KU LEUVEN

Labo Beeldinterpretatie

2018-2019

19/10/2018 - Introductiezitting

Steven Puttemans – steven.puttemans@kuleuven.be

Timothy Callemein – timothy.callemein@kuleuven.be

Doelstellingen

1. Beeldverwerkingsalgoritmes omzetten in praktische toepassingen gebruikmakend van C++ en OpenCV

2. Leren werken met het beeldverwerkingspakket OpenCV

3. Leren probleemoplossend denken op basis van opgedane kennis in de theorielessen beeldverwerking van Toon Goedemé

4. Zelfredzaamheid: gebruik maken van beschikbare media om softwareproblemen aan te pakken, parameters te tunen en te interpreteren.

5. Gebruik van een code management systeem, gebaseerd op GIT

Opbouw vak

- We werken dit jaar met een nieuwe insteek, waarbij we studenten verplichten om elke week de opdracht af te werken + ervaring op doen met GIT code management.

- Algemeen verloop
  - https://github.com/EAVISE/2018_labo_beeldinterpretatie
  - Per sessie komt daar folder met opdracht + nodige data
  - Als student maak je een eigen Github repository aan
  - Daarop plaats je jouw oplossing
    (deadline volgende woensdag 23u55)
  - Oplossingen automatisch binnen gehaald bij begeleiding

Examen

- We werken dit jaar niet meer met een examenopdracht van 4 uur, maar projectgebaseerd.
  - Dit project werk je uit tijdens je laatste lesweken / blok / examenperiode.
  - Mondelinge verdediging tijdens de januariexamens.
  - Beoordelingscriteria komen nog online.

- **30/11** – Lijst beschikbaar met onderwerpen en doorgeven van je persoonlijke top3 tegen woensdag **05/12**
- **07/12** – Je krijgt je definitieve opdracht toegewezen
- **14/12 – 21/12** – Mogelijkheid tot stellen inhoudelijke vragen, de technische oplossing is voor de student

Afspraken

- Labo’s
  - Formule van zelfstudie
  - Je krijgt een voorgeschoteld probleem met deelproblemen
  - Op basis van geziene theorie in lessenpakket
  - Ubuntu 18.04 + Code::Blocks IDE + OpenCV 3.4.x
- Gebruik je contacturen
  - Stel vragen als je niet kan volgen en stel dit vooral niet uit tot in December of Januari.
  - Zorg dat je je labo’s bijwerkt en niet hopeloos achterop loopt.
  - Ook naast contacturen is begeleiding via mail of op afspraak beschikbaar om extra ondersteuning te bieden. Zet beide docenten altijd in aan/cc tijdens communicatie.

Afspraken

- Tijdens elke labozitting
  - Je bent op tijd aanwezig (8u15)
  - Zorg op het einde van de zittingen dat je code op GIT staat
  - Elke student werkt op zelfde computeraccount, dus geen garantie dat je code ook daadwerkelijk blijft staan

- Na elke labozitting
  - Stoel onder tafel
  - Computer volledig uitgeschakeld
  - Scherm, toetsenbord, muis op juiste plaats
  - Scherm uitzetten

Programmeeromgeving

- Software
  - OpenCV bibliotheek: http://opencv.org/
  - Source code: https://github.com/opencv/opencv
  - We gebruiken `master` branch → OpenCV 3.4.x
    - Meest stabiele versie
    - Meest ondersteunde versie → bugfixes

- Software is in C++
  - We weten dat dit een nieuwe programmeeromgeving is
  - Maar met Python en C achtergrond moet dit lukken
  - Programmeerconcepten blijven immers identiek

Programmeeromgeving

- Ontwikkelomgeving Code::Blocks IDE
  - http://www.codeblocks.org/
  - Simpele maar bruikbare interface
  - Eenvoudig te configureren

- We starten telkens met een leeg `command line project`.

