

情報科学演習_第4期課題
マルチメディアプログラミング

6321120
横溝 尚也

提出日：8月7日（火）

担当者名：澤田先生

第 1 章

課題 1：動画の読み込み

1.1 課題内容

動画（People - 84973.mp4）を読み込み、動画の縦・横のサイズ、FPS、総フレーム数を出力してください。

1.2 アルゴリズムの説明

動画を読み込む際は、パスを指定して動画の対象を探索する。動画のサイズ、FPS、縦フレームの出力はその言語ごとに出力する構文があるため、それを使用する。

1.3 プログラムの説明

ソースコードに説明の都合上、行数をふったものをここに記す。

また、インポート部分は初めに記述し、以降は省略する。

プログラム 1.1 :6321120_final_exam

```
1 import cv2
2 from matplotlib import pyplot as plt
3 import numpy as np
4 #1
5
6 path = "/content/drive/MyDrive/Colab Notebooks/dataset"
7 video1 = cv2.VideoCapture(path+"/videos/People - 84973.mp4")
8
9 print("Width:", video1.get(cv2.CAP_PROP_FRAME_WIDTH))
10 print("Height:", video1.get(cv2.CAP_PROP_FRAME_HEIGHT))
11
12 fps = video1.get(cv2.CAP_PROP_FPS)
13 print("FPS:", fps)
14
15 frame_num = video1.get(cv2.CAP_PROP_FRAME_COUNT)
16 print("All Frame:", frame_num)
```

1～4 行目

OpenCV や Numpy、Matplotlib をインポートしている。

6～16 行目

- 6 行目 path に今回の課題で使用する動画や画像を格納する共通のディレクトリ（dataset ディレクトリまで）を格納する。
- 7 行目 動画を video1 に格納
- 9 行目 動画の幅を出力
- 10 行目 動画の高さを出力
- 12 行目 動画の fps を出力
- 16 行目 総フレーム数を出力

1.4 実行結果



The screenshot shows a Jupyter Notebook interface with a dark theme. The title bar at the top reads '課題1 動画の読み込み'. The code cell contains the following Python code:

```
import cv2
from matplotlib import pyplot as plt
import numpy as np
#1

path = "/content/drive/MyDrive/Colab Notebooks/dataset"
video1 = cv2.VideoCapture(path+"/videos/People - 84973.mp4")

print("Width:", video1.get(cv2.CAP_PROP_FRAME_WIDTH))
print("Height:", video1.get(cv2.CAP_PROP_FRAME_HEIGHT))

fps = video1.get(cv2.CAP_PROP_FPS)
print("FPS:", fps)

frame_num = video1.get(cv2.CAP_PROP_FRAME_COUNT)
print("All Frame:", frame_num)
```

The output cell below the code shows the results of the execution:

```
Width: 1280.0
Height: 720.0
FPS: 25.0
All Frame: 241.0
```

図 1.1 実行結果 1

第 2 章

課題 2：サムネイル画像の作成

2.1 課題内容

サムネイルとして動画（People - 84973.mp4）中のフレーム画像を 1 枚出力し、保存してください。

2.2 アルゴリズムの説明

動画は微小時間間隔で画像を集め、つなぎ合わせたものであり、そのうちの 1 枚を読み取り、出力する。特定のフレーム画像を条件文を使用することで出力することもできる。

2.3 プログラムの説明

プログラム 2.1 :6321120_final_exam

```
1 ret, frame = video1.read()
2 show_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
3
4
5 while True:
6     ret, show_frameframe = video1.read()
7     if not ret:
8         break
9     if int(video1.get(cv2.CAP_PROP_POS_FRAMES)) == 40:
10        plt.imshow(show_frame)
11        cv2.imwrite(path + "/videos/output/632120_People_thumb.jpg", frame)
12        break
13 video1.release()
```

1～2 行目動画を読み込み、BGR から RGB に変換

5 13 行目

- 5 行目 次のフレーム画像があれば繰り返し実行
- 9 行目 フレーム数が 40 の時のフレーム画像を出力
- 11 行目 パスを指定し、出力したフレーム画像を保存

- 13行目 メモリを強制的に開放

2.4 実行結果

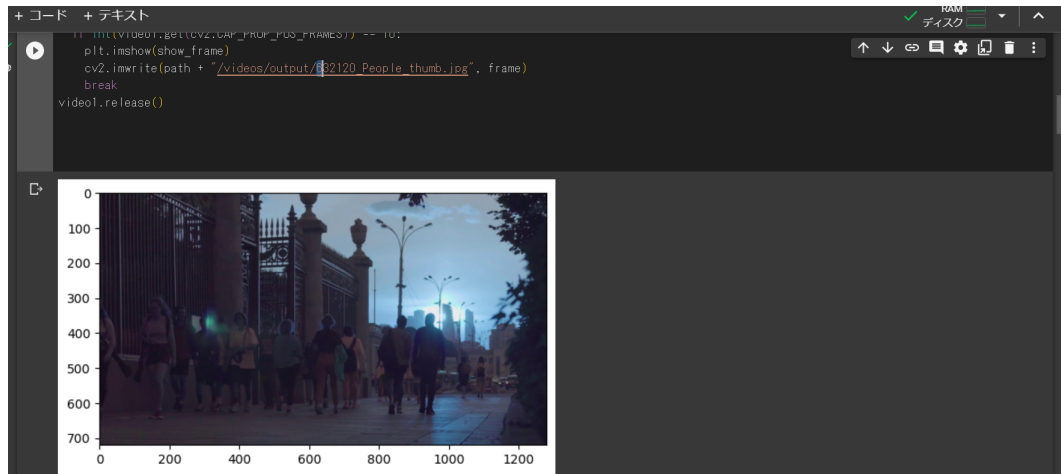


図 2.1 実行結果 2

第 3 章

課題 3：ファイル名のパース

3.1 課題内容

動画のファイル名 (People - 84973.mp4) の文字列を処理して、タイトル (People) と ID (84973) に分けるプログラムを作成し、その結果を出力してください。

3.2 アルゴリズムの説明

ファイル名を受け取り、空白を基準に分割し、各分割後要素を配列に格納する。配列 0 番目要素にタイトル、最終要素が ID が格納されている。ここで、ファイル名は～ (ID) .mv4 と後ろに拡張子が付与されているため、特定の単語を除いた結果を出力する。

3.3 プログラムの説明

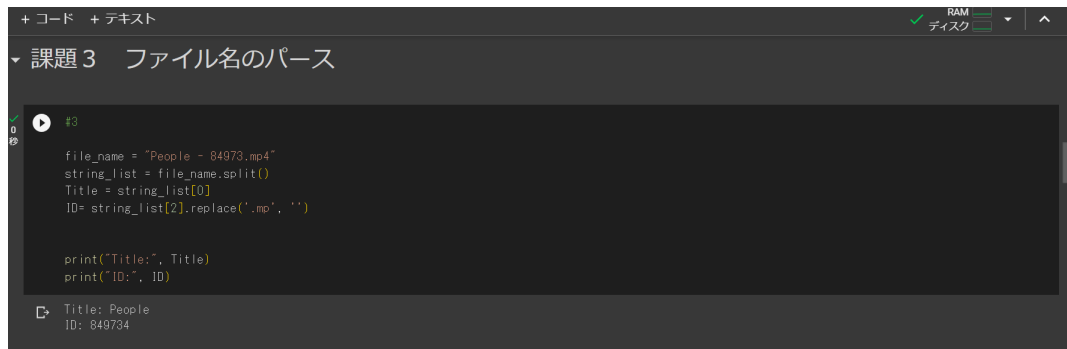
プログラム 3.1 :6321120_final_exam

```
1 file_name = "People - 84973.mp4"
2 string_list = file_name.split()
3 Title = string_list[0]
4 ID= string_list[2].replace('.mp', '')
5
6
7 print("Title:", Title)
8 print("ID:", ID)
```

1～8 行目

- 1 行目 ファイル名を file_name に格納
- 2 行目 ファイル名を空白区切りで分割
- 3 行目 Title にタイトルを格納
- 4 行目 ID を主力するため、拡張子を除去
- 7～8 行目 タイトルと ID を出力

3.4 実行結果



The screenshot shows a code editor window with a dark theme. The title bar at the top indicates '+ コード + テキスト' and shows status for 'RAM' and 'ディスク' (Disk) with green checkmarks. The main area is titled '課題3 ファイル名のパース' (Task 3: Parsing File Name). Below the title, there is a play button icon and the text '#3'. The code is as follows:

```
file_name = "People - 84973.mp4"
string_list = file_name.split()
Title = string_list[0]
ID = string_list[2].replace('.mp', '')

print("Title:", Title)
print("ID:", ID)
```

At the bottom of the editor, the output is displayed:

```
Title: People
ID: 849734
```

図 3.1 実行結果 3

第 4 章

課題 4：二値化と動画の保存

4.1 課題内容

動画 (Cat - 66004.mp4) を 2 値化した動画を作成するプログラムを作成し、その動画を保存してください。閾値は自分で設定してかまいません。

二値化：画素値が閾値より大きければ白を割り当て、そうでなければ黒を割り当てる処理

4.2 アルゴリズムの説明

画像は RGB で描かれているものをある基準を使用して白黒にしていく。この際、画素値が黒 0～白 255 で表されるため、半分の 128 を基準として 128 未満ならば白、128 以上なら黒を出力する。

4.3 プログラムの説明

プログラム 4.1 :6321120_final_exam

```
1 video2 = cv2.VideoCapture(path + "/videos/Cat - 66004.mp4")
2 size = (int(video2.get(cv2.CAP_PROP_FRAME_WIDTH)), int(video2.get(cv2.
    CAP_PROP_FRAME_HEIGHT)))
3 new_size = (int(size[0]*0.3), int(size[1]*0.3))
4 new_video = cv2.VideoWriter(path + "/videos/output/6321120_cat_binary.mp4", cv2.
    VideoWriter_fourcc('M', 'P', '4', 'V'), int(video2.get(cv2.CAP_PROP_FPS)), new_size)
5
6 while True:
7     ret, frame = video2.read()
8     if not ret:
9         break
10
11     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
12     ret, gray2 = cv2.threshold(gray,128,255,cv2.THRESH_BINARY)
13     resize = cv2.resize(gray2, new_size)
14     bgr = cv2.cvtColor(resize,cv2.COLOR_GRAY2BGR)
15     new_video.write(bgr)
```



```
16
17  video2.release()
18  new_video.release()
```

1～4 行目

- 1 行目 対象動画を取得
- 2～4 行目 動画サイズを取得し、サイズを 0.3 倍とし、新たな動画 new_video とする
- 6～15 行目 BGR をグレースケール化し、その画素値で白黒で出力するために、threshold を使用する。threshold とは第 1 引数に画像情報、第 2 引数に閾値、第 3 引数に閾値を超えた際の色の指定、第 4 引数に 2 値化するための条件のタイプを指定
- 16～17 行目 メモリを強制的に開放

4.4 実行結果



図 4.1 実行結果 4

第 5 章

課題 5：音声の可視化

5.1 課題内容

音声 (report_audio.wav) には時間-周波数領域にメッセージが隠されています。パラメータを適宜調節して解析し、可視化したグラフを描画してください。

5.2 アルゴリズムの説明

読み込んだ音声のスペクトラムを描画する。横軸を時間、縦軸を周波数帯とし、グラフを出力する。

5.3 プログラムの説明

プログラム 5.1 :6321120_final_exam

```
1 import librosa
2 import IPython.display
3 import soundfile as sf
4
5 y, sr = librosa.load(path + "/audio/report_audio.wav")
6
7
8
9 D = librosa.amplitude_to_db(np.abs(librosa.stft(y)),ref = np.max)
10 librosa.display.specshow(D, y_axis='log', sr=sr,x_axis='time')
11 plt.colorbar()
```

1～11 行目

- 1～3 行目 必要なものを librosa をインポート
- 5 行目 音声の読みこみ
- 9～11 行目 適切なパラメータを記述してグラフを描画

5.4 実行結果

実行結果から、GOODWork が音声の中に情報として含まれていることが分かった。

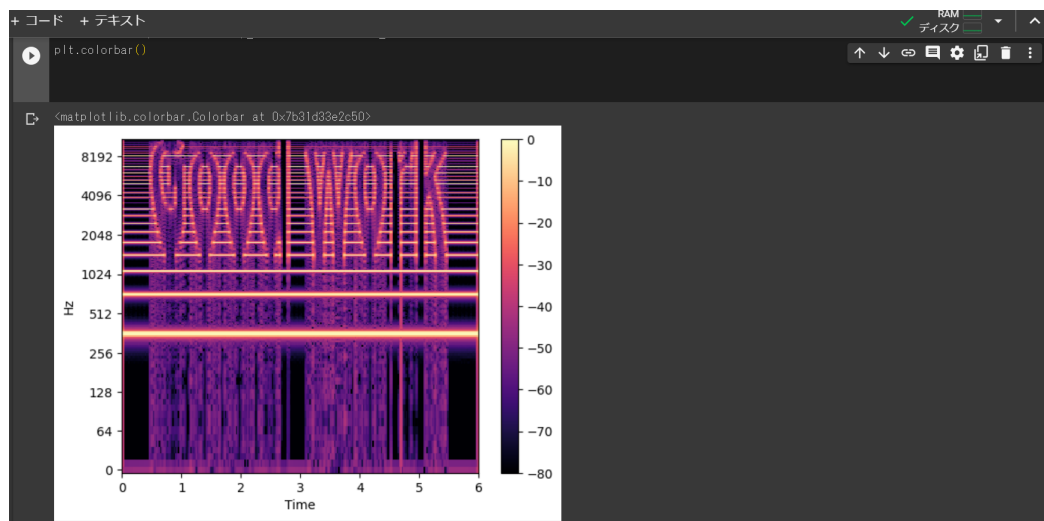


図 5.1 実行結果 5

第 6 章

課題 6：動画の変わり目を検出する

6.1 課題内容

動画 (Detection.mp4) は 4 つの動画を結合したものです。

この動画から動画の変わり目 (3 つ) を以下の方法で自動で検出してください。

6.2 アルゴリズムの説明

動画の変わり目の判別方法は、フレームごとの画素値の差が大きいものを検出する。動画の全フレームに対する差をデータとして集め、基準値を上回るものを検出する流れである。基準値の設定に関しては全データの変化量の平均をある定数倍したものとする。基準値の定め方については考察で詳しく記述する。

6.3 プログラムの説明

プログラム 6.1 :6321120_final_exam

```
1 video3 = cv2.VideoCapture(path + "/videos/Detection.mp4")
2
3 ret, frame1 = video3.read()
4 frame1_g = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
5 change_amounts = []
6
7 while True:
8     ret, frame2 = video3.read()
9     if not ret:
10         break
11
12     frame2_g = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)
13
14     difference = np.abs(frame1_g.astype(np.int16) - frame2_g.astype(np.int16))
15
16     change_amount = np.sum(difference)
17     change_amounts.append(change_amount)
```

```

18
19     frame1_g = frame2_g
20
21     video3.release()
22
23     plt.plot(change_amounts)
24     plt.xlabel('frame num')
25     plt.ylabel("change amounts")
26     plt.show()
27
28     threshold = np.mean(change_amounts) * 10
29
30     change_frames = [i for i, v in enumerate(change_amounts) if v > threshold]
31     print('Change frames:', change_frames)

```

1～5 行目

- 1 行目 パスを指定して動画の読み込み
- 3 行目 動画の 1 フレームの読み込み
- 4 行目 フレームのグレースケール化
- 5 行目 フレームの画素値の変化量を格納する変数を用意

7～21 行目

- 7～12 行目 次のフレームが存在することを条件にループを行う。次のフレームを frame_2 とし、グレースケール化して
- 14 行目 frame_1 と frame_2 の差の絶対値を change_amounts に格納する。その際、NumPy の abs 関数を使用して絶対値を取っている。
- 19 行目 最後に次のループのために、frame_2 を更新
- 21 行目 強制的にメモリを開放

23～31 行目

- 23～26 行目 グラフを描画
- 28～30 行目 閾値をフレーム差の平均値の 10 倍に設定し、閾値を超える変化量があったものを change_frames に格納。
- 31 行目 平均値は NumPy の mean 関数を使用

6.4 実行結果

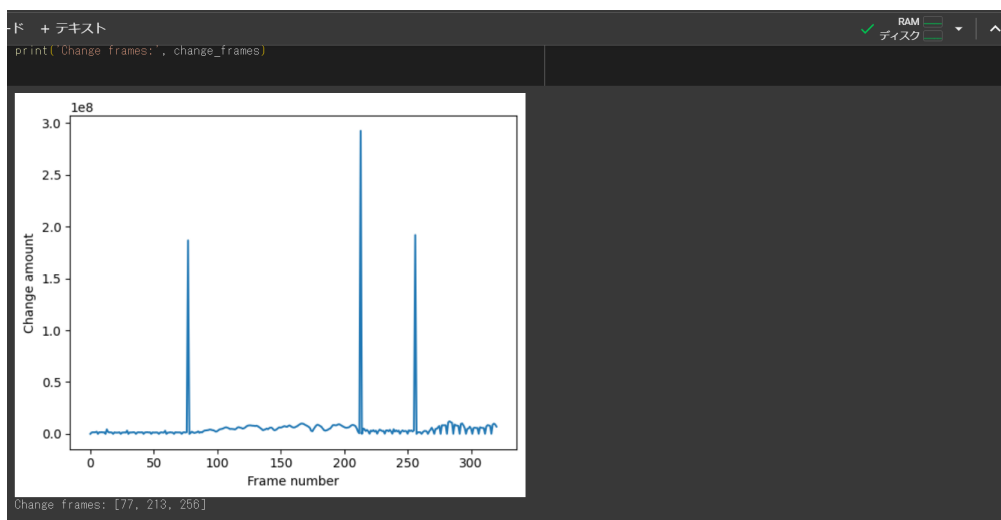


図 6.1 実行結果 6

第 7 章

課題 7：オブジェクト指向：Video クラスの作成

7.1 課題内容

Video クラスは 動画の path を受け取り，インスタンス化する．

Video クラスだけではなく，インスタンス化して動作を確認するプログラムもあわせて作成すること．

インスタンス変数として以下の 4 つを保持する：

- 動画の path
- タイトル
- ID
- サムネイル

コンストラクタ：

- 各変数を初期化

クラスメソッドとして以下の 5 つを用意する：

- サムネイルを作成するメソッドを作成
 - 課題 2 をクラスメソッドにする
 - サムネイルはインスタンス変数に代入
- ファイル名から特定の文字列（タイトルと ID）を取り出すメソッドの作成
 - 課題 3 をクラスメソッドにする
 - タイトルと ID はインスタンス変数に代入
- 4 つのインスタンス変数の値を取得する 4 つのメソッド（ゲッター）
 - get_title()

- get_id()
- get_thumbnail()
- get_path()
- 2値化する動画を作成するメソッドの作成
 - 課題4をクラスメソッドにする
- 2値化する動画を作成するメソッドをフレームを処理する関数と動画処理を行う関数に分離し、引数としてフレームを処理する関数を受け取るように変更してください
 - cf. [高階関数] (<https://www.notion.so/1-Python-fa98e2f92db543c1b0a2f325d2abd760?pvs=21>)

