

max - максимальное значение пикселя канала

shape - форма изображенния канала

:return:

"""

self.min = np.min(self.picture)

self.max = np.max(self.picture)

self.shape = self.picture.shape

def normalise(self):

"""

Сдвигает спектр изображения канала до интервала [0; 255]

:return:

"""

self.picture -= self.min

self.picture = self.picture \* 255 / np.max(self.picture)

self.updateMeta()

def save(self, folder, printMeta=True):

"""

Сохраняет изображение с канала как картинку

:param folder: Путь до папки сохранения

:param printMeta: Нужно ли печатать информацию об успешности сохранения

:return:

"""

try:

cv2.imwrite(folder + "/" + self.longname + ".jpg", self.picture)

if printMeta: print(self.longname + " saved in " + folder)

except:

if printMeta: print("Can't save " + self.name, file=sys.stderr)

def printMeta(self):

"""

печатает метаинформацию о канале

:return:

"""

print(self.name, self.longname, '\n',

'min:', self.min,

'max:', self.max,

'size: ', self.shape, )

class Scene:

def \_\_init\_\_(self, filename, onlyPic=False, onlyFull=False):

"""

Класс спутниковой сцены, которая хранит информацию об изображениях в отдельных каналах

:param filename: название файла, хранящего спутниковую сцену

:param onlyPic: True, если нужно обрабатывать только каналы с картинкой

:param onlyFull: True, если нужно только каналы с полным изображением

"""

self.channels = {}

nc = Dataset(filename, "r")

for channelName in nc.variables:

print(nc.variables[channelName].long\_name)

flags = ''

wavelength = -1

try:

flags = nc.variables[channelName].flag\_meanings

except Exception as e:

print('Empty flags')

try:

wavelength = nc.variables[channelName].wavelength

except Exception as e:

print('Empty wavelength')

channelPicture = np.array(nc.variables[channelName])

channel = Channel(name=nc.variables[channelName].name,

longname=nc.variables[channelName].long\_name,

picture=channelPicture, wavelength=wavelength,

flags=flags)

if (len(channel.shape) > 1 or not onlyPic) and \

(channel.shape[0] - channel.shape[1]) \* 10 < channel.shape[0] or not onlyFull:

self.channels[channelName] = channel

try:

with ZipFile('/var/www/FlaskApp/FlaskApp/1.zip', 'r') as zip:

flags\_input = zip.open(zip.filelist[0].filename+ 'flags.nc')

spectrum\_input = zip.open(zip.filelist[0].filename+ 'Syn\_Oa10\_reflectance.nc')

flags\_output = open('/var/www/FlaskApp/FlaskApp/flag.nc', 'wb')

spectrum\_output = open('/var/www/FlaskApp/FlaskApp/Syn\_Oa10\_reflectance.nc', 'wb')

flags\_output.write(flags\_input.read())

spectrum\_input.write(spectrum\_output.read())

flags\_input.close()

flags\_output.close()

spectrum\_input.close()

spectrum\_output.close()

scene1 = Scene('/var/www/FlaskApp/FlaskApp/flag.nc', onlyPic=True, onlyFull=True)

scene2 = Scene('/var/www/FlaskApp/FlaskApp/Syn\_Oa10\_reflectance.nc', onlyPic=True, onlyFull=True)

general\_mask = scene1.channels['OLC\_flags'].picture

cloud\_mask = scene1.channels['CLOUD\_flags'].picture

cloud\_land\_mask = scene2.channels['SDR\_Oa10\_err'].picture

landpik = np.full((general\_mask.shape + (3,)), [255, 187, 153])

landmask = np.zeros(general\_mask.shape)

landpik[general\_mask // 4096 % 2 == 1] = [0, 200, 0] # OLC\_land

landpik[general\_mask // 1024 % 2 == 1] = [100, 0, 0] # OLC\_fresh\_inland\_water

landmask[general\_mask // 4096 % 2 == 1] = 1

landmask[general\_mask // 1024 % 2 == 1] = 0

cloud\_mask[landmask == 1] = 0

landpik[cloud\_mask % 2 == 1] = [250, 250, 250]

cloud\_land\_mask[landmask == 0] = 1000

landpik[cloud\_land\_mask < 100] = [250, 250, 250]

cv2.imwrite("/var/www/FlaskApp/FlaskApp/output1.jpg", landpik)

print("ok")

except Exception as e:

print(e)

## \_\_init\_\_.py

from flask import Flask, send\_file, request, json, redirect, render\_template

import urllib.request

import numpy as np

sys.path.append("/var/www/FlaskApp/FlaskApp")

import os

import warnings

from subprocess import check\_output

app = Flask(\_\_name\_\_)

@app.route('/CloudDetect')

def clouddetect():

return send\_file('/var/www/FlaskApp/FlaskApp/static/CloudDetect.html', CloudDetect.html")

@app.route('/uploadLabel',methods=['POST'])

def uploadLabel():

if 'myfile.txt' not in request.files:

return "Error"

file = request.files['myfile.txt']

file.save('/var/www/FlaskApp/FlaskApp/1.zip')

a = check\_output("python3.6 /var/www/FlaskApp/FlaskApp/Cloud.py", shell=True)

app.logger.error(a)

return send\_file('/var/www/FlaskApp/FlaskApp/output1.jpg', 'output1.jpg')Servises\IDataStore.cs

## CloudDetect.html

<html>

<head>

<script>

function uploadFile(file) {

var url = '/uploadLabel';

var xhr = new XMLHttpRequest();

var formData = new FormData();

xhr.open('POST', url, true);

xhr.addEventListener('readystatechange', function(e) {

if (xhr.readyState == 4 && xhr.status == 200) {

// Done. Inform the user

}

else if (xhr.readyState == 4 && xhr.status != 200) {

// Error. Inform the user

}

});

formData.append('file', file);

xhr.send(formData);

}

</script>

<style>

.c {

position: relative;

margin:0;

}

.c img {position: absolute;}

.c div{

position: absolute;

background: whitesmoke;

border-radius: 10px;

text-align: center;

align-content: center;

top: 50%;

left: 50%;

margin-top: -50px;

margin-left: -150px;

}

.c form{

margin: 10px;

}

.c a{

margin: 10px;

}

</style>

</head>

<body class="c">

<img src="/pic/CloudBack.jpg" width="100%" draggable="false">

<div class="box">

<form id="uploadbanner" enctype="multipart/form-data" method="post" action="/uploadLabel">

<input id="fileupload" name="myfile.txt" type="file" />

<input type="submit" value="submit" id="submit" />

</form>

<a target="\_blank" href="https://scihub.copernicus.eu/dhus/#/home">

You can take exemples here.

</a>

</div>

</body>

</html>

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |