情報システム工学演習 II 第三部 デジタル画像処理 環境導入編

大倉 史生

0 はじめに

情報システム工学演習 II では、オブジェクト指向、デジタル信号/画像処理を中心に、実用・研究に近い内容の演習を行う。第三部「デジタル画像処理」では、Python を使った画像処理を中心に演習を進める。提出物は最終レポートのみである。自由な発想で好きなものを作って欲しい。

資料やコードなどのバグ報告は大倉 okura@ist.osaka-u.ac.jp まで。

0.1 関連する講義

「デジタル画像処理」

本演習は当該科目の履修を前提とはしないが、演習に深く関わる内容が紹介されているので、興味のある学生は是非受講してほしい。画像処理はかなり一般的なトピックであるので、ネット上にもリソースは多い。特に、今回使用する OpenCV のようなメジャーなライブラリは、無限にリソースがある。演習を進めるにあたって不明な点や、さらに深掘りしたい点があれば、積極的に調べてみると良い。

0.2 本演習の目的

画像処理は、既存ライブラリが充実しており、参入障壁が低い分野の一つである。卒業研究や就職後の人生に向けて、型にはまった演習を抜け出し、クリエイティビティを発揮する練習をしよう。

0.3 演習環境

Python + OpenCV の環境で演習を進める。インストール等の詳細は次章で説明する。OpenCV は、Open Computer Vision Library の略であり、画像処理関連の標準的なライブラリである。

Python、OpenCV ともに習得が容易なので、興味のある学生はいろいろ遊んでみると良い。楽しいと思う。

0.4 参考図書

1. ディジタル画像処理編集委員会, ”ディジタル画像処理 [改訂第二版] ”.
   講義「デジタル画像処理」の教科書。画像処理関連の広い範囲の技術について、フルカラーで解説されている。

2. `https://docs.opencv.org/`.
   OpenCV の公式リファレンス。チュートリアルもあるので、必要に応じて参照しよう。

1 環境の導入

1.1 前提

演習室の環境（Ubuntu、Python3.8、IDE として Visual Studio Code (VSCode) 導入済み）で説明するが、どの環境でもだいたい同じである。以下、コマンド中で `python3` となっている部分は、環境によっては `python` の場合があるので（演習室は両方いける）、各自の環境に合わせて使ってほしい。

Python　Python をインストールしておくこと。Windows の場合、`https://python.org/downloads/` からでも良いし、Microsoft Store からもイントールできる。最新版をインストールしておけばたぶん問題ない。

統合開発環境（IDE）　統合開発環境（IDE）は、是非この機会に導入しておこう。Python の場合、VSCode や PyCharm[1] あたりがメジャー。

Git　下記で、カジュアルに「Git でリポジトリをクローンして...」というような操作が入っている。zip でリポジトリごとダウンロードして解凍して使っても良いが、Git まわりに触ったことがないならこの機会に是非触ってみると良い。Windows の場合は `https://gitforwindows.org/` よりインストールを。

1.2 演習用リポジトリのクローン

本演習用のリポジトリ `https://github.com/fumio125/enshu_ip` をクローンする。以下の通り。

```
$ git clone https://github.com/fumio125/enshu_ip.git
```

これで、カレントディレクトリ以下 `enshu_ip` 内に、Git リポジトリ内のファイルがコピーされた。

1.3 必要なライブラリの導入

仮想環境の作成　Python は、pip などの管理ソフトを使って簡単にライブラリパッケージを導入できる。ただ、全体の python 環境を汚すと後が面倒[2]なので、venv などのツール[3]を使って特定プロジェクト用の「仮想環境」を使ってから行うことが多い。

```
$ cd enshu_ip
$ python3 -m venv .venv
```

これで、Python 周りの環境（インタプリタなど）がごっそり `enshu_ip/.venv/` 以下にコピーされた。作成した仮想環境に含まれる Python インタプリタを使う場合は、

```
$ source .venv/bin/activate  # Ubuntu 等の場合
> ./venv/Scripts/activate    # Windows の場合
```

とすると、以降は `.venv` に入っている Python 環境に対する操作になる。

[1] どちらが良いかは戦争になるので言及を避ける。VSCode は様々な言語に対応しており、Python 用の拡張をインストールして使える。PyCharm は Python 専用。ざっくり言えば、高機能& Python 特化の PyCharm と、軽量&高汎用性の VSCode、という感じか（これはこれで怒られそう）。ちなみに、大倉は PyCharm 派であるが、今のタイミングで初めて統合開発環境を使うなら VSCode のほうがナウいし用途が広いので良いと思う。