- Specifieke configuratie nodig per project
  - Configuratie van waar OpenCV op het systeem te vinden is
  - Configuratie van juiste compiler

Setup sample project

- Specifieke configuratie nodig per project

1) File > New Project > Command line project
2) C++ language
3) Kies gepaste naam & locatie (eigen map!)
4) GNU GCC compiler (zeer belangrijk voor C++ support)
5) Project Name > Right Click > Build Options
   > Linker Settings > Additional Includes

`*pkg-config opencv --libs*`

Setup sample project

- In de main.cpp file van je project

1) Zorgen dat OpenCV herkend word door je project

#include <opencv2/opencv.hpp>

2) Zorgen dat de nodige C++ bibliotheken herkend worden

#include <iostream> (output naar scherm)

3) Namespaces

using namespace std; *OF* *std::functie*

using namespace cv; *cv::functie*

(volgorde belangrijk voor naamconflicten)

Setup sample project

- In de main.cpp file van je project

4) Aanpassen van je main body

```
~~int main(){~~

int main(int argc, const char** argv){
```

5) Capteren van inputgegevens

```
/// Adding a little help option and command line parser input
CommandLineParser parser(argc, argv,
        "{ help h usage ?   |       | show this message }"
        "{ parameter p      |       | (required) message parameter }"
);

if (parser.has("help")){
        parser.printMessage();
        return;
 }
```

Setup sample project

- In de main.cpp file van je project

6) Verwerken inputparameters

```
/// Collect data from arguments
string parameter(parser.get<string>("parameter"));
if (parameter.empty()){
        parser.printMessage();
        return -1;
    }
```

7) Bemerkingen

- type kan varieren: string, int, double, float, …
- geef in je help mee welke parameters <*required*> *en* <*optional*> zijn, en check enkel inhoud waar nodig

Opdrachten

- Heel uiteenlopende beeldverwerkingstechnieken op basis van bestaand onderzoek binnen de EAVISE onderzoeksgroep

Datum	Uur + Locatie	Opdracht
19/10/2018	8U15 - 10U15 / A212	Introductiezitting + initiatie OpenCV
26/10/2018	8U15 - 10U15 / A212	Morphologische operatoren
09/11/2018	8U15 - 10U15 / A212	Color space manipulatie
16/11/2018	8U15 - 10U15 / A212	Template matching
23/11/2018	8U15 - 10U15 / A212	Keypoint detection and matching
30/11/2018	8U15 - 10U15 / A212	Machine learning: Naive Bayes, kNN, SVM
07/12/2018	8U15 - 10U15 / A212	Advanced object detection: boosted cascades, HOG+SVM
14/12/2018	8U15 - 10U15 / A212	Deep learning
21/12/2018	8U15 - 10U15 / A212	Deep learning

Problem solving in OpenCV

- OpenCV online documentatie
  - http://docs.opencv.org/master/
  - Zoekfunctie rechts bovenaan
  - Opgedeeld per module
- OpenCV questions & answers
  - http://answers.opencv.org/questions/
- Bij problemen
  1. Eerst denken →Wat doe ik en wat kan er verkeerd zijn?
  2. http://www.google.be
  3. Zorg dat je oplossingen zoekt voor OpenCV 3.x
  4. Indien dan niet werkt → vraag hulp

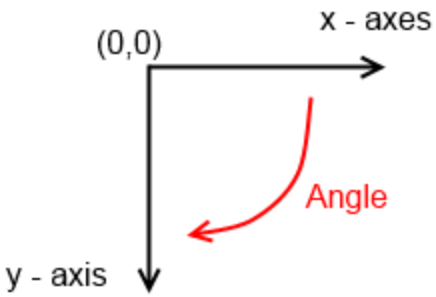
Problem solving in OpenCV

- `*Segmentation fault*`
  - Zowat de meest voorkomende melding binnen OpenCV
  - Meteen dan ook door studenten het meest opgemerkte en geprogrammeerde problem, dat vrij eenvoudig te debuggen is

- Hoe oplossen
  - Meestal 2-3 regels hoger staat een effectieve foutmelding
  - Zoek naar ASSERT() of .type() error
  - Indien geen extra info, start met waarden te visualiseren
    - cout << “foutboodschap” << endl;
    - cerr << “foutboodschap” << endl;
    - .type(), .size(), .empty()

OpenCV specifics

- OpenCV coordinate system



- Toegang tot beeldpixels

  .at<template>(row,col)

  .at<template>(y,x)

  CV_8S -> schar / CV_8U -> uchar

  CV_16S -> short / CV_16U -> ushort

  CV_32S -> int / CV_32F -> float

  CV_64F -> double

- Opgelet, niet elke functie werkt via het row,col principe

  Size(width, height) -> Size(x,y)

  Point(col, row) -> Point(x, y)

- waitKey()
  - interrupt caller
  - zorgt voor een verwerking van input signalen
  - # millisec / 0

- RGB versus BGR
  - Normaal RedGreenBlue
  - OpenCV BlueGreenRed
  - split()