pathways-extensions (training)

Release 20181014.20

Oct 14, 2018

Contents

This is about a *training* for (Young) **Software Engineers**; to become a **Professional Engineer**. They will work on a *big Project*: Team by team, they will work on (a part of an) extensions for Pathways. It is **not** a training on how to develop Pathways extensions!

CHAPTER 1

Part I: The training itself

This is a generic part: for all teams, and about all extensions. Most is written by the trainer (Albert), in his role as coach. It describes the training-concept, some rules and some hints.

It will mature in time. Each (future) team will build on the shoulders of the teams before. And so, the training will evolve as well. Like in a real-world project: each *sprint-retrospective* brings new insight, shifting the focus of improvement.

1.1 The training

English & Dutch

This part is partly in Dutch, and partly in English. It targets to Dutch "apprentices".

This repro and it clone/forks probably have a lot of info for the general public. All useful products will be promoted to "for real" repro.

But feel free to look around and learn! The https://bitbucket.org/ALbert_Mietus/pathways-extensions-training repro is the master version. All other (forks) are made by "apprentices" .

1.1.1 About the Training

A Big project . . .

Pathways is a concept and (reference) implementation for automatic verification (or *'testing'*) a product. Especially, during iterative development; like Agile and Scrum. It also fits in a Lean and Agile SW-development approach.

It is also extendable; anyone can write a *gate* to a new product (or PUT: Product Under Development). Such a *gate* has a small, generic interface: the minimal set of "command and queries" to operate that PUT.

Challenge

In these training such an extensions is made. Or more precise, as (small) part of it. The first extension is called *RekenRobot*, and currently the teams are working on the *DisplayLezen* part.

Remember: it should be (and is) is quite a big, challenging project! At least for junior programmers. They will encounter all kind of **engineering** (and *process*) problems and learn from it.

Code-Quality

The objective is to learn; not to write code quickly. Probably an experienced programmer with in depth knowhow on *robots*, *computer-vision*, *OCR* and *AI* can write this extension is a few days (or less). But it is all new for them ...

So, they will make mistakes. In 'engineering', in (software) architecture & design, and in code-quality.

And they get feedback; but not that much on the code itself. There are other trainings for that!

Version management (rules)

author Albert

...

Forks, Clones & Branches

Elke apprentice maakt zijn eigen copie op de (bitbucket) server (een *fork*) en kopieert die "persoonlijke" repro naar haar/zijn computer (een *clone*). Daar worden alle veranderingen gemaakt. Als nodig in meerdere (locale, named) Branches.

Commit, Pull & Push

Elke verandering in de code, of de documentatie, wordt natuurlijk opgeslagen (*save*) en getest. Af en toe wordt een "tussen-versie" lokaal bewaard in de repro (*commit*). Na een aantal van die iteraties zal 't "*af*-ig" zijn. Dan is het tijd om dat te delen; door een *push* naar de eigen repro. In de *fork* staan dan alle veranderingen (elke commit), maar pas nadat het een zekere (informele) status heeft.

Warning: Only source

This a about **source version control**; not about *asset management*. When a (professional) developer speaks about "all files"; (s)he doesn't really means *all* file, only **all sources**.

So, no downloaded stuff, no libs (*.ddl, *.lib *.so file etc); no *generated* files in general.

A 'source file' is a file that is needed to build the product. And typically typed by you or one of your fellow developers. So, it is hardly over a few KBytes big and in readable text format. Like code (*.py, *.c/c#/c++), textual-doc (*.rst, *.md, *.troff), and build-instructions or configuration (Makefiles, shell-scripts, ini-files) etc.

Please do not (hg) **add** other files, unless consulted with you trainer (beforehand). Also, **NEVER** add/commit files that is copyrighted ("nor yours"). This kind of flaws are very hard to fix. It basically means we have to abandon (delete!) your repro, as well as all that have 'merged' you mistakes.

This has happend once (my fault), lets learn from it: Only source!

Het delen van die files ("kennis" & structuur) gebeurd in twee stappen. Bovenstaande *push* is de eerste stap; die maakt de files beschikbaar voor anderen. Het overnemen gebeurd door een *pull*; waarbij we de veranderingen in de repro van een ander lezen naar de eigen PC. Vaak zullen die veranderingen geïntegreerd moeten worden met de eigen veranderingen. De veranderingen worden dan weer gesaved, ge-commit en uiteindelijk ge*pushed* naar de eigen repro op de server.

Deze cycle kan/moet iedereen naast *–maar niet na–* doen. Conceptueel kan iedereen wel de veranderingen van een ander lezen, maar alleen schrijven naar de eigen repro's. (zowel op de PC als op de server)!

Pull-Request

Na een intergratie, zal er vaak behoefte zijn om die nieuwe versie te delen op de *"master-repro"* (https://bitbucket.org/ALbert_Mietus/pathways-extensions-training); dit gebeurd met een **pull-request**. Dit is een verzoek aan eigenaar van een andere fork, om de fork van de aanvrager te *pull*en, te integreren en te pushed naar zijn (master) server-repro. In veel teams gebeurd door die persoon aan te spreken, te bellen of te mailen. Of door dit verzoek op bitbucket-server te doen; voordeel is dat dan dat alle details automatisch bekend zijn; ook werkt dat handig als de betrokken mensen elkaar nooit zien/spreken.

Project Kaders

author Albert

Platform

De software **moet** (uiteindelijk) werken op een (elke) RaspberryPi. Het **moet** ook werken op elk PC-type computer (Mac, Unix/Linux en Windows); vooral om het ontwikkelen eenvoudiger te maken. (Ook voor demo's en testen)

Taal

De programmeertaal **mag** gekozen worden uit: Python(3)[1] en/of C*[2]; vooral de aanwezige (taal)kennis van de studenten is van belang.

Wel **moet** altijd een (kleine) **Python-3 API** gemaakt worden. Zodat de gehele robot vanuit python aangestuurd kan worden.

Verder:

- Alle documentatie **wordt** gemaakt met Sphinx; in rst-files en/of in Python docstrings,
- Alle files **moeten** opgeslagen worden in UTF-8, met standard (unix) regeleindes (dus zonder `^M`; zoals op Windows)

GUI/Editors

De keuze voor een editor is vrij. Wel moet deze alle alle "file-eisen" ondersteunen (zoals `^J` regeleindes, utf-8, indentatie met spaties, etc).

[1] Python heeft op dit moment de voorkeur. Vooral omdat de meeste deelnemers weinig (C) programmeer-ervaring hebben (zeker in de C-taal).
[2] C*: C, C++, Objective-C, of C#; de laatste (twee) zijn hier nauwelijks een optie gezien het platform.

Kwaliteit

TDD

Het gebruiken van de *Test-Driven-Development* aanpak is **verplicht**; zowel op unit-, module- en systeem-level.

- Zorg dat ieder (onder)deel los van andere testbaar is (*decoupling*)
- Voor elke functie, class en/of file moeten *"voldoende"* unit-test geschreven worden
  - Gebruik bij voorkeur pytest
  - *"Voldoende"*:
    - Elke regel, elk statement moet minimaal eenmaal doorlopen worden (in een test)
    - Voor een `if` zijn dus minimaal 2 tests nodig; bij geneste `if's` verdubbeld dat telkens!
    - Niet het *aantal* testen is belangrijk, maar de zekerheid dat code (wijziging)
- Ook modele (andere) levels moeten voldoende (automatiche) testen opgeleverd worden; die aantonen dat 't werkt.

Code-kwaliteit

- Het gebruik van pylint en/of andere code-checkers is **verplicht**. Bij elke (sprint) oplevering moet de kwaliteit aangetoond worden
- Code reviews zijn **verplicht**; zowel op functionele correctheid, als lees- & onderhoudbaarheid. Toon dit aan! (gebruik de bitbucket fasaliteiten)

  Dit geldt ook voor documentatie; behalve voor (eigen) 'team-pages'

1.1.2 Getting started

Note: This part is mostly provided by the *SeaBroomPhi team* and focuses on the RekenRobot.

- It is (structurally) updated to integrate it by Albert4
- Future teams should update it! (both context & structure)

Last update: sPYinkenHof (07-02-2018)

Section author: SeaBroomPhi

This section describes how to get the software and tooling used in this project/training up and running.

Summary

Version control

Version control allows you to work as a group on one final version on your own hard drive and not in a browser. This is done with the use of Bitbucket and Mercurial. Bitbucket is the host repository where all versions are stored, Mercurial is the tool to share this data. Before files are shared it is possible to show a local version of the documentation on your device by using Sphinx. This tool can transcript reStructuredText files (.rst) to a HTML website.

Documentation using Sphinx

Documentation is written in plain text files using the .rst format. These files can be converted to HTML using a python package called Sphinx

Raspberry Pi

The Raspberry Pi is a minicomputer which will be used to run the RekenRobot software. At the start of the course you will receive on from the teacher (Albert). Some of the Pi's may already have software installed on it and it could be ready to run. If this is not the case, you will have to install a number of programs in order to run the RekenRobot software on the Pi. Programs needed to interact and use the Raspberry Pi are described below.

Editors

An editor is a software tool used to edit certain types files. For example, if you would want to edit a python file you would need an editor to do so. Two editors will be described in this section: a Python editors and an all-round editor.

Running the Demo

Ultimately, the RekenRobot will become a "plug in" of the Pathways-project. For now, a (standalone) demonstrator is used.

Although it should be run on the RaspberryPi; one can try it first on the PC.

Getting to work . . .

Version Control

Wat is versiebeheer, en wat heeft dat met jou te maken? Versiebeheer is het systeem waarbij veranderingen in een bestand of groep van bestanden over de tijd wordt bijgehouden, zodat je later specifieke versies kan opvragen. Het doel van versiebeheer is om als groep met de laatste versie op je harde schijf te werken en niet op de browser te werken. Dit doen we door Bitbucket en Mercurial te gebruiken. Sphinx is een tool om documenten van reStructeredText files(.rst) om te zetten naar bijvoorbeeld een HTML website.

Bitbucket

Maak een account op Bitbucket.org om de codes met elkaar te kunnen delen. Als je een account hebt gemaakt, zoek naar Pathways-Extension (Training) van Albert Mietus. Op de hoofdpagina van Pathways-Extension (Training) staat de algemene informatie over dit project.

Fork

Iedereen die aan het project werkt maakt zijn eigen Fork op Bitbucket: Fork > De standaard instellingen staan goed en maak je Fork repository. Jou FORK staat nu ook op de hoofdpagina en op je eigen frontpag. Wanneer je al een Fork gemaakt hebt en je wilt de naam veranderen: Ga naar je eigen Fork pagina, Settings (linksbeneden).

Clone Als je clonet zet je een kopie van wat er op de online repository staat op je eigen PC. Op deze kopie kan je vervolgens wijzigingen aanbrengen en pushen naar je online repository. Zorg ervoor dat je je eigen repository cloned, want dat is de enige repository waar je schrijfrechten op hebt.

Clonen doe je door op je eigen pagina op 'clone' (linkerkant) te drukken. Kopieer de link en open de command prompt. Indien de knop 'clone' niet aanwezig is, kan je ook je URL nemen. Verander de locatie in je command prompt als je dat wilt en tik "hg clone" in en plak erachter de link:

```
hg clone [link]
```

Om dit commando te gebruiken moet eerst Mercurial zijn geïnstalleerd, zie de link bij Mercurial hieronder.

Master

De master van dit project is Albert Mietus en zorgt dat alles dat erop komt, goed staat. De Master accepteert alle wijzigingen die door de anderen zijn gedaan.

Mercurial

Mercurial is te downloaden op https://www.mercurial-scm.org/. Als je iets wilt aanpassen, pull dan de laatste versie en merge het met die van jou. Zorg wel ervoor dat jouw status gecommit is en dat je de laatste versie hebt om die met een ander te mergen. Door dit te doen is jouw Local up to date. Hierna kan je de aanpassingen toevoegen en committen. Uiteindelijk heb jij de laatste versie met de aanpassingen en kan je het naar de anderen pushen. Door te pushen kunnen andere teamleden de laatste versie van jou pullen en ermee verder werken.

Hieronder staan kort de basis commandos van Mercurial beschreven. Voor meer informatie zie: https://www.mercurial-scm.org/wiki/Tutorial

Pull and Push

Bij een pull zet je de wijzigingen van een andere online repository in jouw eigen locale repository. Dit doe je in de command line met:

```
hg pull [link]
```

waarbij [link] de URL is van een andere repository. Deze kun je vinden op bitbucket.

Bij een push zet je jouw gecommitte wijzigingen op je eigen online repository. Dit doe je door middel van:

```
hg push
```

Merge, Commit and Add

Bij een commit zet je je eigen wijzigingen klaar om gepusht te worden naar jou repository. Het is belangrijk dat je dit doet voordat je pullt en pusht. Committen doe je met:

```
hg commit
```

Hierbij opent er een txt bestand waarin gewijzigde documenten zichtbaar zijn. Type bovenaan een bericht en sluit het document. Hierna zullen je wijzigingen gecommit zijn.

Als je add worden nieuwe of verwijderde bestanden klaargezet voor je commit. Als je dit niet doet, worden ze niet meegecommit. Gebruik hiervoor:

```
hg add
```

Met een merge voeg je gepullde wijzigingen van een andere repository samen met je eigen wijzigingen. Dit doe je met

```
hg merge
```

Status

De status geeft weer of er gewijzigde bestanden zijn die nog gecommit moeten worden. De status wordt weergegeven met:

```
hg status
```

Hierbij zie je bestanden samen met de status hiervan. De betekenis van alle statussen staat hieronder. Zorg ervoor dat als er bestanden zijn die niet Modified zijn dat je hg add gebruikt.

Symbol	Description
M	Modified
A	Added
?	Unknown
!	Missing
I	Ignore
R	Removed

Grafisch interface

Als iets niet lukt bij in command prompt, kun je ook het grafische interface van TortoiseHg gebruiken. Dit kun je gebruiken door op je computer naar de directory te gaan waar de rekenrobot repository staat. Vervolgens kan je rechtsklikken en commando's van Hg gebruiken, zoals hg commit of hg status.

Sphinx-doc

Overview

Op deze pagina: http://www.sphinx-doc.org/en/stable/install.html#windows-install-python-and-sphinx staan de stappen die je moet maken om sphinx te installeren op windows. Als alle stappen zijn voltooid Sphinx worden gebruikt vanuit de command-line. Heel globaal worden de volgende stappen gebruikt:

1. Python installeren
2. Sphinx (inclusief plugins & extensies) installeren
3. `Sphinx-build` aanroepen

Daarna komt nog een klein stukje over editors e.d. om de documentatie toe te voegen

Python installeren

Dit kan worden gedaan via een URL op de website. Je doorloopt hierbij een installer. Het is hier belangrijk dat je aanvinkt dat je Python wil toevoegen aan environment variables. Ook zal een optie zijn om pip te installeren. Dit moet je ook aanvinken. (Het installeren via Anaconda is ook een mogelijkheid)

Note: Gebruik altijd de laatste, stabiele python3 versie!

Sphinx (& extensies)

Sphinx is geschreven in python en wordt met de python-package manager **pip** geïnstalleerd. Let op dat je _pip3_ gebruikt, als je ook een python2 geïnstalleerd hebt.

```
pip install sphinx
```

```
pip install sphinxcontrib.napoleon
pip install sphinxcontrib-plantuml
```

Mocht het niet lukken om PlantUML te installeren dan moet je eerst pip updaten:

```
pip install --upgrade pip setuptools
```

Plantuml (de jar file)

De plantuml-extensie gebruikt het java programma PlantUML. Dat je kunt downloaden van: http://plantuml.com. PlantUML gebruik (onderwater) "graphviz" (ook wel: dot, dotty); zie: https://www.graphviz.org. En natuurlijk heb je java nodig - dat is gelukkig redelijk standaard. En je moet al die zaken correct aan elkaar *knopen*.

Al met al is dat best lastig. Maar er zijn wat tussenstappen mogelijk. Zorg dat je eerst PlantUML (zonder Sphinx) aan de slag krijgt. Dat staat goed beschreven op: http://plantuml.com/starting

De laatste, lastigste, stap is PlantUML automatisch laten aanroepen vanuit Sphinx; dat doet de eerder genoemde extensie, aam de hand van de `conf.py` file. Die laatste mag je niet aanpassen; als kan het handig zijn om dat lokaal tijdelijk te doen, als je er anders niet uitkomt.

Deze laatste stap hoort "out-of-the-box" te werken, mits je *plantuml.jar* op een van de standaard plekken geïnstalleerd hebt. Of je een (bash, hulp) scriptje `plantuml` gebruikt.

> **Warning:** TOOLS ARE NOT PART OF THE SOURCE
>
> Some advices "on the internet" will you to place the plantuml.jar file *inside* the project environment. And commit it to version-controll...
>
> THAT IS WRONG! You are not support (nor allowed) to do so!

Documentatie builden

Documentatie builden betekent dat je de rst bestanden omzet naar een werkende website die je lokaal kan bekijken. Om te builden, ga in de command-line naar `...\pathways-extensions-training\docs` Eenmaal hier type het volgende commando in:

```
sphinx-build -c doc -b html doc __result\html
```

Na de build is er een build directory (`__result/html`) toegevoegd. Ga hier in en open `index.html`. Hiermee kan je de documentatie op de website bekijken. Iedere keer als je nieuwe wijzigingen wil bekijken of testen moet je dit build commando gebruiken.

Note: Die gegenereerde file (`__result/...`) moeten natuurlijk niet gecommit worden.

In de standaard configuratie gebeur dat niet. Doordat die directory opgenomen in de `.hgignore` file

---

Documentatie aanpassen

Notepad++

Notepad++ is de editor die gebruikt wordt voor het bewerken van rst en andere typen bestanden van de documentatie. Deze kan gevonden worden op: https://notepad-plus-plus.org/download

Using RST

Sphinx heeft een eigen documentatie over het gebruik van RST. Deze kan hier worden gevonden: http://www.sphinx-doc.org/en/stable/rest.html Naast deze documentatie zijn er ook andere voorbeelden te vinden op het internet.

Theme veranderen

> **Warning:** Not needed
>
> You may ignore the remainder of this article
>
> There is no need to change the *theme*. The theme only specifying the layout and colors of generated documentation. By having other colors for a local build and the official (RTfD) ones, it easy to see to which version you are looking!
>
> Whenever, you give it a try; be sure you **NEVER** commit conf.py!

Het thema zorgt ervoor dat je build er precies hetzelfde uitziet als de live website. Hiervoor moet je twee stappen volgen:

1. Installeer het correcte thema:

```
pip install sphinx_rtd_theme
```

2. Voeg het volgende toe in conf.py:

```
html_theme = "sphinx_rtd_theme"
```

Belangrijk: Zorg ervoor dat je deze wijziging in conf.py niet meecommit. Gebruik dit alleen om de build lokaal te testen.

Als deze stappen gevolgd zijn kan je het nogmaals builden. Als je de site dan bekijkt, zul je zien dat het thema hetzelfde is als de live website.

Raspberry Pi

Before you will be able to run the RaspberryPi with the RekenRobot functions, you will have to install different packages. Below is mentioned which packages are needed, what they do and how to install them.

Connecting to the RaspberryPi using SSH (Windows)

Putty allows you to enter the command line of the RaspberryPi on you laptop. This allows you to open and run files on your RaspberryPi. Putty should be installed on your laptop and can be found on http://www.putty.org/ For the first connection between Putty and your RaspberryPi it is important to join the same network, either with wifi or cable. The connection is made by knowing the IP address of the RaspberryPi. This IP address can be found by connecting the RaspberryPi to an external screen. The IP address can be found by:

Open the terminal, give the command ifconfig and the IP address will be given for each connection.

Knowing the IP address a wireless connection between the RaspberryPi can be made, keep in mind that both devices have to be connected to the same network. Opening Putty will allow you to fill in a host name. Type in pi@(IP address) and click open. In the next terminal a password will be asked, this is rekenrobot. if the terminal responds with pi@Raspberrypi:~S the connection was successful and you will be able to give commands to the RaspberryPi.

connect to the RaspberryPi using SSH (other OS)

On linux and MacOS is ssh mostly installed by default and can be run via the command line.

connect to the Raspberry pi using VNC

For explanation and installation of the VNC software click on the link below.

VNC (Virtual Network Computing)

In dit document is beschreven hoe VNC opgezet kan worden om de RaspberryPi te besturen vanaf de computer.

- In RaspberryPi: Open terminal.

```
$ sudo raspi-config
```

- In het configuratie scherm: Zet 'Enable Camera' tot 'Yes' en zet in 'Advanced Options' -> 'SSH' en 'VNC' op 'Enable'.
- Voer uit in de terminal

```
$ sudo apt-get update
$ sudo apt-get install tightvncserver
$ vncserver :1
```

- Op je laptop download en installeer VNC Viewer: vanhttps://www.realvnc.com/en/connect/download/viewer/
- Open VNC Viewer en start een nieuwe connection en voer als 'VNC-Server' [ip-address van de Raspberry]:1.

Vanaf nu kan je via VNC Viewer je Raspberry Pi besturen.

Nu moeten we nog instellen dat de VNC-Server draait vanaf boot zodat we in het vervolg alleen nog maar de raspberry pi nodig hebben.

- In de Raspberry Pi terminal, voer het volgende in:

```
$ cd /home/pi
$ cd .config

$ mkdir autostart
$ cd autostart
$ sudo nano tightvnc.desktop
```

- Voer in deze nieuwe file het volgende in:

```
[Desktop Entry]
Type=Application
Name=TightVNC
Exec=vncserver :1
StartupNotify=false
```

- Verander het wachtwoord van de RaspberryPi

```
$ passwd

Password = raspberry
new password = rekenrobot
```

Nu hoeven we alleen nog een static IP in te stellen zodat we de VNC Viewer een profile kunnen aanleveren.

Voor een wireless verbinding:

Op het Sogeti kantoor in Amersfoort heeft High Tech zijn eigen netwerk genaamd High Tech. De Raspberry Pi kan verbinding maken met dit netwerk en een statisch IP reserveren.

- In de terminal

```
$ sudo nano /etc/network/interfaces
```

- Zoek de wireless interface (wlan0).
- Noteer daar het volgende:

```
iface wlan0 inet static
address 192.168.1.[uniek nummer tussen 200 en 254]
subnet 255.255.255.0
gateway 192.168.1.254
```

- Save deze file met Ctrl+O en sluit hem af met Ctrl+X

Voor een bekabelde verbinding:

- Voer in de terminal het volgende in:

```
$ route -ne
```

- Noteer de Gateway IP.
- Voer nu in:

```
$ sudo nano /etc/resolv.conf
```

- Noteer de Domain Name Server IP's
- Voer nu in:

```
$ sudo nano /etc/dhcpcd.conf
```

- Voeg het volgende toe aan het eind van de file:

```
interface eth0
static ip_address=[De gateway IP van de route-ne maar verander het laatste getal naar␣
↪243]

static routers=[De Gateway IP van de route -ne]
static domain_name_servers=[De Domain name Servers van de resolv.conf gescheiden met␣
↪een spatie]
```

- Save deze file met Ctrl+O en sluit hem af met Ctrl+X
- Reboot de Raspberry Pi door het volgende in de terminal in te voeren:

```
$ sudo reboot
```

Vanaf nu heb je geen scherm, toetsenbord of muis meer nodig. Je kan bij het opstarten de VNC-Viewer opstarten met de informatie van de bovenstaande stappen. (VNC-Server = [static ip_address]:1)

Setting up the RaspberryPi work enviroment

The Raspberry Pi that is part of this project should be already setup correctly. However it could be that stuff is broken or a new Pi is added to the project and you want to setup the Pi (again).

There are two option for this. A short and easy way and long (should be easy but can be difficult way).

Option 1 is to flash a pre-formated image to the SD-card with all programs and settings ready to go.

Option 2 is to follow the steps below and do a clean install of all software.

Virtual Environment

Nu installeren we een virtual environment.

- Voer het volgende in de terminal in:

```
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/.cache/pip
```

- Open de '.profile' file door middel van het volgende commando.

```
$ sudo nano ~/.profile
```

- Voeg aan het eind van deze file het volgende toe:

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

- Update de file door het volgende commando:

```
$ source ~/.profile
```

Attention: Dit commando moet gebruikt worden elke keer dat de terminal opnieuw opent!

- Voer het volgende in de terminal in:

```
$ mkvirtualenv rr -p python3
```

Attention: Om in de virtuele omgeving te werken gebruik je het commando:

```
$ workon rr
```

Je weet dat je in de virtuele omgeving zit door de aanduiding (rr) aan het begin van elke regel in terminal.

OpenCV

Beschrijving

OpenCV (Open Source Computer Vision Library) is een open source computer visie en machine learning software library. Het is gemaakt om het gebruik van perceptie door middel van machines te vergroten.

Note: It seems that the first part of this page describes how OpenCV can be downloaded in Windows. If you want to install it on the pi, just start at "OpenCV Installeren"

Download:

http://opencv.org/downloads.html (ik heb 3.1.0. Gebruikt, 3.2.0 is beschikbaar)

Voor het installeren van openCV moet je de volgende stappen nemen:

- OpenCV installeren in je libraries
  - Open Pycharm>File>Settings>Project:pathways-extension-training>Project Interpreter
  - Kies als Project Interpreter bovenaan voor: 3.5.2 (C:Users'yourname'AppDataLocalProgramsPythonPython35-32python.exe)
  - Klik op de plusteken aan de rechterkant om een package te installeren en installeer opencv-python.
- Het programma OpenCV zelf installeren - Nu moet je een bestandje van de ene naar de nadere folder verplaatsen:
  - In de map opencv/build/python/2.7 kun je het bestand cv2.pyd vinden. Kopier dit bestand
  - Plak het in de volgende map: C:/Python35/lib/site-packages.
- Om te kijken of het werkt, gebruik je console of IDLE en check het versienummer:
  - import cv2
  - print (cv2.__version__)

  Als dit werkt, is het gelukt om OpenCV te installeren.

OpenCV Installeren

Nu gaan we OpenCV installeren. Deze stap zal langer dan een uur duren, neem dit mee in je planning.

Eerst gaan we wat ruimte vrij maken.

- Voer het volgende in de terminal in:

```
$ sudo apt-get purge wolfram-engine
```

Hiermee verwijder je wolfram en maak je ~700 Mb vrij.

- Upgrade bestaande packages door de volgende commando's in de terminal in te voeren:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Dependancies installeren

- Voer de volgende regels in terminal in:

```
$ sudo apt-get install build-essential cmake pkg-config
$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
$ sudo apt-get install libgtk2.0-dev
$ sudo apt-get install libatlas-base-dev gfortran
$ sudo apt-get install python2.7-dev python3-dev
```

Nu gaan we OpenCV downloaden.

- Voer het volgende in de terminal in:

```
$ cd ~
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
$ unzip opencv.zip
$ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.1.0.
↪zip
$ unzip opencv_contrib.zip
```

Nu installeren we pip om python packages te installeren.

- Voer het volgende in de terminal in:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
```

Compileren en installeren van OpenCV

Attention: Dit onderdeel duurt 1h12 min op de Raspberry Pi 3 en 1h35 op de Raspberry Pi 2!

- Zorg dat je in de virtuele omgeving zit door het volgende commando te gebruiken (zie *Virtual Environment* voor meer info):

```
$ workon rr
```

Als je in deze omgeving nog geen numpy heb geinstalleerd doe dat NU via dit commando:

Numpy is a library adding support for large multi dimensional arrays and matrices, and allowing to operate this arrays. This is needed to operate the displaylezen module, for specific foto editing such as the gaussian blur, to optimize the image processing. For installation you can either download the package from numpy.org or via the command line by entering the following command.

```
> pip install numpy          # This command will install numpy and set it in your␣
↪python path
```

```
$ pip install numpy
```

- Nu gaan we de build klaarzetten met de volgende commando's:

```
$ cd ~/opencv-3.1.0/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \ -D CMAKE_INSTALL_PREFIX=/usr/local \ -D INSTALL_
↪PYTHON_EXAMPLES=ON \ -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.1.0/modules \ -
↪D BUILD_EXAMPLES=ON ..
```

(continues on next page)

(continued from previous page)

- Nu compilen we OpenCV met alle 4 kernen door het volgende commando:

```
$ make -j4
```

Dit is het gedeelte wat 1h12min in beslag neemt!

- Nu gaan we OpenCV installeren met de volgende commando's

```
$ sudo make install
$ sudo ldconfig
```

- Nu gaan we een filename veranderen om latere bugs te voorkomen. Voer het volgende in je terminal in:

```
$ cd /usr/local/lib/python3.4/site-packages/
$ sudo mv cv2.cpython-34m.so cv2.so
```

- En we sym-linken onze OpenCV met onze virtuale omgeving.

```
cd ~/.virtualenvs/rr/lib/python3.4/site-packages/
ln -s /usr/local/lib/python3.4/site-packages/cv2.so cv2.so
```

- Nu kun je de installatie testen door het volgende in de terminal in te voeren.

```
$ source ~/.profile
$ workon rr
$ python
>>> import cv2
>>> cv2.__version__
```

Als dit zonder errors '3.1.0' returned is de installatie succesvol.

Tesseract OCR

- Install Tesseract OCR with the following command:

```
$ sudo apt-get install tesseract-ocr
```

- Install pytesseract with the following command:

```
$ sudo pip install pytesseract
$ sudo pip3 install pytesseract
```

- Install python imaging:

```
$ sudo apt-get install python-imaging
```

- Install imutils:

```
$ sudo pip3 install imutils
```

- Install Pillow:

```
$ sudo pip3 install --upgrade pillow
```

Note: Upgrade pillow might give an error. In that case, first install pip as described in OpenCV.rst (after "Nu installeren we pip om python packages te installeren.")

- Reboot the Raspberry Pi.

```
$ sudo reboot
```

- Download letsgodigital.traineddata:

```
$ wget -O letsgodigital.traineddata "https://github.com/arturaugusto/display_ocr/blob/
↪master/letsgodigital/letsgodigital.traineddata?raw=true"
```

- Put letsgodigital.traineddata in the right directory:

```
$ sudo mv letsgodigital.traineddata /usr/share/tesseract-ocr/tessdata
```

- Make symbolic links with the installed packages and the virtual environment.

```
cd ~/.virtualenvs/rr/lib/python3.4/site-packages/
ln -s /usr/local/lib/python3.4/site-packages/imutils imutils
ln -s /usr/local/lib/python3.4/site-packages/PIL PIL
ln -s /usr/local/lib/python3.4/site-packages/pytesseract pytesseract
```

Note: At this point, you are finished with the installation of tesseract OCR! The following paragraphs give some more information about the training and preprocessing of the data.

Aan de slag

Wanneer je tesseract hebt geinstalleerd, is de stap snel gemaakt om een programmaatje te schrijven waarmee plaatjes waar cijfers op staan om te zetten in een string. Echter, voor deze opdracht willen we zeven-segment cijfers kunnen inlezen. Hiermee heeft tesseract wat meer moeite, omdat hij dit lettertype niet kent. Hiervoor moet een dataset getraind worden.

Training van de tesseract data

Tesseract is in staat om letters te lezen die hij "geleerd" heeft door middel van een trainingsdataset. Deze trainingsdatasets zitten in het mapje "tessdata", en eindigen met .traineddata. Met nummers op een zeven segment display heeft tesseract moeite, omdat deze nummers nog niet getraind zijn. Het is mogelijk om een getrainde dataset van internet af te halen en in deze directory te plakken (bijvoorbeeld https://github.com/arturaugusto/display_ocr/tree/master/letsgodigital). Een andere optie is om zelf (door middel van scriptjes die op internet te vinden zijn) de data te trainen.

Voorbewerken van de plaatjes

Tesseract lijkt moeite te hebben met de "witte" stukken tussen de letters, wat kenmerkend is voor een zeven segment display. Het voorbewerken van de plaatjes lijkt de performance van van tesseract aanzienlijk te verbeteren, zoals een blur of een threshold (zie http://stackoverflow.com/questions/28935983/preprocessing-image-for-tesseract-ocr-with-opencv voor wat voorbeelden).

Raspberry Pi Camera

- Connect de Raspberry Pi camera met de Raspberry Pi.
- Als je nog niet in de virtualenv zit, voer dan het volgende in de terminal in:

```
$ source ~/.profile
$ workon rr
```

- installeer de picamera module met de volgende commando's:

```
$ pip install "picamera[array]"
```

- Om te testen of je camera functioneert kun je het volgende script gebruiken:

```
# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2

# initialize the camera and grab a reference to the raw camera
camera = PiCamera()
raw_capture = PiRGBArray(camera)

# allow the camera to warmup
time.sleep(0.1)

# grab and image from the camera
camera.capture(raw_capture, format="bgr")
image = raw_capture.array

# display the image on screen and wait for a keypress
cv2.imshow("Image", image)
cv2.waitKey(0)
```

- mocht deze code crashen probeer:

```
$ sudo rpi-update
```

hierdoor zullen alle raspberry pi drivers and firmware worden geupdated

RekenRobot

To be able to use the RekenRobot, the code should be installed on the raspberry pi.

- Download Mercurial

```
$ sudo apt-get install mercurial
```

- Clone the most recent version of the RekenRobot

```
  $ hg clone <link>

Een copie van jou locale repository staat nu op de RasberryPi en kan uitgevoerd␣
↪worden.
```

Setup on laptop

Python

Install python3, a 64 bits version (3.6.4 is available on time of writing)

Tip: install python in C:python36 instead the default directory. Installing Python can also be done using Anaconda.

Open cmd to work via the commandline. Enter the following commands to modify the path settings:

```
> pip install pillow          # This command will install pillow and set it in your␣
↪python path
> pip install imutils         # This command will install imutils and set it in your␣
↪python path
> pip install numpy           # This command will install numpy and set it in your␣
↪python path
> pip install pytesseract     # This command will install pytesseract and set it in␣
↪your python path, you will need this for the step of this manual
```

PyTesseract

Now you will need to go to your python folder that will be in: C:\Users\%username%\AppData\Local\Programs\Python\Python%version%
packages\pytesseract

or go to

C:\Python36\Lib\site-packages\pytesseract

In this folder you will find pytesseract.py, you have to open this with your editor and there you will have to change line 26.

It will have to look like this:

```
# CHANGE THIS IF TESSERACT IS NOT IN YOUR PATH, OR IS NAMED DIFFERENTLY
tesseract_cmd = r'C:\\Program Files (x86)\\Tesseract-OCR\\tesseract.exe'
```

Now you will need to install tesseract, you can download tesseract from his link: https://github.com/UB-Mannheim/tesseract/wiki If you download tesseract-ocr-setup-3.05.01.exe just follow the steps from the installation.

To add the data of the calculator's font, you have to go to this link: https://github.com/arturaugusto/display_ocr/blob/master/letsgodigital/letsgodigital.traineddata and click on dowload. Now you have to copy what you just downloaded and paste it in the Tesseract-OCR\tessdata folder (for example C:\Program Files (x86)\Tesseract-OCR\tessdata).

OpenCV

To install opencv open the cmd and type:

```
> pip install opencv-python
```

And you will need to download it from the link: http://www.lfd.uci.edu/~gohlke/pythonlibs/ and download opencv_python3.4.0cp36cp36mwin_amd64.whl (or win32 depending on the python installation) then open the CMD terminal to the download directory and type in the following:

```
> pip install opencv_python3.4.0cp36cp36mwin_amd64.whl    # This command will install␣
↪opencv and set it in your python path
```

if this does not work, it says somethink about incorrect version; try running

```
> pip install --upgrade pip setuptools
```

Roundup

To round up the setup: Check if python is added to your path: Advanced system settings –> Environment Variables –> Path. Also check if TESSDATA_PREFIX is added as a Variable.

Editors

An editor is a software tool used to edit certain types files. For example, if you would want to edit a python file you would need an editor to do so. Two editors will be described in this section: a python editors and an all-round editor.

Eclipse

Eclipse is a code-development tool. It can be used for languages like Java or C. Eclipse can be downloaded from: https://www.eclipse.org/downloads/

Java is required in order to install Eclipse. If it isn't installed, Eclipse will open up a browser tab and prompt you to do so. Please note that you must install the 64-bit JDK in order for it to work.

Eclipse by default does not allow you to work on Python files. This must be done via a plugin which can be installed from the Eclipse marketplace. Go to: `Help > elcipse Marketplace.`

Here you can search for **pydev**. Only one plugin should be returned. Select this one and follow the installer in Eclipse. After this you will be able to edit python files in Eclipse.

To Import the project into eclipse go to: `File > Open projects from file system`

After this, search for the Rekenrobot project and it will be imported.

Notepad++

Notepad++ is an allround editor which can be used to edit a lot of different types of files, such as Python files. Python editors however are a lot more effecient to edit Python files with, so Notepad++ isn't used for that. Instead, it is mostly used to edit RST files.

To install Notepad++ go to the following Link: https://notepad-plus-plus.org/download/

After downloading and going through the installer you'll be ready to use it. When you right click on a file and select 'edit with notepad++', you'll be able to edit the file.

Other editors

The two editors described above are examples of editors you could use to work on the project. However, there are more different possibilities. For example, Pycharm is a different Python that could also be used.

Keep in mind however when choosing different editors how this could affect the team's progress and the sprint planning.

Run the demos

Ultimately, the RekenRobot will become a "plugin" of the pathways-project; running on a PC and/or a raspberryPi. For now, a (standalone) demonstrator is used. Which should run on the raspberryPi. We start with in on the PC however; *if it doesn't run there, running it on the raspberryPi is even harder*.

- Open cmd to work via the commandline
- Enter the following commands:

```
> cd ~/pathways-extensions-training/RekenRobot/Src/DisplayLezen/Demo          #␣
↪naviagate to the right directory
> python MainfileReadimage.py
```

on a raspberryPi

After proper installation of the tools and RekenRobot scripts on the Raspberry, a demo (DisplayLezen) can be run as following.

- Open VNC to work from the Raspberry pi
- Open the terminal, enter the following commands:

```
$ source ~/.profile          # always update file for virtual environment when␣
↪opening terminal
$ workon rr                  # always work in the virtual environment when running a␣
↪script on the raspberryPi
$ cd ~/pathways-extensions-training/RekenRobot/Src/DisplayLezen/Demo          #␣
↪naviagate to the right directory
$ python 'namefile'.py       # run the script of the demo
```

1.1.3 Python docstrings (Training Howto)

Docstrings in je python code kun je semi-automatisch omzetten naar API documentatie. Hoe je dit doet vind je hier.

Python Docstrings builden

Met de docstrings in je python-code kun je semi-automatisch een API documentatie builden. Voor docstrings gebruiken wij de Google stijl conventie. Klik hier voor een voorbeeld.

Om de Google stijl docstrings te kunnen parsen gebruiken we Napoleon, een Sphinx extensie. Deze download je als volgt in CMD:

```
pip install sphinxcontrib-napoleon
```

In je std_conf.py bestand hoor je de Napoleon en Autodoc extensies te appenden als volgt (als het goed is, is dit al gedaan):

```
extensions.append('sphinx.ext.autodoc')
extensions.append('sphinxcontrib.napoleon')
```

Je kunt nu in de command line je API documentatie builden. Side note: voor deze stap moet er bij elke module die je wilt builden een (lege) __init__.py zitten.

```
sphinx-apidoc -f -o destination_folder src_code_path
```

De destination_folder is de folder waarin de gebouwde .rst bestanden komen te staan. De src_code_path is het pad naar de folder van je source code. Diezelfde src_code_path moet je in je std_conf.py zetten zodat het je code kan vinden (als het goed is, is dit ook al gedaan):

```
import sys
sys.path.insert(1, src_code_path)
```

Bij de volgende stap zullen de docstrings daadwerkelijk uit de code geextraheerd worden. Dit doe je door je documentatie als volgt te builden:

```
sphinx-build -b html source_folder destination_folder
```

De source_folder is de folder waar de conf.py van sphinx staat (niet hetzelfde als de std_conf.py die in een andere folder zit!). De destination_folder is de folder waarin je je .html bestanden wilt plaatsen.

See also:

Een voorbeeld van hoe de API documentatie er uiteindelijk ziet, kun je vinden in:

- *RekenRobot API doc*

CHAPTER 2

Part II: The extensions (under development)

This is about the extension(s) and it parts. Each extensions has requirements, components, a design, lots of code, etc. This comes with several kinds of documentation.

Some is written (or given by) the trainer (in his role as *product-owner*). Some is written by the teams. Most focus in the product, and how to use it. See the *SUMs* (Software User Manuals).

2.1 Extensions

2.1.1 RekenRobot

Opdracht

Algemeen

Attention: This part is coming from the *BureauLade* section of the Pathways documentation. See there for more context.

Achtergrond

Pathways is zowel een concept, als een referentie implementatie. Ook zijn een aantal eenvoudige voorbeelden toegevoegd; bedoelt om mee te kunnen *spelen*. Het standaard voorbeeld is een rekenmachine. Iedereen snapt de werking hiervan; de (voorbeeld) testen zijn dus begrijpbaar, en geschikt als voorbeeld van pathways zelf.

Er zijn meerdere rekenmachines; zoals een (stand-alone) webapp en een xmlrpc-server. Al zijn de testen (deels) gelijk, de koppeling tussen het pathways-framework en het product-onder-test (*PUT*) zijn natuurlijk anders. Daarmee wordt de (ont)koppeling –in pathways– verduidelijkt. En het is een voorbeeld van hoe met deze *gate* kan maken make naar een *PUT*

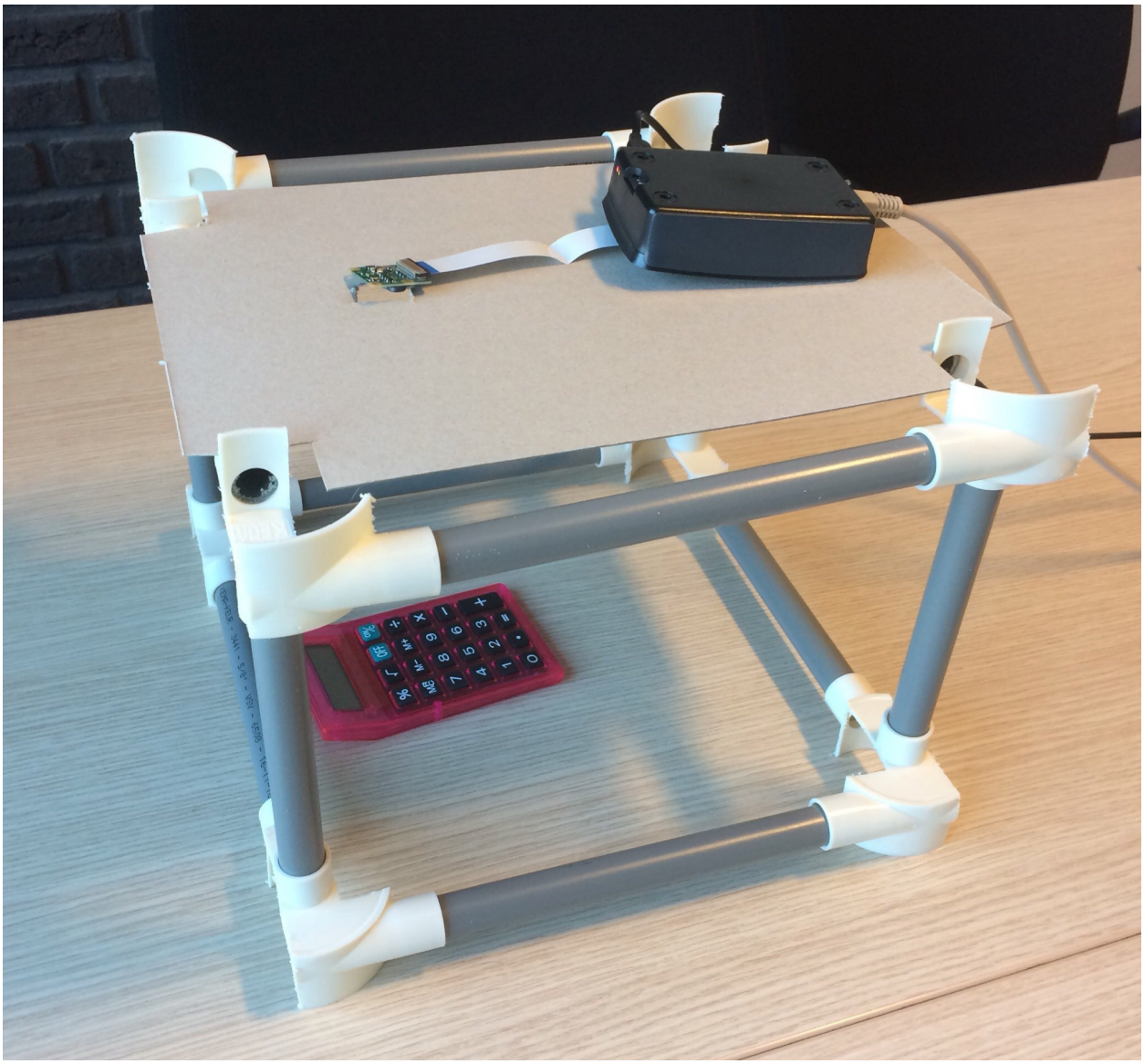


Fig. 1: The first prototype
A RaspberryPi camera-module looking-down (there is a hole in the cardboard) to the 'RekenMachine'.

In deze training gaan we een extra *PUT* aansluiten: een echte, *fysieke* rekenmachine. De *gate* moet dus de knoppen bedienen en het display lezen (met OCR!).

Doel: Een rekenmachine bedienen

Een *echte* rekenmachines heeft (electro-mechanische) knoppen en vaak een lcd-display (soms een 7-segment led versie) Om zo'n rekenmachines te testen moeten die knoppen bedient worden. En het display worden gelezen! Dit moet mechanische en/of via elektrische signalen en is dus heel anders dan de software-interface bij de eerdere voorbeelden. (Al zal dat voor de testen, de *ATS*en niets uitmaken.)

Hiervoor zijn een aantal onderdelen nodig:

- Een *robots* om de knoppen te bedienen en het display te lezen.
- Een *software-lib* om die robot te aan te sturen; vanuit pathways gezien is dat de *gate*.
- Een aantal *testen* (*ATS*en) en/of *brick*s voor deze rekenmachine(s). Hierbij kunnen bestaande (shared) testen en bricks hergebruikt worden.

Het *robot*-onderdeel is het nieuwe en meest complexe (qua ontwikkeling) deel van de opdracht. Dat krijgt hieronder de meeste aandacht. Men mag echter niet vergeten dat het pas nuttig is als he geheel werkt!

Deze 'robot' bestaat weer uit (minimaal) twee hoofd delen: het *DisplayLezen* en de *KnoppenDrukker*. Beide bestaan weer uit diverse onderdelen.

Opties

Er zijn meerdere mogelijkheden om deze robot te realiseren. Het uitzoeken wat 'het beste is', is onderdeel van de opdracht.

Het lijkt verstandig om klein te beginnen. Dus voor één eenvoudige zakjapanner, die eenvoudig te koppelen is. Bij voorkeur moet de robot (later, eenvoudig) uit te breiden zijn voor andere/meerdere rekenmachines. Ook moet de gekozen rekenmachine alom beschikbaar zijn. (Denk aan de "speel"-voorbeeld doelstelling).

Het robotje moet (minimaal) het volgende kunnen:

1. De knoppen (0 .. 9, +,-,*,÷ en '=' (*uitkomst*, *enter*, . . .) **bedienen**. Dit mag mechanisch of door de "onderligende schakelaars" elektrische te stimuleren. Mogelijk zijn er ook andere opties.
2. Het (lcd/led) display **lezen** (op commando). Waarbij de waarde als tekst-string (of getal) beschikbaar moet komen. Dit is mogelijk het lastigste om goedkoop en eenvoudig te realiseren.

Natuurlijk mogen bovenstaande functies ondersteund worden door (python) software. Deels is deze al beschikbaar; bijvoorbeeld om een getal op te breken in aparte cijfers.

Het 'rekenen' **moet** uiteraard op de rekenmachine uitgevoerd worden. Ook verder moet de rekenmachine zoveel mogelijk *'natuurlijk'* gebruikt worden; we willen immers die rekenmachine testen. Daarom heeft 'indrukken' van de knoppen ook de voorkeurover elektrische aansluitingen. Anderzijds, kan deze "elektrische simulatie" de opdracht veel eenvoudiger maken; zeker voor het lezen van het display.

Omgeving

De software moet gaan werken op de raspberry-Pi (dus onder Linux). Maar het is handig (verplicht) om het ook te laten werken op een standaard PC (en de Mac); waarop ontwikkeld wordt.

Tip: Hou 'linux' en 'embedded' in gedachte bij elke keuze.

Dat *'t werkt op jou PC* is niet de opdracht; het moet werken op de Raspberry!

- De 'zero' uitvoering kost slechts 5 euro. En is daarom de voorkeur van de opdracht gever

  Tip: Kosten

  De 'zero' heeft geen netwerk aansluiting. De duurdere 'B' wel en is wellicht handiger.

  - Maakt dat iets uit? Voor wie?
  - Kun je de opdrachtgever hiervan overtuigen?
  - Als je over kosten nadenkt, denk dan ook aan ontwikkel-kosten en aantallen!

Tip: Meer tips

- De mogelijkheden om de rekenmachine (elektrisch) te koppelen zijn sterk afhankelijk van de kosten van die calculator. Kost die ook 5 euro, dan zal solderen een optie zijn. Maar bedenkt dat, ook dan veel potentielle gebruikers hier tegen kunnen opzien. Alles dat complexer is dan een schroevendraaier kan de speel-doelstelling te moeilijk maken.
- Wellicht kan het lezen van het display gebeuren met een webcam; zoals ingebouwd in veel laptops of telefoons. Voordeel: goedkoop beschikbaar en laagdrempelig. Nadeel: meer & complexere software nodig (beeldverwerking)
- Een andere, soortgelijke optie is de raspberry-camera (€20,=). Dit moet verder uitgezocht worden.
- De software moet een ruime reeks van webcams' (kunnen) ondersteunen.
- Voor het benodigde stukje computer-visie **zal** openCV en/of Tesseract gebruikt worden; tenzij er een goede reden is om hiervan af te wijken en dit besproken (en goedgekeurd) is met/door de coach.

DisplayLezen

author albert

In deze (deel)opdracht wordt de module *'het (uit)lezen van het display van een zakrekenmachine'* beschreven.

Doel

See also:

Zie ook *Project Kaders*

Globaal

- Lezen van het (7-segment, LCD) Display.
- Te starten met 1 stuk type (naar keuze)
- Uiteindelijk zullen 'alle' rekenmachientjes ondersteund worden.

Uitgangspunt is dat er één (display) regel –op commando— gelezen **moet** worden en als text en/of getal (float/int) teruggegeven wordt. Ook **moeten** errorcode's, etc terug gegeven worden. Er **mag geen** interpretatie plaatsvinden (behalve het omzetten van een plaatje naar test/getallen); dat gebeurd extern (in de aanroeper). Ofwel: alles was op het rekenmachine display staat moet (begrijpbaar) opgeleverd worden.

Niet-Functionele Eisen

Het lezen **moet** zeer betrouwbaar en snel gebeuren; zonder menselijk interventie of supervisie.

- Bij voorkeur duurt het lezen niet langer dan 50ms. Het **moet** altijd binnen 0.2sec gebeuren. Het systeem **moet** continue kunnen werken. Dus 5 à 20 lees-commando's per seconde. En dat urenlang!
- Een eenmalige calibratie van de opstelling (ook fysiek) is toegestaan; dit **moet** eenvoudig en snel kunnen, aan de hand van een duidelijke gebruikershandleiding. Daarna **zal** menselijk handelen niet meer nodig zijn.
- Het lezen van de correcte getallen is belangrijk. Een eis van maximaal 1 fout opgeleverd per honderdduizend lees-commando's is zeer wenselijk. Optioneel mag een 'kan-niet-lezen' melding gebruikt worden (dat liever dan een fout getal). Dat mag echter niet vaker dan 1 op de duidend keer gebeuren; als er een getal op het display staat.
- Ook bij niet-getallen, als die op het display staan mag eens in de duizend keer onjuiste opgeleverd worden. Maar dit mag nooit een getal opleveren; lees: dat telt mee voor de max 1-op-100000 fouten eis.

Opleveren

1. Een 'library' die alle benodigde functionaliteit bevat. Tezamen met een gedocumenteerde API hoe deze gebruikt **zal** worden. Het 'lees-commando' is hiervan het belangrijkste onderdeel.
2. Een eenvoudig, standalone, *"command-line"* programma, dat die library gebruikt en als 'demo' dient (voor HighTech-type medewerkers). Ook dit **zal** gedocumenteerd worden
3. Alles (programmatuur, documentatie, etc) **zal** zowel als source als kant en klaar opgeleverd worden. Ook **moet** het 'bouw' process eenvoudig en gedocumenteerd zijn.
4. Project documentatie; zodat anderen gemakkelijk dit deel kunnen gebruiken en/of verder kunnen ontwikkelen.
5. Ook alle test, zowel op unit- als systeem-niveau (en alles daartussen). Liefst als Automatische- Test-Scripts. En anders als handmatige test (ontwerpen).
6. Optioneel: tools die gebruikt zijn cq handig zijn tijdens (door)ontwikkelen.

KnoppenDrukker

author albert

In deze (deel)opdracht wordt de module *'het bedienen van de (fysieke) knoppen van een zakrekenmachine'* beschreven.

Doel

See also:

Zie ook *Algemeen*

Globaal

Todo: Later in te vullen

Provisionally design (of RekenRobot)

Warning: Currently, *RtFD* does not support plantUML drawings.

This implies you can't see the UML diagram below. For now, view it locally!

Apidoc

RekenRobot API doc

See also:

- *Python docstrings (Training Howto)*

Demo package

Submodules

Demo.MainfileReadimage module

Module contents

FlexPreprocessInitializer package

Submodules

FlexPreprocessInitializer.InitializeFlexPreprocessor module

This module is responsible for determining the selection and order for preprocessing the image.

class FlexPreprocessInitializer.InitializeFlexPreprocessor.**InitializeFlexPreprocessor**
Bases: object

This class allows the user to select preprocessors to be used.

The user can determine the selection, as well as the order in which preprocessors are used. The user input is checked against possible conditions for validity.

static create_preprocessor_sequence(*pp_dict*, *default_pp*)
Determines the preprocessor selection and sequence based on user input.

Outputs warnings via print statements if the user input is invalid.

Todo:

- re.match("^[1-9,]*$", pp_sequence) is line 48 allows all number above 1!

Parameters

- **pp_dict** – Dictionary containing the available preprocessors and their corresponding keys.
- **default_pp** – A list containing the default selection and order of preprocessors.

Returns List of selected preprocessors in the requested order.

static print_instructions()
Prints the instructions for the user.

init_preprocessor_sequence()
Initializes the class and list/dictionary variables which store the default preprocessors.

Returns The list of selected preprocessors, in the order they are chosen.

Module contents

ImageCropper package

Submodules

ImageCropper.AutoDetect module

This module is responsible for detecting a calculator display from a camera feed.

class ImageCropper.AutoDetect.**AutoDetector**
Bases: object

This class detects a calculator display from a camera feed automatically.

The user only supplies the class with an image and the class detects a frame. For proper operation it is recommended to have a clear view of the display with a blank background.

static detect(*image*)
Detects a frame from an image

The detect method performs the following actions to detect a display from an image.

- **pyrMeanShiftFiltering - This filters the image so that it can be** segmented.
- **cvtColor YCrCb and HLD - This converts the filtered image to YCbCr** and HLS color spaces.
- split - Splits the color channels.
- **equalizeHist - Normalizes the brightness and increases the contrast** of the split images.
- merge - Merges the split images and inverts them.
- Threshold - Thresholds the image.
- morphologyEx - Opens the image (erosion followed by dilation)
- findContours - Finds contours in a binary image.
- **Box - The found contours are used to determine the rectangle** coordinates for cropping.

Coordinates are returned in the following form: [[(rec1_x1, rec1+y1),(rec1_x2, rec1+y2),(rec1_x3, rec1+y3),(rec1_x4, rec1+y4)], [(rec2_x1, rec2+y1),(rec2_x2, rec2+y2),(rec2_x3, rec2+y3),(rec2_x4, rec2+y4)], [etc. . .]]

Parameters **– RGB image in which a display can be detected.** (*image*) –

Returns Array of coordinates describing rectangles.

..todo:: Make sure that the found contour is not outside the image. This can cause errors in the cropping (which are now worked around).

ImageCropper.Crop module

This module is responsible for cropping an image based on coordinates.

class ImageCropper.Crop.**Cropper**

Bases: object

This class allows users to crop an image.

The user supplies the class with an image and coordinates and the class will return a cropped image.

static crop_image(*image*, *cords*)

Crops the image based on supplied coordinates.

This method extracts the full image from the coordinates and the uses the coordinates to calculate the angle, determine the orientation and warp the image accordingly.

It does this through the following actions:

- It extracts crops the image to include every coordinate.
- It determines the bottom side of the display.
- It calculates the angle based on the orientation of the rectangle.
- It rotates the rectangle using warpAffine.

Coordinates are given in the following form:

Parameters

- **– RGB image as source to cut display out of** (*image*) –
- **– Array of coordinates describing rectangles** (*cords*) –

Returns RGB image of the display only (cropped out of the original image and rotated for a horizontal image) If the cords are outside the image, a string ('failed to crop') is returned.

ImageCropper.FixedDetect module

This module is responsible for providing the crop module with coordinates based on user-input regarding the location of a calculator display.

class ImageCropper.FixedDetect.**FixedDetector**

Bases: object

This class provide coordinates for the cropper based on an image and the user-provided location info.

static get_cords(*image*, *x*, *y*, *w*, *h*, *a*)

Converts user location input into coordinates for the crop module.

The user provides the location, size and angle of the display within the image.

The rectangle is constructed using basic trigonometry. The rectangle is then used to construct the coordinates for the crop module.

This is accomplished by drawing 4 lines based on the coordinates and the angle on a white copy of the image and then using findContours to get the coordinates.

Coordinates are returned in the following form: [[(rec1_x1, rec1+y1),(rec1_x2, rec1+y2),(rec1_x3, rec1+y3),(rec1_x4, rec1+y4)], [(rec2_x1, rec2+y1),(rec2_x2, rec2+y2),(rec2_x3, rec2+y3),(rec2_x4, rec2+y4)], [etc...]]

Parameters

- **– RGB image in which a display can be detected.** (*image*) –
- **– Location of display on x-axis in pixels.** (*x*) –
- **– Location of display on y-axis in pixels.** (*y*) –
- **– Width of display in pixels.** (*w*) –
- **– Height of display in pixels.** (*h*) –
- **– Angle of display in degrees.** (*a*) –

Returns Array of coordinates describing rectangles.

ImageCropper.ManualDetect module

The ManualDetect module allows the user to select a region of interest to crop out of an image.

class ImageCropper.ManualDetect.**ManualDetector**

Bases: object

This class lets the user select three points on the supplied image in order to get the four coordinates needed to make a rectangle.

click_and_detect(*event*, *x*, *y*, **flags*, ***param*)

Listens for mouse click events.

Records the (x,y) coordinates when the left mouse button is clicked. Lines are drawn between selected points.

Args: event - The keyboard or mouse event that took place. x - The x coordinate of said event. y - The y coordinate of said event. flags - Relevant flags that are passed by OpenCV. param - Extra parameters supplied by OpenCV.

manual_detect()

Makes an array of the coordinates that come from the click_and_detect function.

manual_detect_setup(*img*)

Uses the click_and_crop function to crop an image.

The 'r' and 'c' keys reset and crop the image respectively. Furthermore, the cropped image is shown in a separate window.

Args: img - RGB image in which a display can be detected.

Returns: Array of coordinates describing rectangles.

static init_manual_detect(*img*)

Initializes the manual detector.

Parameters – RGB image in which a display can be detected. (*img*) –

Returns Array of coordinates describing rectangles.

Module contents

Preprocessor package

Submodules

Preprocessor.PreprocessImages module

This module is responsible for preprocessing images to make them more suitable for Tesseract-OCR.

class Preprocessor.PreprocessImages.**Preprocess**
Bases: object

This class preprocesses the image by applying filters and manipulations.

static gray_scale(*image*)
Applies grayscale on the image.

Parameters **image** – The image to be preprocessed.

Returns Grayscaled image.

static gaussian_blur(*image*)
Applies gaussian blur on the image.

Parameters **image** – The image to be preprocessed.

Returns Gaussian blurred image.

static filter2d(*image*)
Applies 2D convolution on the image.

Parameters **image** – The image to be preprocessed.

Returns 2D Convoluted image.

static binary_threshold(*image*)
Applies a binary threshold combined with Otsu's binarization on the image.

Parameters **image** – The image to be preprocessed.

Returns Binary thresholded image (i.e. black and white).

static binary_adaptive_threshold(*image*)
Applies a adaptive binary threshold.

Parameters **image** – The image to be preprocessed.

Returns Binary thresholded image (i.e. black and white).

static normalisation(*image*)
Applies a histogram based normalisation to image to increase contrast.

Parameters **image** – The image to be preprocessed.

Returns Normalisated image

static white_boarder(*image*)
Applies a white boarder to image the size of the hight of the given image

Parameters **image** – The image to be preprocessed.

Returns Image with white boarder

static median_blur(*image*)
Applies a median blur on the image.

Parameters **image** – The image to be preprocessed.

Returns Median blurred image.

static erosion(*image*)
Applies erosion on the image.

Parameters **image** – The image to be preprocessed.

Returns Eroded image.

static perspective_transform(*image*)
Corrects for deformations created by the camera.

Parameters **image** – the (color) image to be preprocessed.

Returns Corrected image

static preprocess_image(*pp_sequence*, *image*)
This function runs the preprocessing based on a supplied selection and sequence.

The getattr() function is used to call a function based on a String value. For example, the string 'erosion' can be used to call the method erosion() in this module.

Parameters

- **pp_sequence** – The selection and sequence of the preprocessor operations
- **image** – The image to be preprocessed.

Returns Preprocessed image.

Module contents

ReadDisplay package

Submodules

ReadDisplay.ReadImage module

This module is responsible for reading numbers off an image.

class ReadDisplay.ReadImage.**ReadImage**
Bases: object

This class opens an image, converts it to a string and prints it.

static readimage(*image*)
This function reads a number off of an image.

The image is converted from numpyarray to image, and then from image to string. The image_to_string() function is adjusted slightly for one digit numbers.

Parameters **image** – Input image from which the number will be read.

Returns A string containing the number that is read off of the image.

Module contents

TestEvaluateOcr package

Submodules

TestEvaluateOcr.ImageGenerator module

The ImageGenerator takes the images from "_font_numbers". Pastes them together to create a random number. If desired do some modifications to make them more 'real'. The numbers are saved in the "_testImages" folder. The filename of the created images describes the rendered number.

`TestEvaluateOcr.ImageGenerator.`**`main`**`()`

The main() function checks the available options. These options are presented to the user to pick from. The user can select his/her preference of background and font. The main() function will then call Digits() to generate the images. Once completed it will notify the user that the script is done.

Todo:

- **Create the option to generate:**
  - images of all backgrounds in one call
  - all digit ranges in one call
  - images of all possible fonts in one call
  - all possibilities in one call
- Refactor code in sub functions

`TestEvaluateOcr.ImageGenerator.`**`Digits`**(*digit_num*, *digit_type*, *amount*, *font*, *background*, *process*)

The Digits() function creates a set of images It places those images in _testImages, which is ignored by Mercurial It creates a folder per image set that is based on font and background

It also checks if the amount given is more than the available options If it's more than is possible to generate The script will create each possibility effectively Otherwise it will generate random numbers The images are saved immediately in the correct folder

---

Todo:

- Print complete numbers on 1 background, instead of combining individual numbers + backgrounds.
- Ensure a 0 can't be placed in front of any other number
- Create a folder for processed and non-processed images

Parameters

- **`digit_num`** – The length of numbers (e.g. 3 digits == 123)
- **`digit_type`** – Which digits are included
- **`amount`** – The amount of images to be created
- **`font`** – The path of the fontNumbers created by DigNumberGenerator

- **background** – The background image that is selected
- **process** – To evaluate if the image needs to be processed

TestEvaluateOcr.ImageGenerator.**adaptImage**(*image*)

Adapts a 'good' image to a 'real' image, by blurring, adding noise, and deforming the image (i.e., the distance between the two upper corners is smaller than the distance between the two lower corners, which is the case by the images obtained with the web cam).

Todo:

- Make different steps more flexible.

Parameters **image** – The image to be adapted.

Returns An image that is similar to a photo.

Return type result_image

TestEvaluateOcr.DigNumberGenerator module

The DigNumberGenerator generates images of the numbers 0 - 9. The user can select the desired font and background. The images are placed in "_font_numbers" within the current directory. Under the selected font and background.

TestEvaluateOcr.DigNumberGenerator.**main**()

The main() function checks the possible backgrounds and fonts. It then presents these options to the user so he/she can select them. If the font is installed and the input is valid it will start. Once the generation is comlete it will close immediately.

Todo:

- Allow the fonts to be used without being installed.

TestEvaluateOcr.test_evaluate_non_pp_ocr module

The test is responsible for evaluating the accuracy of Tesseract.

Before running this test: Test images can be generated using DigNumberGenerator and ImageGenerator. (see __README__.txt)

This script iterates through all the generated images. Feeds these images to PreprocessImages.py and ReadImage.py In order to evaluate the performance of the latter. As well as the effect that the PreprocessImages module has on the OCR. If the ReadImage module achieves a higher accuracy on preprocessed images. The PreprocessImages module has a positive effect on the accuracy.

The test images stored in the folder should be named accordingly. If the image depicts the number '1238', the file should be named '1238.png' The extension is irrelevant as long as OpenCV can read the image (This will always be the case if the images are generated)