(画像処理をする関数を増やしたとしても、その関数を渡すだけで動画処理ができるようにする)

- 引数として受け取ったフレームに対して二値化する関数を作成
- 引数として受け取ったフレームに対してグレースケール化する関数を作成
- 上の2つの関数のどちらかを引数として受け取り、その関数が適用された動画を生成する関数を作成する

より良いと思う設計があれば適宜変更しても良いです（評価に含む）

その際は考察に書いてください

7.2 アルゴリズムの説明

7.3 プログラムの説明

プログラム 7.1 :6321120_final_exam

```

1 class Video:
2     def __init__(self, video_path):
3         self.path = video_path
4         self.title = None
5         self.id = None
6         self.thumbnail = None
7         self.video4 = cv2.VideoCapture(self.path)
8         self.frame_size = (int(self.video4.get(cv2.CAP_PROP_FRAME_WIDTH)), int(self.video4.
           get(cv2.CAP_PROP_FRAME_HEIGHT)))
9         self.new_size = (int(self.frame_size[0]*0.3) , int(self.frame_size[1]*0.3))
10
11     def mkthumbnail(self):
12         retval, frame = self.video4.read()
13         self.thumbnail = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
14         plt.imshow(self.thumbnail)
15         cv2.imwrite(path + "/videos/output/" + self.id + "_thumb.jpg" , frame)

```



```

16
17 def split_name(self):
18     video_name = self.path.split('/')[-1]
19     parts = video_name.split(' - ')
20     self.title = parts[0]
21     rmid = parts[1]
22     self.id = rmid.split('.')[0]
23
24 def get_title(self):
25     self.split_name()
26     return self.title
27
28 def get_id(self):
29     return self.id
30
31 def get_thumbnail(self):
32     return self.thumbnail
33
34 def get_path(self):
35     return self.path
36
37 def binary_process(self, frame):
38     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
39     ret, binary = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY)
40     resize = cv2.resize(binary, self.new_size)
41     bgr = cv2.cvtColor(resize, cv2.COLOR_GRAY2BGR)
42     return bgr
43
44 def gray_process(self, frame):
45     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
46     resize = cv2.resize(gray, self.new_size)
47     bgr = cv2.cvtColor(resize, cv2.COLOR_GRAY2BGR)
48     return bgr
49
50 def mkprocess_video(self, process_fun):
51     w_video = cv2.VideoWriter(path + "/videos/output/" + self.title + "-" + self.id +
52                               "_processed.mp4", cv2.VideoWriter_fourcc('M', 'P', '4', 'V'), int(self.video4.get(
53                                   cv2.CAP_PROP_FPS)), self.new_size)
54
55 while True:
56     ret, frame = self.video4.read()
57     if not ret:
58         break
59     process_frame = process_fun(frame)
60     w_video.write(process_frame)

```

```
60     self.video4.release()
61     w_video.release()
62
63
64 video = Video(path + "/videos/People - 84973.mp4")
65
66 video.split_name()
67 print("Title:" , video.get_title())
68 print("ID:" , video.get_id())
69
70 video.mkthumbnail()
71
72 print("path:" , video.get_path())
73
74 video.mkprocess_video(video.gray_process)
75 video.mkprocess_video(video.binary_process)
```

1～22 行目

- 2 行目 `__init__`関数では受け取ったパラメータから ID、タイトル、サムネイルなどの初期化を行う。
- 8～9 行目 読み込んだフレームのサイズを読み取り、そのサイズを 0.3 倍したものを新たなサイズとしている。
- 11 行目 サムネイルの作成を行う関数である。フレームを読み込み、RGB 化、サムネイルの保存するパスを指定している。
- 17 行目 ID から必要な情報であるタイトルを抽出している。その際、`split` 関数を使用している。