[2] 他のプロジェクトと衝突したりする。特にライブラリのバージョン指定などに齟齬があると大変。

[3] Python だけでなく、もっと広い範囲を仮想化するようなツール（Docker など）を使っても良い。

必要なパッケージの導入　仮想環境を activate した状態で、以下のように OpenCV とその拡張パッケージ[4]、それ以外に使いそうなパッケージをインストールする。また必要になったら、必要なものをその都度入れたら良い。

```
$ python3 -m pip install -U pip  # pip のアップデート。1 回だけやれば OK。
$ python3 -m pip install opencv-contrib-python  # OpenCV
$ python3 -m pip install matplotlib, scipy, Pillow, jupyter  # その他のライブラリ
```

ちなみに、今回の場合は Git リポジトリに必要ライブラリのリストが `requirements.txt` として用意されているので、以下でも OK。

```
$ python3 -m pip install -r requirements.txt
```

1.4　IDE の設定とプログラムの実行

サンプル用のソースコードが `enshu_ip/hello.py` にある。当然、任意のテキストエディタでこれらを編集し、（↑の仮想環境に activate した状態で）コマンドライン上で Python を呼べば実行できる。たとえば：

```
$ python3 hello.py
```

とすれば、画像が描画されたウインドウが開く（何かキーボードから入力すれば実行終了する）。しかし、コーディングやデバッグの容易性などあらゆる面で IDE の方が優れているので、この機会に触れておこう。以下は VSCode の例だが、他の IDE でも大体同じような感じである。

基本的な流れ

1. VSCode を開き、File – Open Folder から、リポジトリのルートのフォルダ（`enshu_ip`）を開く。
2. View – Command Palette より、“Python: Select Interpreter” より、.venv: venv に対応するものを選択[5]。
3. 左（環境による）にある Explorer[6] より、`hello.py` をダブルクリック
4. 右上（環境による）にある再生ボタンっぽいものを押すと起動。なお、再生ボタンっぽいものの右にある矢印より、Debug を選択するとデバッガと一緒に起動できる（Run – Start Debugging からでもいける）。起動中に終了したい場合は、起動するとでてくるウインドウから停止ボタンっぽいものを押す。

Tips

- 作業中に、VSCode のプラグインが足りないと「〜をインストールしますか？」というようなことを聞かれるので、OK しておく。
- View – Terminal を開いておくと、指定されたインタプリタに対応する仮想環境に activate された状態の端末が開くので、pip とかできて便利（ターミナルの他に、エラーやプログラムの出力なども見られる）。
- コードエディタで適当に何かを入力すれば、その先の補完案や、関数引数の定義が表示される。これがないと生きていけない。例えば、`hello.py` の適当な行で `cv2.` とでも入力してみると良い。

[4] OpenCV の拡張パッケージが入ってないのは opencv-python。contrib には、特許関連で商用利用しづらいアルゴリズムや、新機能なども含まれる。
[5] VSCode を通して自動的に venv を作ってもらうこともできる。慣れたらそっちのほうが簡単。
[6] 見当たらない場合は左端のアイコンのそれらしいやつをクリック。

デバッグ　IDE を使うなら、ぜひデバッガを使おう。例えば、以下のような操作が可能なのでやってみよう。他にも様々な機能があるので、ぜひ遊んでみると良い。

1. コードエディタの任意の行の左端をクリックすると、赤い丸が現れる（ブレークポイント）。
   試しに、`hello.py` の 12 行目にブレークポイントを置き、デバッグ起動してみよう。
2. ブレークポイントで実行が一時停止される。ブレークポイントによる停止時（あるいはエラー停止時、例外発生時など）、変数リストやコールスタック（どの関数がどの関数を呼んで今に至るかなど）を見ることができる。
3. ターミナルウインドウ内のデバックコンソールより、適当な Python コードを入力すると実行できる。たとえば `src.shape` と入力すると、読み込まれた画像 `src` のサイズを出力できる。
4. Step Over のボタンを押すと、一行だけ実行できる。Continue を押すと、現在の場所から続きが実行される。

2　チュートリアル

OpenCV でできるごく簡単な処理について、Jupyter notebook で書かれたチュートリアルを用意している。Jupyter notebook は、インタラクティブな環境で python を試すことができるものであり、ノートブックに埋め込まれたコードスニペット（セル）ごとに実行できるため、Python プログラミングの学習や、ちょっとしたお試しプログラミング用に大変便利である[7]。各セルに記載されたコードは、通常の Python コード内でも動くので、課題プログラムの作成において、必要に応じて使用すると良い。

