Backlog部分

上传视频文件

使用OpenCV读取视频文件

将视频拆分为帧

使用帧差异技术进行背景减法

计算前景掩码和前景百分比

使用概率背景减法技术进行背景减法

保存图片

导出为pdf文件

SRS部分

1. Introduction

1.1 Purpose

Video Slideshow Extraction and PDF Converter" is designed to provide users with an efficient and convenient way to extract slideshow images from video files and convert them to PDF format. As a widely used electronic document format, PDF has good readability and portability, allowing users to easily view and share the extracted slides on different devices.

1.2 Range

The software is designed to provide an efficient and convenient tool that enables users to easily extract slideshow pictures from video files and convert them to PDF format. By automating the identification and conversion process, users can save a lot of manual processing time and effort. At the same time, the software also focuses on user experience and data security to ensure that users enjoy smooth and stable services during operation.

1.3 References

User demand research

Market research report

Relevant laws and regulations

2. General description

2.1 Product Perspective

The software will function as a mobile application, allowing users to extract slideshow images from video files and export them as PDF files.

2.2 Product Functions

Video file reading

Slide image extraction

PDF export

Batch processing

Error handling

3. Specific requirements

3.1 Video file reading

3.1.1 The software should support reading multiple common video file formats, including but not limited to MP4, AVI, WMV, MOV, etc.

3.2 Slide image extraction

3.2.1 The software should be able to automatically recognize the slide image in the video and extract it.

3.2.2 The extracted images should be legible and maintain the original resolution and quality.

3.3 PDF export

3.3.1 The extracted slideshow pictures should be able to be automatically organized and exported as a PDF file in the order in which they appear in the video.

3.3.2 The exported PDF file should maintain good typography and clarity.

3.4 Batch processing

3.4.1 The software should support batch processing of multiple video files to improve work efficiency.

3.5 Error handling

3.5.1 In the process of reading videos, extracting slides and exporting PDFS, the software should be able to properly handle various abnormal situations, such as video file corruption, failure to recognize slides, and give clear error prompts.

3.6 Data security and stability

3.6.1 The software should ensure the security of user data and prevent data leakage or malicious tampering.

3.6.2 The software should have good stability to avoid data loss or damage caused by program errors.

3.7 Processing speed and resource usage

3.7.1 The software should be able to complete the video file reading, slide picture extraction and PDF file export in a reasonable time. The exact time depends on the size of the video file and the number of slide pictures, but should ensure smooth user operation.

3.7.2 Resource usage: The software should reasonably occupy system resources during operation to avoid system stalling or crash.

3.8 User interface requirements

3.8.1 The user interface should be simple and clear, easy to understand and operate.

3.8.2 The software should support multiple languages to meet the needs of users in different regions.

3.8.3 The software should provide detailed help documentation so that users can quickly find solutions when they encounter problems during use.

@startuml

skinparam sequenceMessageAlign right

Teacher -> System : PressBtnMakePhoto

activate System

System ->Recogniser: frame = MakePhoto()

System ->Recogniser: teble = Recognise(frame)

activate Recogniser

Recogniser -> Tabler: "create"

deactivate Recogniser

deactivate System

@enduml

@startuml

class Syestem {

PressBtnMakePhoto ()

}

class Recogniser {

MakePhoto()

Recognise(frame)

}

Syestem \*- Recogniser : has

Recogniser --> Table : > depend on

@enduml

class Table:

def init(self):

pass

class Recogniser:

def init(self):

pass

def MakePhoto(self):

pass

def Recognise(self,frame):

t = Table

return t

if name == "main":

r = Recogniser

r.MakePhoto

import unittest

class Frame:

def init(self):

pass

class Table:

def init(self):

pass

def getRawNum(self):

return 0

def getColNum(self):

return 0

class Recogniser:

def init(self):

pass

def MakePhoto(self):

pass

def Recognise(self,frame):

t = Table ()

return t

class TestRecogniser(unittest.TestCase):

def setUp(self):

pass

def test\_ckech\_empty\_frame\_recognition(self):

r = Recogniser ()

f = Frame () # Empty frame

t = r.Recognise(frame = f)

self.assertEqual(t.getColNum(), 0)

self.assertEqual(t.getRawNum(), 0)

if name == "main":

# r = Recogniser

# r.MakePhoto

unittest.main()