24～35 行目

インスタンス変数の値を取得する 4 つのメソッドを問題指示通り記述

37～51 行目

- 37 行目 2 値化を行う関数。課題 4 と類似しているため省略
- 44 行目 グレースケール化を行う関数。課題 4 と類似しているため省略
- 50 行目 引数として受け取った処理関数と対象動画によって処理を行い、新たな動画として指定されたパスに保存

53～75 行目

- 64 行目 `Video` クラスのインスタンスを作成して、課題 1 の動画のタイトル、ID、サムネイルを出力している。

7.4 実行結果

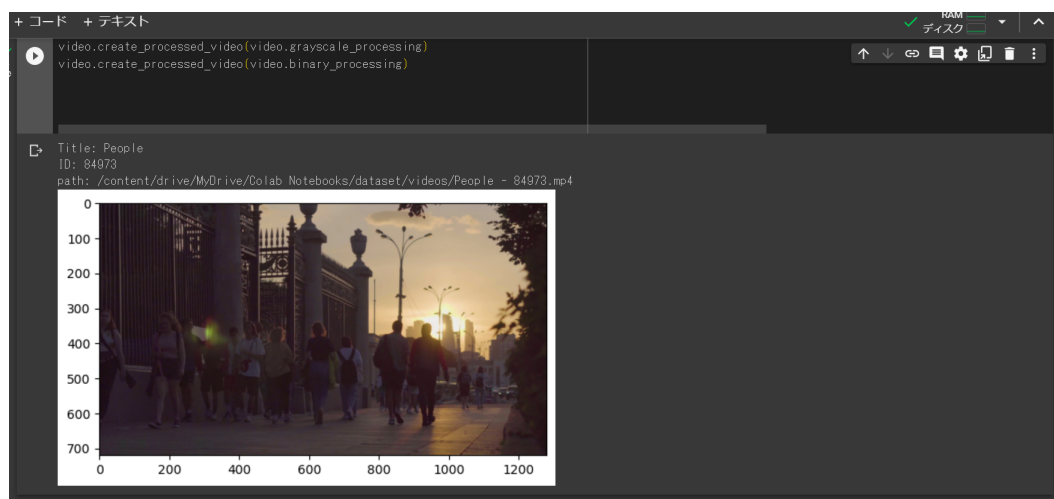


図 7.1 実行結果 7

第 8 章

考察

8.1 フレーム数について

課題 1 で出力したフレーム数や FPS は動画がどれほど滑らかに動くかを表すパラメータである。動画は静止画を多くの数つなぎ合わせたものであり、総フレーム数が大きければ大きいほど人間が見て、より滑らかに見え、小さければ、パラパラ漫画のように画像から画像の遷移が荒くなる。

8.2 サムネイルの作成について

課題 2 では指定された動画のサムネイルを作成した。あるフレームを読み込むだけならば、動画の一番先頭フレームをサムネイルと保存すればよい。しかし、ある特定の時間の画像をサムネイルにしたければ (k 番目のフレーム) = (総フレーム数) * (サムネイルにしたい画像の秒数) / (動画の総時間) によって求めることができ出力画像を変更することもできる。

8.3 動画の変わり目の検出方法

課題 6 では動画の変わり目を検出するプログラムを作成したが、図 6.1 のグラフの描画より、明らかにフレーム数が 80, 210, 250 付近変化量とびぬけて高いため、変わり目であることはわかる。しかしこの辺別をプログラムに正確に行わせるために、閾値を適正な値にする必要がある。今回、自分は変化量の平均値を 10 倍した値を閾値として判別を使用した。この値が低すぎれば判定が甘くなり、余分なフレームも出力されてしまう。逆に、高すぎれば判定がきつすぎて出力されてほしいものもされなくなってしまう。この閾値の設定をより正確に行うことで、今回の課題だけでなく様々なプログラムで正確な判定ができると考えられる。その為には閾値の設定をほかのより良いアルゴリズムで行ったり、平均値をいくら定数倍すれば最良の閾値になるか考える必要がある。