とりあえず、`tutorial.ipynb` を開いてみよう。コマンドラインからは

```
$ jupyter notebook
```

でブラウザ上で jupyter を起動し、`tutorial.ipynb` を開くことができる。

VSCode 上からは、Explorer から直接開いて編集や実行できる。以下、VSCode でノートブックを編集・実行する際の Tips である：

- 最初の実行時にインタプリタを選択する必要があるので、上述と同じ venv を選択すると良い。
- コードセル左上の再生ボタンっぽい何かを押すと、当該セルを実行できる。再生ボタン横の矢印より、デバッグ起動も可能。ブレークポイントも置ける。
- コードセル上で Shift + Enter すると実行し、次のセルにカーソルが合う。
- ハングして応答が帰ってこなくなったら、上部の「Restart」を押す。
- セルとセルの間にカーソルを持ってくると、Markdown（テキスト）や Code（コード）のセルを作成できる。

[7] 一方、大きなプログラムを書いたりデバッグしたりするのには向かないので、ある程度大きなプログラム（例えば今回の演習課題や卒業研究のプロジェクト）を書く場合は、普通の Python ファイルとして作成する（あるいは、実行用のインタフェース部分だけノートブック形式にするなど）ことをオススメする。

3　課題の作成・提出

さて、いよいよ課題の作成に入ろう。課題は以下の通りである。

3.1　課題

OpenCV を使って、なにか「**自分（あるいは誰か特定のユーザ）が得する**」システムを作る！

最低限の条件　下記、最低限の条件を守りながら、自由な発想で画像処理アプリを作成しよう。

- 画像処理を活用するプログラムであること。
- 諸々の資料を参考にしたり組み合わせるのは良いが、そのままコピーするのはダメ（`tutorial.ipynb` の内容は除く）。
- 参考にした資料があるなら、レポート中に（引用として）書いておくこと。

好ましい条件

- 実験用素材（入力画像など）は、構築したアプリを良くアピールできるような例を自分で撮影すると良い。
- レポートに、うまくいく例、うまくいかない例を示し、うまく行かない例を改善するための可能性を考察する。
- しっかりした問題意識や需要に基づくアプリの提案。ただし、その問題や需要が一般的であるかどうかは一切問わない。
- ユニークな発想。

3.2　提出用コード

`enshu_ip/08D12345.py` として、課題提出用のファイルを用意した。冒頭部に、必要事項を記入しよう。

- `submission_id` に学籍番号、`submission_name` に氏名を記入
- `submission_highlights` に、アピールポイントを簡潔に記入。
- 実行に準備（インストールや実行環境）が必要な場合、`submission_requirements` に記入。

あとは、適宜書き換えて課題を作成すること。

3.3　レポート

卒業論文の多くは、LaTeX で作成することが多いと思われる。この課題では、レポートを LaTeX で作ってみよう。レポートテンプレートがリポジトリ内の `report/08D12345.tex` にあるので参照すること。レポートのコンパイル方法やフォーマットに関する注意事項、内容例なども記載されている。参考まで、コンパイル後の pdf は `report/08D12345.pdf` にある。

- 提出時は “学籍番号.pdf” にリネームして提出すること。
- 添付のテンプレートで、ページ数は 1〜2 ページ程度を目安とする（が、それより長くても良い）。日本語か英語で記述すること。
- 本レポートのタイトルを、実装したアプリの内容をうまく表現するようなタイトルに適切に変更すること。

3.4 ヒント

何を作ればよいか思いつかない　`doc` ディレクトリ以下に、ちょうど良さそうな例をいくつか紹介したスライドを用意したので、それらに取り組んでも良い。ただし、それらにとらわれず、自由な発想で取り組むことをおすすめする。

どれくらいの規模感のものを作れば良い？　小さいもので構わない。上記のチュートリアルの内容を組み合わせた、あるいは少々書き換えただけのようなものでも、その有用性や実装内容がレポートで十分に説明されていれば問題ない。 ちなみに、`tutorial.ipynb` はあくまで参考資料の一つ、という立ち位置である。 よって、チュートリアルに記載されたコードを使う場合も、それ自体の説明もきちんとレポート中に含めること。

こういうのが作りたいんだけど、どのように実装すれば良いかわからない　大倉（okura@ist.osaka-u.ac.jp）までお気軽にご相談を。