TestEvaluateOcr.test_evaluate_non_pp_ocr.**test_evaluate_non_pp_ocr**()

TestEvaluateOcr.test_evaluate_non_pp_ocr.**digitTest**(*directory*)

TestEvaluateOcr.test_evaluate_pp_ocr module

The test is responsible for evaluating the accuracy of Tesseract.

Before running this test: Test images can be generated using DigNumberGenerator and ImageGenerator. (see __README__.txt)

This script iterates through all the generated images. Feeds these images to PreprocessImages.py and ReadImage.py In order to evaluate the performance of the latter. As well as the effect that the PreprocessImages module has on the OCR. If the ReadImage module achieves a higher accuracy on preprocessed images. The PreprocessImages module has a positive effect on the accuracy.

The test images stored in the folder should be named accordingly. If the image depicts the number '1238', the file should be named '1238.png' The extension is irrelevant as long as OpenCV can read the image (This will always be the case if the images are generated)

The raw input image and preprocessed image are visualized using imshow() These visualizations are purely there to provide insight. The visualizations can be removed to improve the performance of the script.

The waitKey() function of OpenCV can be set to 1 to auto-iterate all images By setting it to 0 the user can control the iterations. On key-press(e.g. enter), the script will move to the next image.

`TestEvaluateOcr.test_evaluate_pp_ocr.`**`test_evaluate_pp_ocr`**`()`

`TestEvaluateOcr.test_evaluate_pp_ocr.`**`digitTest`**`(`*directory*`)`

TestEvaluateOcr.test_improve_pp module

This test will test whether the improved preprocessing increase the number of correct estimations of the pp+ocr by 10%.

This module is a script that iterates through all images in the specified folder, feeds these images through PreprocessImages.py and ReadImage.py in order to evaluate the performance of the latter, as well as the effect that the PreprocessImages module has on ReadImage. This will be done twice: first for the default preprocessing, then for the increased preprocessing.

The test succeeds if the number of correct readings with the improved preprocessor is equal or more than 2x the number of the correct readings with the default preprocessor.

The test images stored in the folder should be named according to the number depicted in it. For example, if the image depicts the number '1238', the file should be named '1238.jpg' or '1238.png'. The extension is irrelevant as long as OpenCV can read the image using it's imread() function.

`TestEvaluateOcr.test_improve_pp.`**`test_improve_pp`**`()`

`TestEvaluateOcr.test_improve_pp.`**`digitTest`**`(`*directory*`)`

TestImageCropper package

Submodules

TestImageCropper.test_AutoDetect module

This module is responsible for testing the AutoDetect module.

`TestImageCropper.test_AutoDetect.`**`test_01`**`()`

This test feeds AutoDetect a test image and compares the result to the correct coordinates.

The test image contains a rotated calculator. The expected coordinates represent the screen. The difference between the actual en returned coordinates are compared with a tolerance, so everyting within that margin is accepted.

TestImageCropper.test_Crop module

This module is responsible for testing the Crop module.

TestImageCropper.test_Crop.**assert_crop_straight**(*straight_image*)

TestImageCropper.test_Crop.**assert_crop_rotate**(*rotate_image*)

TestImageCropper.test_Crop.**test_crop_straight**()

TestImageCropper.test_Crop.**test_crop_45angle**()

TestImageCropper.test_Crop.**test_crop_vertical1**()

TestImageCropper.test_Crop.**test_crop_vertical2**()

TestImageCropper.test_Crop.**test_crop_wrong_contour1**()

TestImageCropper.test_Crop.**test_crop_wrong_contour2**()

TestImageCropper.test_FixedDetect module

TestImageCropper.test_FixedDetect.**assert_zero**(*cords*)

TestImageCropper.test_FixedDetect.**assert_twenty**(*cords*)

TestImageCropper.test_FixedDetect.**assert_forty_five**(*cords*)

TestImageCropper.test_FixedDetect.**test_zero**()

TestImageCropper.test_FixedDetect.**test_forty_five**()

TestImageCropper.test_FixedDetect.**test_twenty**()

TestImageCropper.test_ManualDetect module

This module is responsible for testing the ManualDetect module.

TestImageCropper.test_ManualDetect.**assert_straight_rectangle**(*straight_cords*)
Asserts coordinates of a straight selection

TestImageCropper.test_ManualDetect.**assert_rotated_rectangle**(*rotated_cords*)
Asserts coordinates of a rotated selection

TestImageCropper.test_ManualDetect.**assert_skewed_rectangle**(*skewed_cords*)
Asserts coordinates of a skewed selection

TestImageCropper.test_ManualDetect.**assert_max_input**(*nr_user_input*)
Asserts userinput more than 3 inputs

TestImageCropper.test_ManualDetect.**assert_min_input**(*nr_user_input*)
Asserts userinput less than 3 inputs

TestImageCropper.test_ManualDetect.**assert_misfit**(*misfit_error_code*)
Asserts when userinput causes the cropped rectangle to be out of the area of image.

TestImageCropper.test_ManualDetect.**test_horizontal**()
Tests the ManualDetect with a straight rectangle.

A new ManualDetector object is created. The four mouse-clicks from the user are then imitated and coordinates are retrieved.

TestImageCropper.test_ManualDetect.**test_rotated**()
Tests the ManualDetect with a 45 degrees rotated rectangle.

A new ManualDetector object is created. The four mouse-clicks from the user are then imitated and coordinates are retrieved.

TestImageCropper.test_ManualDetect.**test_vertical**()
Tests the ManualDetect with a irregular rectangle.

When the manualdetect gets an irregular rectangle, it is supposed to transform it into a straight rectangle.

A new ManualDetector object is created. The four mouse-clicks from the user are then imitated and coordinates are retrieved.

TestImageCropper.test_ManualDetect.**test_max_input**()
Tests if the ManualDetect can process a maximum of 3 userinputs. This involves the click_and_detect function. The possibilty of clicking once or twice is not taken into account in this test. This test can/should? be added for completeness.

TestImageCropper.test_ManualDetect.**test_min_input**()
Tests if the ManualDetect can process less than 3 userinputs. This involves the manual_detect function.

TestImageCropper.test_ManualDetect.**test_misfit**()
Tests what happends when a drawn rectangle has a point outside the area of the image to process.

TestImageCropper.test_FixedDetectCompletelyOutOfBorder module

TestImageCropper.test_FixedDetectNegativeValues module

Module contents

TestInitializeFlexPreprocessor package

Submodules

TestInitializeFlexPreprocessor.test_InitializeFlexPreprocessor module

This module is responsible for testing the InitializeFlexPreprocessor.py module

The dictionary containing the correct key & preprocessor pair is initialized here as pp_dict. The list containing the default selection and sequence of preprocessors is initialized here as default_pp.

TestInitializeFlexPreprocessor.test_InitializeFlexPreprocessor.**test_init_flex_pp_1**()
Tests a valid user input (only gray scale).

This test asserts that the InitializeFlexPreprocessor module returns the correct preprocessor (i.e. gray scale) based on a user input of '1'. The values 1-6 are keys for corresponding preprocessors as defined in the dictionary pp_dict.

TestInitializeFlexPreprocessor.test_InitializeFlexPreprocessor.**test_init_flex_pp_4_1**()
Tests an invalid user input (binary threshold before gray scale).

This test handles two user inputs: an invalid one ('4,2,1') followed by a valid input ('1,2,4'). The first input is invalid because the input image for the binary threshold function should be gray scale. In other words, the image has to pass through the gray scale function ('1') before passing the binary threshold module ('4').

The first assertion checks if the InitializeFlexPreprocessor module prints the correct warning to stdout when an invalid input is used. The second assertion checks whether the module returns the correct list containing the selection and sequence of preprocessors following the correct user input.

TestInitializeFlexPreprocessor.test_InitializeFlexPreprocessor.**test_init_flex_pp_4**()
Tests an invalid user input (choosing only binary threshold without gray scale).

This test handles an invalid and a valid input. The first assertion checks whether the InitializeFlexPreprocessor module prints the correct warning message to stdout when an invalid input is used (i.e. only choosing binary threshold without gray scale). The second assertion checks whether the correct list of preprocessors is returned by the module based on the following correct input.

TestInitializeFlexPreprocessor.test_InitializeFlexPreprocessor.**test_init_flex_pp_a0**()
Tests an invalid user input (invalid input characters).

This test handles three user inputs. The first two contain invalid input characters. The third one is valid. The first two inputs contain invalid characters (i.e. 'a' and '9' respectively). The test asserts whether the InitializeFlexPreprocessor module prints the correct warning message to stdout. The second assertion checks whether the module returns a correct list of preprocessors based on valid user input.

TestInitializeFlexPreprocessor.test_InitializeFlexPreprocessor.**test_init_flex_pp_55**()
Tests an invalid user input (choosing duplicate preprocessors).

This test handles an incorrect user input containing a duplicate ('1,2,5,5'), as well as a correct input. The test asserts whether the InitializeFlexPreprocessor module prints the correct warning message to stdout. The second assertion checks whether the correct list of preprocessors is returned based on valid user input.

TestInitializeFlexPreprocessor.test_InitializeFlexPreprocessor.**test_init_flex_pp_default**()
Tests the default selection and sequence of preprocessors as defined in the list default_pp.

The test handles a single user input. It simulates the "Enter" key being pressed. Hence, the input is (''). This triggers the default setting of preprocessors in InitializeFlexPreprocessor. The assertion in this test checks whether the module returns the correct list of preprocessors.

Module contents

TestPreprocessor package

Submodules

TestPreprocessor.test_Preprocessor module

This module is responsible for testing the PreprocessImages.py module.

There are two different types of tests: firstly, whether the PreprocessImages module (defined as Pp in this test) returns an image that has the same dimensions (i.e. shape), and type (i.e. numpy.ndarray) as the input image (i.e. expected image). The expected image and it's attributes are defined below.

Secondly, it is tested whether error messages are returned if the input image is in color, while gray scale images are required by that preprocessor (i.e. binary threshold and normalization).

The actual content of the output image has to be checked manually. Alternatively the EvaluateReadDisplay.py module can be run to check whether the ReadDisplay module achieves a higher accuracy as a result of the new preprocessing output of Pp. The reason why the image content (i.e. pixels) of the Pp output are not checked (i.e.

pixels_of_expected_image == pixels_of_output_image) is because even the slightest change in parameters in Pp can change the pixels of an image drastically, and it does not necessarily mean that the module does not work anymore. Additionally, a slight change in pixels can have a big impact on the accuracy of ReadDisplay. Therefore, it's best to check the quality of the Pp output images by eye as well as through EvaluateReadDisplay.

TestPreprocessor.test_Preprocessor.**assert_img_attributes**(*output_img*)
This method asserts whether the image attributes of the output image of Pp match those of the expected image.

The two image attributes that are checked are (1) the image type, and (2) the image shape.

Parameters **output_img** – The output image returned by Pp.

TestPreprocessor.test_Preprocessor.**assert_img_attributes_white_boarder**(*output_img*)
This method asserts whether the image attributes of the output image of Pp match those of the expected image. Here the expected image is larger that the original image.

The two image attributes that are checked are (1) the image type, and (2) the image shape.

Parameters **output_img** – The output image returned by Pp.

TestPreprocessor.test_Preprocessor.**assert_binary_thresh_error**(*direct_method_call*, ***kwargs*)
This method asserts whether the Pp module returns the correct warning message when triggered.

The warning message is printed to stdout when an image is passed to the binary_threshold() function in Pp that is not gray scale. The method can be called directly (via 'binary_threshold(img)' or indirectly (via 'preprocess_image(list, img)').

If the binary threshold method is called indirectly, an extra input parameter is required which is handled by **kwargs in this function.

Parameters

- **direct_method_call** – True if binary threshold method is called directly,
- **if called indirectly.** (*false*) –
- **kwargs** – A list ('pp_list') containing a selection and sequence of
- **preprocessors.** –

TestPreprocessor.test_Preprocessor.**assert_binary_adap_thresh_error**(*direct_method_call*, ***kwargs*)
This method asserts whether the Pp module returns the correct warning message when triggered.

The warning message is printed to stdout when an image is passed to the binary_threshold() function in Pp that is not gray scale. The method can be called directly (via 'binary_threshold(img)' or indirectly (via 'preprocess_image(list, img)').

If the binary threshold method is called indirectly, an extra input parameter is required which is handled by **kwargs in this function.

Parameters

- **direct_method_call** – True if binary threshold method is called directly,
- **if called indirectly.** (*false*) –
- **kwargs** – A list ('pp_list') containing a selection and sequence of
- **preprocessors.** –

TestPreprocessor.test_Preprocessor.**assert_normalisation_error**(*direct_method_call*, ***kwargs*)
This method asserts whether the Pp module returns the correct warning message when triggered.

The warning message is printed to stdout when an image is passed to the binary_threshold() function in Pp that is not gray scale. The method can be called directly (via 'binary_threshold(img)' or indirectly (via 'preprocess_image(list, img)').

If the binary threshold method is called indirectly, an extra input parameter is required which is handled by **kwargs in this function.

Parameters

- **direct_method_call** – True if binary threshold method is called directly,
- **if called indirectly.** (*false*) –
- **kwargs** – A list ('pp_list') containing a selection and sequence of
- **preprocessors.** –

TestPreprocessor.test_Preprocessor.**assert_binary_thresh_and_normalisation_error**(*direct_method_*
***kwargs*)

This method asserts whether the Pp module returns the correct warning message when triggered.

The warning message is printed to stdout when an image is passed to the binary_threshold() function in Pp that is not gray scale. The method can be called directly (via 'binary_threshold(img)' or indirectly (via 'preprocess_image(list, img)').

If the binary threshold method is called indirectly, an extra input parameter is required which is handled by **kwargs in this function.

Parameters

- **direct_method_call** – True if binary threshold method is called directly,
- **if called indirectly.** (*false*) –
- **kwargs** – A list ('pp_list') containing a selection and sequence of
- **preprocessors.** –

TestPreprocessor.test_Preprocessor.**test_individual_pp**()

This test checks if each individual preprocessor method (direct call) returns the correct output.

The getattr() function is used to call a function based on a String value. For example, the string 'erosion' can be used to call the method erosion() in the Pp module.

The test asserts whether the output image is valid for each of the individual preprocessor methods. The binary_threshold and normalisation methods are checked with a gray scale image, the perspective_transform for both color and grey scale image, and the white_border method is checked individually as the output of this function is expected to have a larger size than the input.

TestPreprocessor.test_Preprocessor.**test_default_pp**()

This test checks whether the Pp module returns the correct output given the default list of preprocessors.

The list containing the default selection and sequence of preprocessors, default_pp, is passed to the Pp module. The difference between this test and test_individual_pp() is that the preprocessor methods are not called directly, but via preprocess_image() instead, which requires a list of preprocessors and image as input.

TestPreprocessor.test_Preprocessor.**test_default_pp_reverse_order**()

Tests whether the Pp module returns the correct output given the reversed list of default preprocessors.

The list containing the default selection and sequence of preprocessors is reversed. Therefore, three assertions are required. The first assertion checks whether the output image is correct. The second assertion checks whether the warning message is printed to stdout since '4'is called before '1' (i.e. binary threshold before gray scale). The thirt check whether the warning message is printed to stdout since '8' is called before '1' (i.e. normalisation before gray scale).

TestPreprocessor.test_Preprocessor.**test_pp_binary_thresh**()
Tests whether the binary threshold method returns the correct warning message.

The binary threshold method is called (indirectly) without the gray scale method being called.

TestPreprocessor.test_Preprocessor.**test_pp_binary_adap_thresh**()
Tests whether the binary threshold method returns the correct warning message.

The binary threshold method is called (indirectly) without the gray scale method being called.

TestPreprocessor.test_Preprocessor.**test_pp_normalisation**()
Tests whether the normalisation method returns the correct warning message.

The normalisation method is called (indirectly) without the gray scale method being called.

TestPreprocessor.test_Preprocessor.**test_pp_4_before_1**()
Tests whether the binary threshold method returns the correct warning message.

The binary threshold method is called (indirectly) before the gray scale method is called.

TestPreprocessor.test_Preprocessor.**test_pp_8_before_1**()
Tests whether the normalisation method returns the correct warning message.

The normalisation method is called (indirectly) before the gray scale method is called.

Module contents

TestReadDisplay package

Submodules

TestReadDisplay.test_ReadDisplay module

This module is responsible for testing the quality of ReadImage.py.

TestReadDisplay.test_ReadDisplay.**test_read_display**()
This test checks whether the ReadDisplay module returns the correct output.

The assertions checks whether the output string is equal to the expected numbers as depicted in the input images, for each of the three test images.

Module contents

CHAPTER 3

Part III: Team pages

This is a collection of (informal) notes, ordered by team. They contain *quick tip* on various topics; made by (individual) apprentices. They may be useful, but may be outdated.

You can find lessons-learned about (installing) the tools, the training, the extensions. Every apprentice is ask to add as many notes as possible; often in a *FAQ* style. One is also allowed to improve *older* pages; even of other teams.

Each team has a its own (new) directory, named after the team. One is allowed to add a page(s) about the team (members); like a logo.

3.1 Team pages

author Albert

date 23 Dec 2017

Concept

Each team (training-group) get an *own* directory (a subdir of this one), to place notes, store know-how, etc. That does not imply they are the only ones who may change those articles; it just a convenient place to start.

So whenever you find out something, that may be interesting for your team, or a future team: write a note, an article, and place it in your team-section. Or update an existing one (even of another team). In all cases, add you name as author or section-author and update the date

As a team, you may also decided (*are encouraged!*) to **improve** the general documentation. This may/does include collecting "notes" and moving them the a more generic place. Either about the training, or about the extension.

This part consist of a lot of 'notes'. They may be new, maintained or outdated. Also the general quality rules are lowered. This is un purpose: it is better to have and share a brainwave, then to have it forgotten. As long a the reader knows it status: by this intro everbody should know!

3.1.1 BubbleFish & BitVision (pilots)

Fig. 1: Team logo for the first pilot

author Albert

date Dec 2017

These two teams work party together during (two) pilot-phase of the training. Due them, I (Albert, the trainer) could finalize the training-concept. Like, having a team-section; the articles are written by the team; this intro (and the dir) is created afterwards.

(Thanks)

Basisopleiding Software Engineering in C++

Besturingssysteem: Windows 7

Fig. 2: Team logo for the second pilot

Intro

Om OpenCV werkend te krijgen zijn een paar eigenaardigheden gevonden. De standaard .lib files in OpenCV 3.1 zijn niet compatibel met Windows 7 - 64bit en CodeBlocks, maar het werkt wel op Visual Studios 2013. De bedoeling is om het werkend te krijgen op meerdere platformen, daarvoor is een vereiste om je eigen lib bestanden te bouwen met cmake. Je hoeft cmake niet te downloaden.

Installatie C++

Microsoft Visual Studio Express 2013 Voor dit project moet je Microsoft Visual Studio Express 2013 downloaden op de volgende link: https://www.microsoft.com/en-us/download/details.aspx?id=44914 Na het klikken op download, kan je wdexpress_full.exe downloaden met een size van 1,1MB. Het installeren van dit programma duurt lang. Ondertussen kan je OpenCV downloaden.

OpenCV

OpenCV is op de volgende link te downloaden van Windows: http://opencv.org/downloads.html

Als deze link niet werkt dan: opencv.org > documentation > tutorials > Introduction to OpenCV

Voor mensen die Windows gebruiken, volg de stappen bij 'Installation in Windows' t/m 4 en 5 wordt hieronder beschreven:

- Start>Control Panel (typ bij search: environment)>Edit the system environment variables >Advanced>Environment Variables.
- Bij System variables>Zoek 'Path' en Edit .
  - Voeg aan het uiteinde van de 'Variable value': een punt-komma en dan (daarna) de locatie van je bin.
  - Die staat bij opencv\build\x64\vc12\bin.

  Voorbeeld: `;D:\\Users\\Name\\Documents\\HighTech\\C++\\opencv\\build\\x64\\vc12\\bin`.

Om te kijken of het werkt: Open Command prompt en typ:: > path.

Ongeveer aan het einde staat de locatie die je zonet hebt geplakt.

Microsoft Visual Studio Express 2013 met OpenCV

Open Microsoft Visual Studio * Maak een nieuw C++ project, onder 'project' bij de tab Example properties:

- Configuration Properties->VC++ Directories -> Include Directories -> opencv\build\include.
- Configuration Properties->VC++ Directories -> LibraryDirectories -> opencv\build\x64\vc12\lib.
- Configuration Properties->Linker -> Input -> Additional Dependencies -> opencv\build\x64\vc12\lib\opencv_world310d.lib
- Configuration Properties->Linker ->General -> Additional Library Directories -> opencv\build\x64\vc12\lib.
- Testen met example op http://docs.opencv.org/3.1.0/d0/d7a/convexhull_8cpp-example.html

  Voordat je het programma uitvoert -> de vector<int> hull; in de example bij lijn 35(Ln35) is de code te groot ->om te resizen schrijf eronder: `hull.resize(points.size());`

Dan staat er het volgende onder elkaar:

```
vector<int> hull;
hull.resize(points.size());
convexHull(Mat(points), hull, true);
```

Alles staat nu goed en je kan het laten uitvoeren door op 'Local Windows Debugger of F5' te klikken. Als het niet werkt, kijk dan of je de stappen goed hebt gevolgd. Door spatie of enter te drukken ga je naar het volgende plaatje en als je op esc drukt, dan sluit het programma.

Voorbeeldprogramma's

Voor de OpenCV tutorials: http://docs.opencv.org/3.1.0/examples.html

Proces

De codes van C++ werken nu alleen op de mac en op Linux werkt alleen tesseract. Op windows werkt OpenCV met Microsoft Visual Studio, maar het werkt nog niet op Tesseract.

Basisopleiding Software Engineering in Python

> **status** This page is quite outdated! –Albert

Installatie Python

Omdat het de laatste versie is, wordt Python 3.5 gebruikt. Download: https://www.python.org/downloads/

De IDE die gebruikt is, is PyCharm. Dit is een IDE die vergelijkbare functionaliteiten heeft als Eclipse, maar makkelijker is om te downloaden (op de Sogeti laptops geeft ecplipse netwerkerror). Download: https://www.jetbrains.com/pycharm/

Opdracht: Een plaatje van zeven-segment display omzetten naar een string

Voor het inlezen van een image naar tekst kan het beste tesseract gebruikt worden. Zie het volgende stackoverflow topic om de discussie te lezen welke library je het best waarvoor kunt gebruiken. http://stackoverflow.com/questions/11489824/how-do-i-choose-between-tesseract-and-opencv Mijn idee is om beide naast elkaar te gebruiken: OpenCV voor het voorbewerken van de afbeelding, maar voor het specifieke inlezen van afbeelding naar text zou ik tesseract gebruiken.

Tesseract

Beschrijving

Tesseract is een programma voor optical character recognition. Hiermee kun je van een image waarop een tekst afgebeeld staat een string als output geven.

Download laatste release

https://github.com/tesseract-ocr/tesseract/releases (ik heb 3.04.01 gebruikt)

Voor het installeren van tesseract moet je de volgende stappen nemen:

- Pytesseract installeren (eventueel via je package mananger) in je libraries.
- Het programma tesseract zelf installeren. Dit kun je doen door de installer te gebruiken van https://github.com/UB-Mannheim/tesseract/wiki Tesseract installer
- Als laatste moet je in pytesseract.py aan je programma vertellen waar je het programma zelf hebt geinstalleerd op je computer:

```
# CHANGE THIS IF TESSERACT IS NOT IN YOUR PATH, OR IS NAMED DIFFERENTLY
tesseract_cmd = r'C:\\Program Files (x86)\\Tesseract-OCR\\tesseract.exe'
```

OpenCV

Beschrijving

OpenCV (Open Source Computer Vision Library) is een open source computer visie en machine learning software library. Het is gemaakt om het gebruik van perceptie door middel van machines te vergroten.

Download:

http://opencv.org/downloads.html (ik heb 3.1.0. Gebruikt)

Voor het installeren van openCV moet je de volgende stappen nemen:

- OpenCV installeren in je libraries
  - Open Pycharm>File>Settings>Project:pathways-extension-training>Project Interpreter
  - Kies als Project Interpreter bovenaan voor: 3.5.2 (C:Users'yourname'AppDataLocalProgramsPythonPython35-32python.exe)
  - Klik op de plusteken aan de rechterkant om een package te installeren en installeer opencv-python.

- Het programma OpenCV zelf installeren - Nu moet je een bestandje van de ene naar de nadere folder verplaatsen:
  - In de map opencv/build/python/2.7 kun je het bestand cv2.pyd vinden. Kopier dit bestand
  - Plak het in de volgende map: C:/Python35/lib/site-packages.
- Om te kijken of het werkt, gebruik je console of IDLE en check het versienummer:
  - import cv2
  - print (cv2.__version__)

  Als dit werkt, is het gelukt om OpenCV te installeren.

Aan de slag

Wanneer je tesseract hebt geinstalleerd, is de stap snel gemaakt om een programmaatje te schrijven waarmee plaatjes waar cijfers op staan om te zetten in een string. Echter, voor deze opdracht willen we zeven-segment cijfers kunnen inlezen. Hiermee heeft tesseract wat meer moeite, omdat hij dit lettertype niet kent. Hiervoor moet een dataset getraind worden.

Training van de tesseract data

Tesseract is in staat om letters te lezen die hij "geleerd" heeft door middel van een trainingsdataset. Deze trainingsdatasets zitten in het mapje "tessdata", en eindigen met .traineddata. Met nummers op een zeven segment display heeft tesseract moeite, omdat deze nummers nog niet getraind zijn. Het is mogelijk om een getrainde dataset van internet af te halen en in deze directory te plakken (bijvoorbeeld https://github.com/arturaugusto/display_ocr/tree/master/letsgodigital). Een andere optie is om zelf (door middel van scriptjes die op internet te vinden zijn) de data te trainen.

Voorbewerken van de plaatjes

Tesseract lijkt moeite te hebben met de "witte" stukken tussen de letters, wat kenmerkend is voor een zeven segment display. Het voorbewerken van de plaatjes lijkt de performance van van tesseract aanzienlijk te verbeteren, zoals een blur of een threshold (zie http://stackoverflow.com/questions/28935983/preprocessing-image-for-tesseract-ocr-with-opencv voor wat voorbeelden).

Pytest

Pytest wordt gebruikt om kleine testen te schrijven om je programma te testen. Volg de stappen van deze site om pytest-3.0.4 te installeren voor Python: http://doc.pytest.org/en/latest/getting-started.html

Bij het voorbeeld: Als het niet werkt met alleen pytest, typ dan bij Command prompt bij de juiste map:

```
pytest 'filename'.py
```

(bijvoorbeeld test.py)

Training Tesseract

1. Create a bunch of .tif files containing numbers in calculator-like font. Name the files as follows:

   ```
   calc.7digitregular.exp0.tif
   ```

(where calc is language, 7digititalics is font, 0 is file number, you can make up your own names as long as it follows the convention). If you created images in Paint, tesseract will probably give you all sorts of warning. I used GIMP instead (GNU Image Manipulation Program, an open source graphics editor)

2. (optional) Blur the images.
3. Open CMD and run the following command:

   ```
   tesseract calc.7digitregular.exp0.tif calc.7digitregular.exp0. batch.
   nochop makebox
   ```

   Here tesseract tries to search for characters in your dataset and guess what those characters are. Most probably it will guess wrong. The characters that it managed to find will be stored in a file calc.7digitregular.exp0.box. If the file is empty, it means tesseract didn't find anything, and you can't work with this! Try creating a file with different number strings, spacing the numbers differently, blurring the image.
4. Open the .box file in a text editor, in the first column you will see the characters that Tesseract thinks it found, and in the other columns are the coordinates (in pixels) for each of the character. Most probably half of the time it guessed wrong. Just delete the wrong characters here and type the correct ones instead.
5. Train Tesseract:

   ```
   tesseract calc.7digitregular.exp6.tif calc.7digitregular.exp6 box.train
   ```
6. Generate a file with set of possible characters that can be encountered in your "language" on the basis of .box files:

   ```
   unicharset_extractor calc.7digitregular.exp6.box
   ```
7. Create a separate file defining font properties: Fontname <italic> <bold> <fixed> <serif> <fraktur>
8. Do feature clustering and some other stuff:

   ```
   shapeclustering -F font_properties -U unicharset calc.7digitregular.exp0.tr calc.
   ↪7digitregular.exp6.tr ... <list all your .tr files>
   mftraining -F font_properties -U unicharset -O calc.unicharset calc.7digitregular.
   ↪exp0.tr calc.7digitregular.exp6.tr ... <list all your .tr files>
   cntraining calc.7digitregular.exp0.tr calc.7digitregular.exp6.tr ... <list all␣
   ↪your .tr files>
   ```
9. Rename all files that have just been created (unicharset, shapetable, normproto, inttemp, and pffmtable) with the prefix calc. Execute this: combine_tessdata calc.
10. You can now use your new trained set to recognize some more new fonts or new examples, repeat steps 3-9.

    ```
    tesseract calc.7digitregular.exp5.tif calc.7digitregular.exp5 -l
    calc batch.nochop makebox      tesseract calc.7digititalics.exp1.tif calc.
    7digititalics.exp1 -l calc batch.nochop makebox
    ```

Version Control

Algemeen

Het doel van versiebeheer is om als groep met de laatste versie te werken op je harde schijf en niet op de browser te werken. Dit doen we door Bitbucket en Mercurial te gebruiken. Sphinx is een tool om documenten van reStructered-Text files(.rst) om te zetten naar bijvoorbeeld een HTML website.

Bitbucket

Maak een account op Bitbucket.org om de codes met elkaar te kunnen delen. Als je een account hebt gemaakt, zoek naar Pathways-Extension (Training) van Albert Mietus. Op de hoofdpagina van Pathways-Extension (Training) staat de algemene informatie over dit project.

Fork

Ieder maakt zijn eigen Fork op Bitbucket: Fork > Alles staat goed en maak je Fork repository. Nu staat er bij de hoofdpagina dat ook jouw fork erbij staat. Als je al een fork hebt gemaakt en je wilt je naam veranderen: Ga naar je eigen Fork pagina en ga naar Settings die linksbeneden staat. Je kan dan je naam veranderen.

Clone

Clone je fork van de site, zodat je het op de computer hebt staan. Dit doe je door op je eigen pagina op clone aan de linkerkant te drukken. Kopieer de link en open de command prompt. Verander de locatie als je dat wilt en tik hg clone in en plak erachter de link. Als je dit hebt gedaan dan zou de clone op de aangegeven locatie zijn.

Master

De master van dit project is Albert Mietus en zorgt dat alles dat erop komt, goed staat. De Master accepteert alle wijzigingen die door de anderen zijn gedaan.

Mercurial

Mercurial is te downloaden op de volgende link: https://www.mercurial-scm.org/ Als je iets wilt aanpassen, pull dan de laatste versie en merge het met die van jou. Zorg wel ervoor dat jouw status gecommit is en dat je de laatste versie hebt om die met een ander te mergen. Door dit te doen is jouw Local up to date. Hierna kan je de aanpassingen toevoegen en committen. Uiteindelijk heb jij de laatste versie met de aanpassingen en kan je het naar de anderen pushen. Door te pushen kunnen andere teamleden de laatste versie van jou pullen en ermee verder werken.

Pull and Push

Open de map van je project en gebruik de rechtermuisknop op een leeggedeelte van de map>TortoiseHG>Synchronize. In het midden staan de pull en push knoppen en daaronder van wie je het wilt pullen. Als je het van iemand anders wilt pullen, zoek zijn/haar link op via BitBucket.

Merge and Commit

- Mergen: map>TortoiseHG>Update
- Commit: map>HGCommit

Command prompt

Verander eerst je locatie naar die map door (change direction): cd C:...\pathways-extensions-training Daarna kan je door: hg push, hg pull, hg merge, hg commit Daarnaast kan je hg heads gebruiken om de verschillende lijnen te bekijken en om te bekijken met wat je gaat mergen als je dat wilt. Hg status geeft aan of er iets is veranderd en nog niet gecommit is. Voor problemen kan je altijd kijken naar de status: hg status of TortoiseHG>View file status

Symbol	Description
M	Modified
A	Added
?	Unknown
!	Missing
I	Ignore
R	Removed

Als je bijvoorbeeld een document wilt toevoegen, voeg het eerst bij een directory toe. Daarna moet je naar: map>TortoiseHG>Add files om ze toe te voegen.

Sphinx-doc

Op deze pagina: http://www.sphinx-doc.org/en/1.4.8/install.html#windows-install-python-and-sphinx staan de stappen die je moet maken om sphinx te installeren op windows, maar niet alle stappen komen overeen dus volg de stappen hieronder:

Python

Als je Python nog niet hebt, download het op de volgende site: https://www.python.org/downloads/ Download Python 3.5.2 en installeer. Tijdens het installeren, komt er een keer 'Add python . . . ' vink dat aan.

Download get-pip.py bij: https://bootstrap.pypa.io/get-pip.py.

- Maak een map aan om dat erin te doen. Open get-pip.py om het te installeren.
- Als je het hebt geinstalleerd, open command prompt en typ: pushd D:...\Sphinx (dus pushd met de locatie van de Sphinx map).
- Ga naar Control PanelSystem and SecuritySystem > Advanced system settings > Environment Variables.
- Voeg bij System Variables>Path de locatie van de pip.exe file toe aan de achterkant. Je voegt dus het volgende aan toe: C:...\Scripts
- Ga daar met je command prompt naartoe en als je van D: naar C: moet gaan. Typ C:
- Dan verandert het en uiteindelijk kan je via cd (change direction) dan de locatie erna toevoegen.
- Als je de juiste locatie hebt, typ : pip

Als dat werkt, tik Sphinx-quickstart. Dan vraagt Command prompt om de locatie waar je het wilt hebben. Vul dat in en daarna:

- Separate source and build directories: No
- Name prefix for templates and static dir: Enter
- Fill your project name and after that your name
- Project version: 1.0
- Project release [1.0]: 1.0.0
- Source file suffix [.rst]: Enter
- Name of your master document (without suffix) [index]: Houd het op default, dus alleen enter
- Autodoc: yes

Daarna komen er andere vragen, maar die kan je negeren totdat ze vragen over 'Create Makefile'. Tik y in en daarna vragen ze of je het op Windows doet.

- Verander de locatie bij Command prompt naar de locatie waar sphinx-build staat (Scripts map).
- Typ dan bij Command prompt: Sphinx-build –b html 'Source directory' 'Build directory'
- Het is dus bijv: Sphinx-build –b html D:...\ Sphinx D:...\Sphinx_build
- Als het de map niet kan vinden, kopieer je map naar de desktop en verander hetgene wat je typte naar de locatie van je Sphinx map op de desktop. Door dit te doen, maak je een html.
- Door de locatie van je Command prompt te veranderen naar de locatie van make.bat die bij de map Sphinx staat, kan je wel 'make html' gebruiken.

- Je kan je HTML openen in de map van Sphinx>_build>html>index.html.

Als je er niet uitkomt, zoek op youtube naar 'Using Sphinx to Document Python Code'. Daar leggen ze uit wat je moet doen, alleen is het voor linux bedoelt.

Documentatie bewerken

De html site is gemaakt uit de index.rst die je met Notepad kan openen en bewerken. Als je het hebt bewerkt moet je bij Command prompt opnieuw 'make html' doen om het up te daten.

Thema veranderen

Als eerst installeer je een nieuwe thema door bij command prompt: 'pip install sphinx _rtd_theme'. Je locatie bij command prompt is bij je Sphinx map waar conf.py staat. Na het installeren moet je de conf.py editted door Edit with IDLE> Edit with IDLE 3.5 (32-bit). Je opent dan een document en zoek naar html_theme. Verder de huidige theme naar 'sphinx_rtd_theme' die je zojuist hebt geinstalleerd. Als je nu weer make html doet dan is de thema van je html verandert.

Add Figure

'.. figure:: 'LOCATIE"

Sphinx kan geen jpeg openen, maar wel png.

Link

Als eerst moet je een extra RST file aanmaken om die te linken aan het woord. De RST file staat dan in je Sphinx map.

Voor meer informatie kan je de documentatie bekijken: https://media.readthedocs.org/pdf/sphinx/stable/sphinx.pdf

Notepad ++

Download Notepad ++ 7.2 op de volgende site: https://notepad-plus-plus.org/download/v7.2.html Notepad ++ maakt het makkelijker voor Sphinx en hgignore.

3.1.2 Nautilus Pi

date Jan 4th 2017 - Feb 2nd 2017

Welkom op de team-pagina van Nautilus Pi.

Fig. 3: Logo of the first official trainings-team

Algemeen

Hier vind je algemene notities van het team. Het eerste stuk gaat over version control. Vervolgens beschrijven we de bevindingen van ons onderzoek naar decoupling van bestaande code en unit-testen van de pilotgroepen.

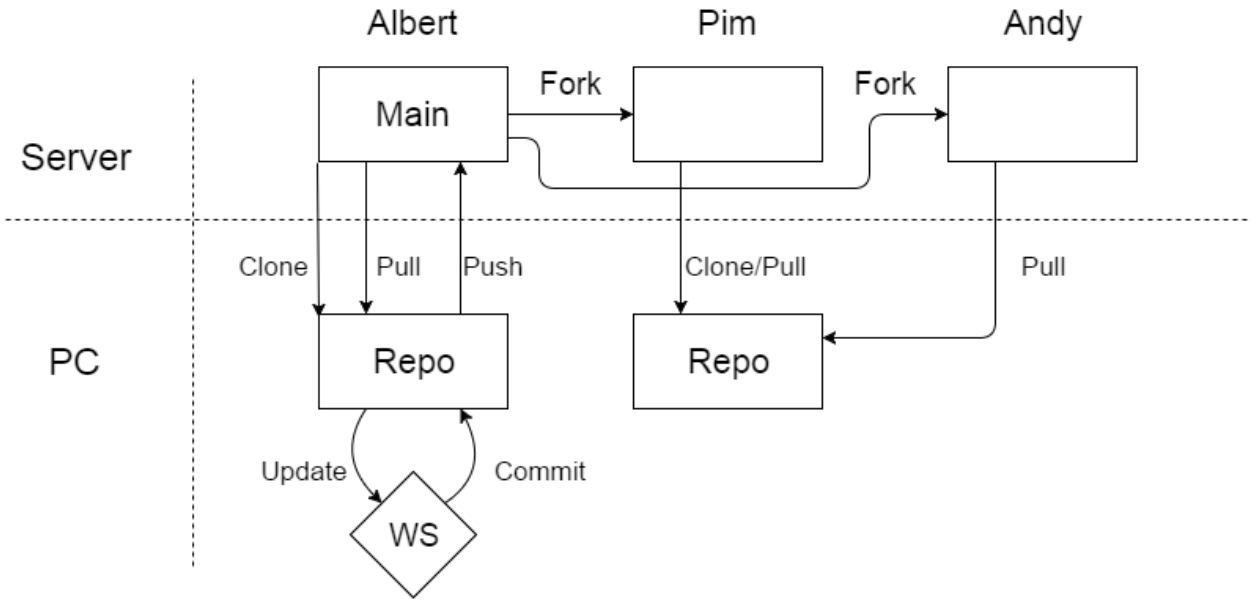
In de MOP gallerij tref je foto's aan die gemaakt zijn van aantekeningen op het bord tijdens de (sprintwissel) sessies.

Notities van de basisopleiding Software Engineering

Dit document bevat de notities van de basisopleiding Software Engineering van Nautilus Pi

Mercurial (Hg)

In het onderstaande diagram wordt de basis van versiebeheer weergeven.



De main wordt op de server gehost. Om deze files lokaal te verkrijgen moet een “pull” gedaan worden.

Dit kan gedaan worden doormiddel van “hg pull [source html]” vanuit de target directory. Hierdoor ontstaat er een lokale repository. Een pull “merged” ook automatisch de server repository met de bestaande repository.

Attention: De eerste keer dat je een server versie lokaal wilt maken zul je een “hg clone [source html]” moeten uitvoeren in plaats van een pull.

Om gebruik te kunnen maken van deze files moet een “hg update” gedaan worden om een workspace te verkrijgen. Hierin maakt men aanpassingen.

Als aanpassingen gemaakt zijn moet er gesaved worden om lokaal je bestanden op te slaan. Vervolgens kan je doormiddel van een “hg commit” de locale repository updaten.

Om de veranderingen beschikbaar te maken naar de rest van het team moet je een “hg push” doen om de repository op de server te zetten.

Om een kopie van andermans server repository te krijgen moet een “fork” gedaan worden binnen de gebruikte server. Deze kopie kan vervolgens aangepast worden zoals beschreven. Wanneer een aanpassing “gepushed” is zullen andere gebruikers een “hg pull” van jou “fork” moeten doen om de aanpassingen te mergen met hun versie.

Tot slot kan je ook lokaal een kopie maken van je repository door deze te “branchen”. Dit kan wenselijk zijn om meerdere versies tegelijkertijd te hanteren.

Wanneer doet men wat

- Save - Wanneer je wilt runnen.
- Commit - Als je een versie hebt waar je naar terug wilt kunnen.
- Push - Als je iets hebt wat beschikbaar gemaakt moet worden aan anderen.

- Pull - Als jij wijzigingen van anderen nodig hebt.

Decoupling

Bij het schrijven van code is het goed gedrag om verschillende modules te “decouplen”. Dit houdt in dat elke module apart te gebruiken is en dat elke module uit het systeem gehaald kan worden om te testen. Wanneer dit succesvol gedaan is wordt het systeem als *loosely coupled* gezien.

Decoupling van bestaande software

Onderzoek naar modulariteit software

Huidige opbouw en naamgeving

Er zijn op dit moment drie modules aanwezig met eigen functionaliteit

Design

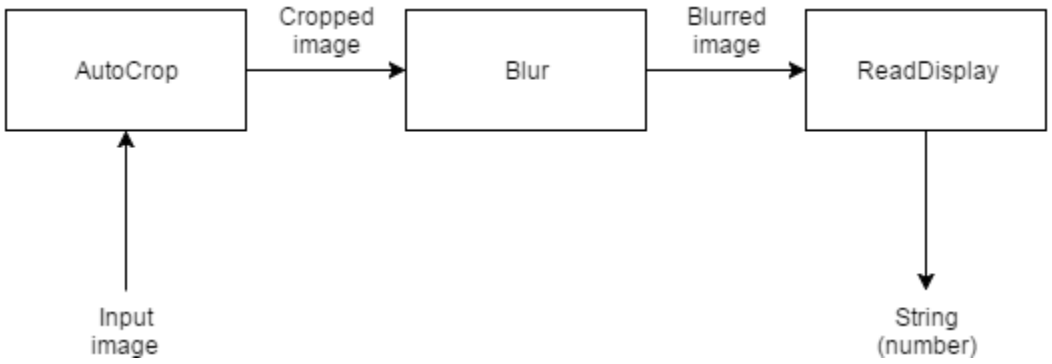


Fig. 4: LVL 1 design van alle modules.

Overzicht

- **AutoCrop** is in staat om een grijs scherm te herkennen van een rekenmachine. Na het herkennen van het scherm, is deze module in staat om een uitsnede van dit scherm te maken. De output van dit scherm is ook een vierkante image waar alleen het scherm van de rekenmachine in te zien is.
  - Input =>Image
  - Output=>Image (Cropped image)
- **Preprocessor** zorgt voor de preproccessing van de image. Hierdoor is makkelijker voor de software om de tekst in de image te lezen. Binnen deze module wordt alle onnodige en onbelangrijke informatie binnen de image verwijderd.
  - Input =>Image
  - Output=>Image (Preprocessed image)
- **ReadDisplay** extraheert de tekst uit een image. Dit kunnen meerdere lines en/of woorden zijn.

  - Input =>Image
  - Output=>String (Gelezen tekst)

Coupling

Alle modules zijn losgekoppeld van elkaar en zijn in staat om seperaat te runnen. De modules *Preprocessor* en *AutoCrop* zijn 100% procent op zichzelf werkend. Als in beide willekeurige images worden gestopt zullen deze modules altijd een image teruggeven. Echter kan de teruggegeven image bij de *AutoCrop* niet de juiste zijn voor de invoer van *ReadDisplay*.

De module *ReadDisplay* geeft een string terug als het tekst herkent in een image. Hoe deze tekst er uitziet kan verschillen. Zo kunnen er lijnen, spaties of tabs teruggeven worden. Het is niet goed te voorspellen wat deze module teruggeeft; De output is afhankelijk van de input.

Conclusie

Alle modules werken volledig los van de rest en zijn niet afhankelijk van ander modules.Toch wordt aanbevolen om een controle module the maken na de *ReadDisplay* module. Dit door de output die kan verschillen.

Unit test test rapport

Introductie

Dit document beschrijft het onderzoek naar de aanwezige tests voor de code voor de module DisplayLezen van de RekenRobot.

Er wordt gekeken naar 3 zaken:

- Het functioneren van de tests
- De resultaten van de tests
- Het aantal tests in vergelijking met de cyclomatische complexiteit van de code

De tests

De code is opgebouwd in de volgende modules:

- **AutoCrop** Deze module bevat de code voor het detecteren van het display.
- **Preprocessor (voorheen "Blur")** Deze module bevat de code voor het voorbewerken van het verkregen beeld.
- **ReadDisplay** Deze module bevat de tesseract code voor het verkrijgen van het getal uit het voorbewerkte beeld.
- **Demo** Deze module bindt alle voorgaande modules aan elkaar.

De volgende tests zijn aanwezig:

- TestAutoCrop
- TestPreprocessor
- TestReadDisplay

Resultaat

De volgende tabel laat zien of de tests succesvol uitgevoerd werden, hoeveel tests er per testfile aanwezig zijn en de resultaten van die tests.

Test	Runnable	Aantal Tests	Resultaat
TestAutoCrop	Ja	1	OK
TestPreprocessor	Ja	1	OK
TestReadDisplay	Ja	1	NOK, 1 Failed

Cyclomatische complexiteit

Cyclomatische complexiteit wordt gedefinieerd als het aantal lineair onafhankelijke paden. De cyclomatische complexiteit wordt gebruikt om het aantal testen te bepalen die gedraaid moeten worden om alle paden te doorlopen.

Waarin het volgende geldt:

- Minst aantal tests = Cyclomatische complexiteit
- Meest aantal tests = 2^Cyclomatische complexiteit

De cyclomatische complexiteit voor elke module is beschreven in de onderstaande tabel.

Module	Cyclomatische complexiteit
AutoCrop	7
Preprocessor	2
ReadDisplay	2

Op basis van deze cyclomatische complexiteiten kunnen de volgende minimale en maximale ondergrens voor het aantal tests bepaald worden.

Module	Minimale ondergrens tests	Maximale ondergrens tests
AutoCrop	7	128
Preprocessor	2	4
ReadDisplay	2	4

Conclusies

Het functioneren van de tests

De tests kunnen allemaal probleemloos uitgevoerd worden.

De resultaten van de tests

De bestaande tests slagen met uitzondering van de TestReadDisplay. Deze test levert 3 testbeelden van rekenmachine LCD's met getallen erop en test vervolgens of het programma de juiste getallen output. In deze test output het programma 2 van de 3 getallen correct en 1 incorrect. Er moet gekeken worden naar de module om te kijken of deze verbeterd kan worden.

Het aantal tests

In de vergelijking van de aantal tests met de minimaal aantal tests benodigd volgens de berekende cyclomatische complexiteit blijkt dat elke module te weinig tests bevat. Er moet gekeken worden of de code minder complex gemaakt kan worden. Anderzijds kan er gekeken worden of er meer tests gemaakt kunnen worden om max path coverage te kunnen bereiken.

MOP Gallerij

Op deze pagina worden aantekeningen die gemaakt zijn tijdens de bijeenkomsten bijgehouden (nieuwste bovenaan). Klik op de foto om deze te Vergroten.

date Do 26 jan 2017

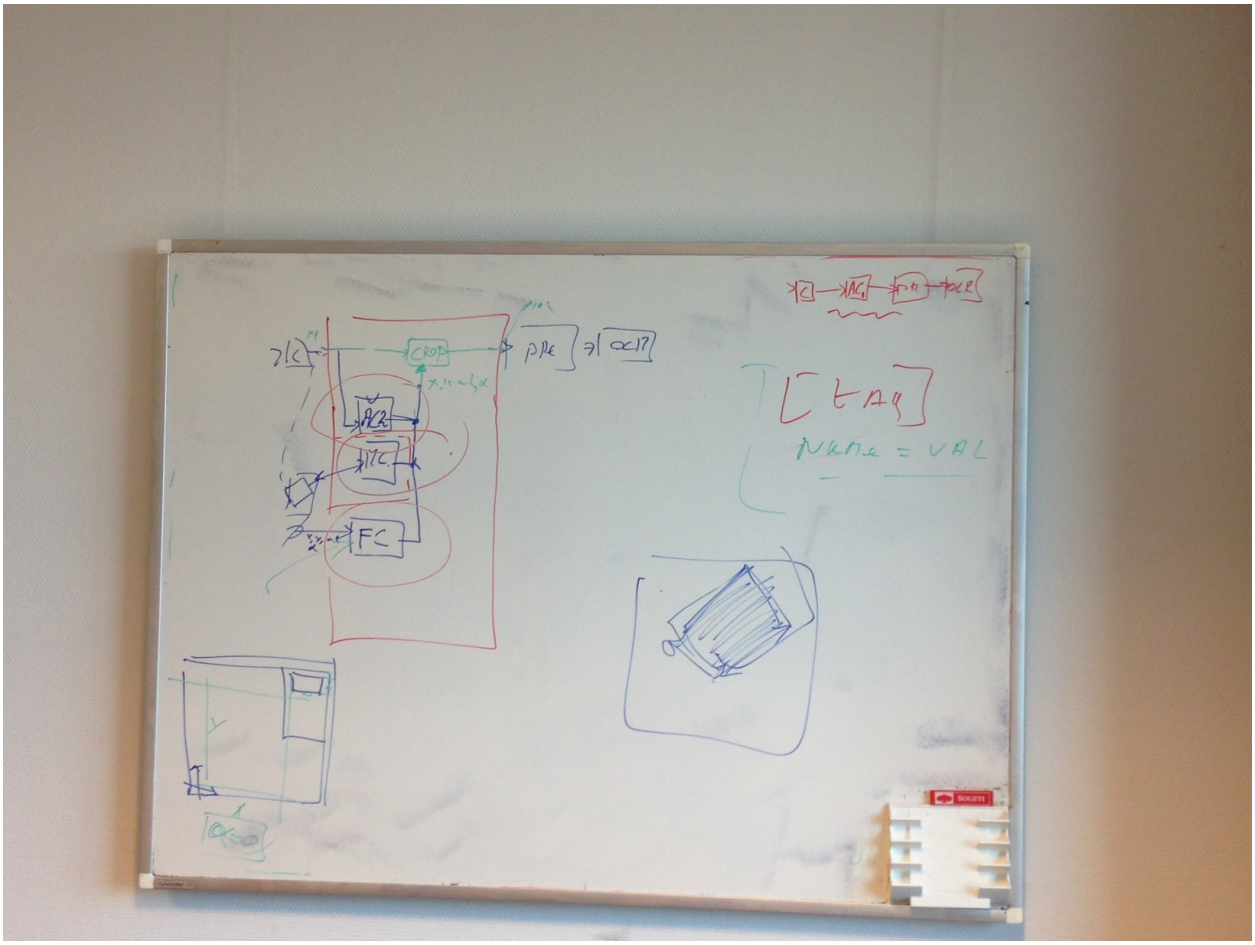


Fig. 5: Vernieuwde design met AutoCrop2, ManualCrop, FixedCrop. Crop coordinaten worden via een van deze modules (afhankelijk van eigen keuze) doorgestuurd naar Crop.

date Do 19 jan 2017

date Do 12 jan 2017

date Ma 9 jan 2017

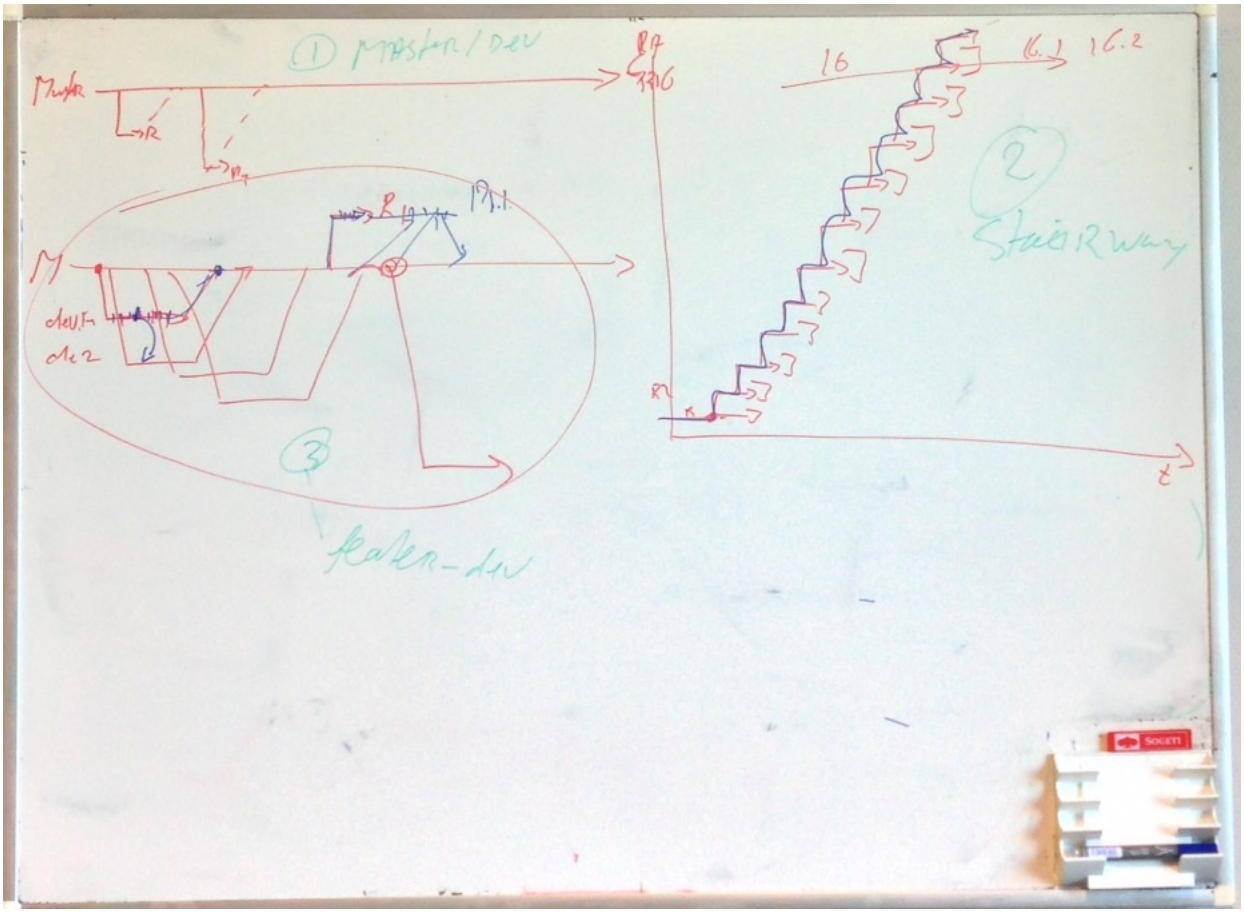


Fig. 6: 3 Verschillende manieren om te branchen van de master repo

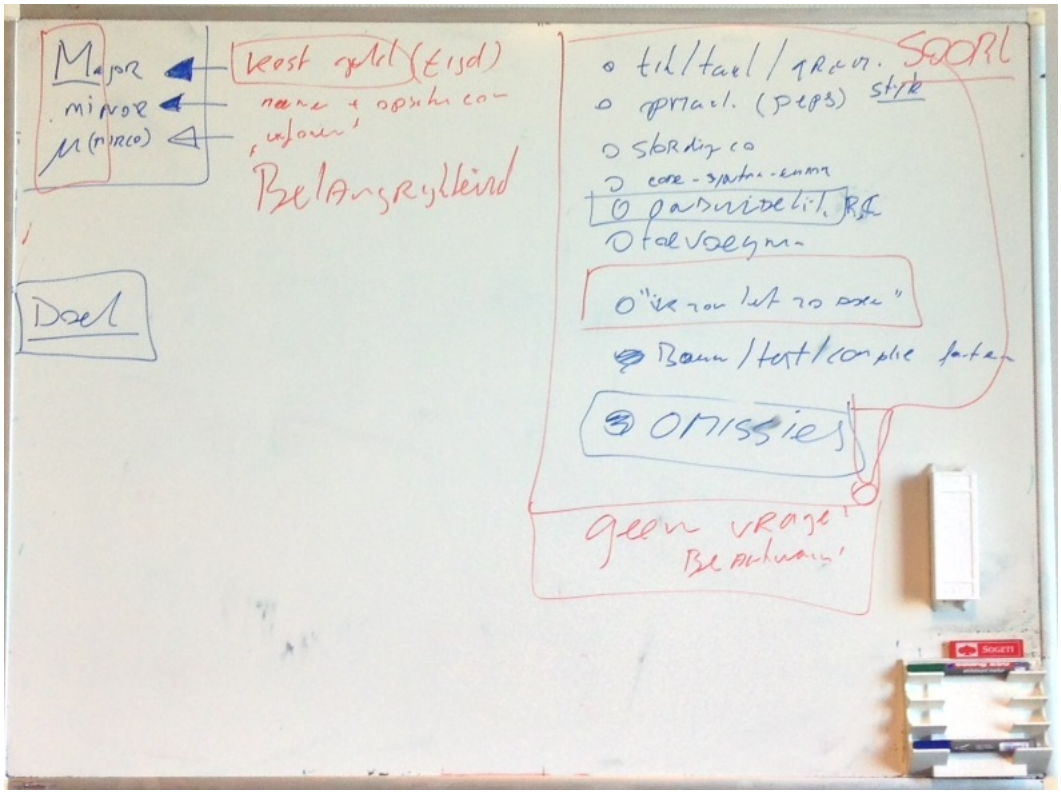


Fig. 7: De drie niveau's van code reviews (links) en soort fouten waar je op moet letten tijdens een dergelijke review (rechts).

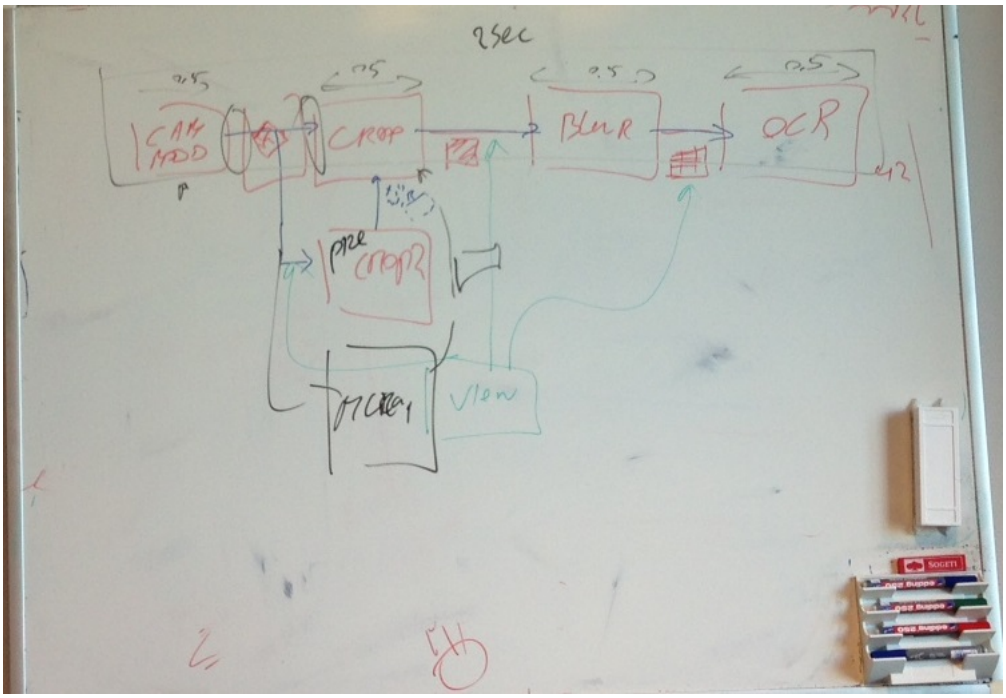


Fig. 8: Het geoogde design voor sprint 3 met precrop en manualcrop.

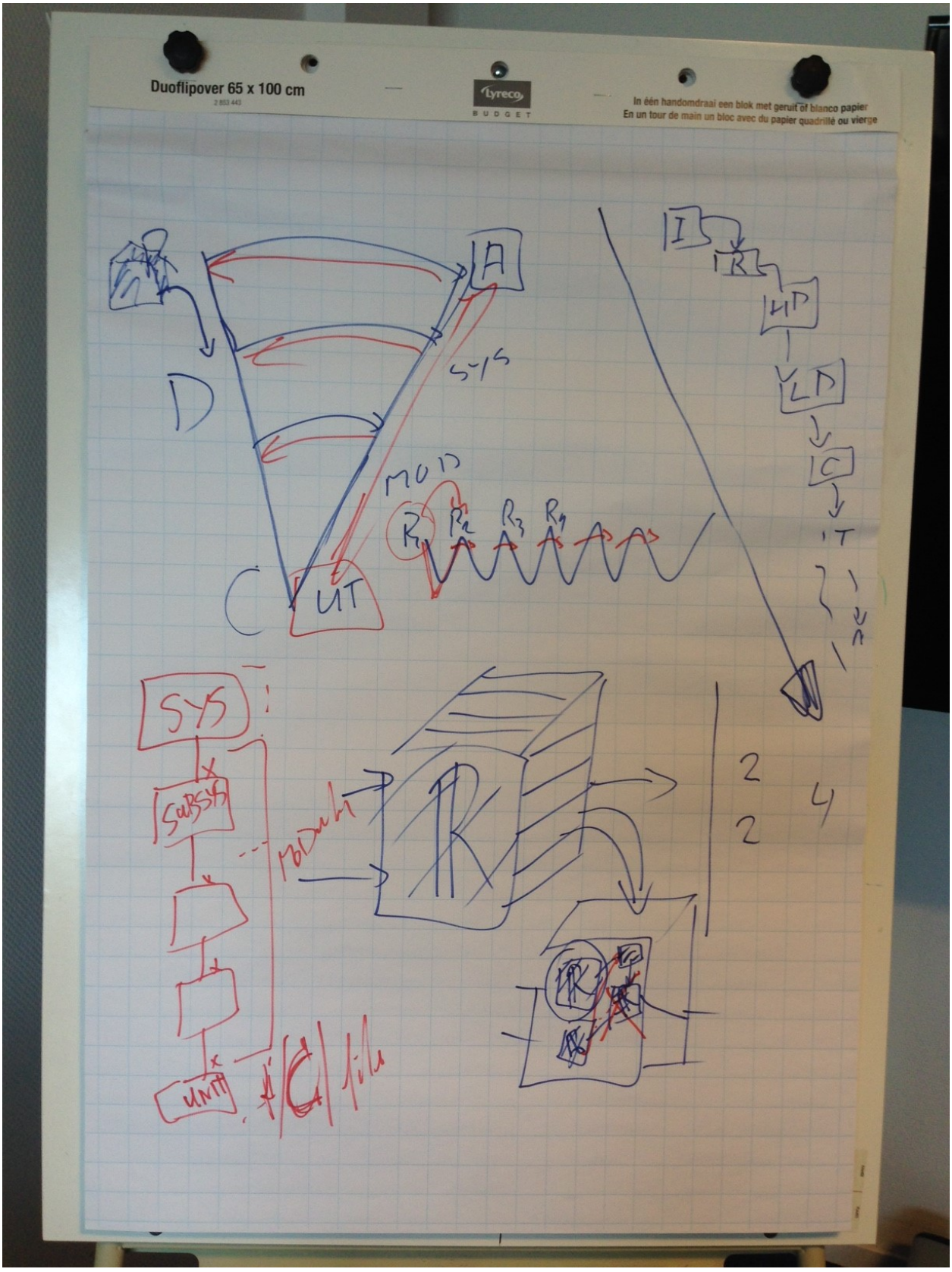


Fig. 9: Het onderscheid tussen module- en unittesten.

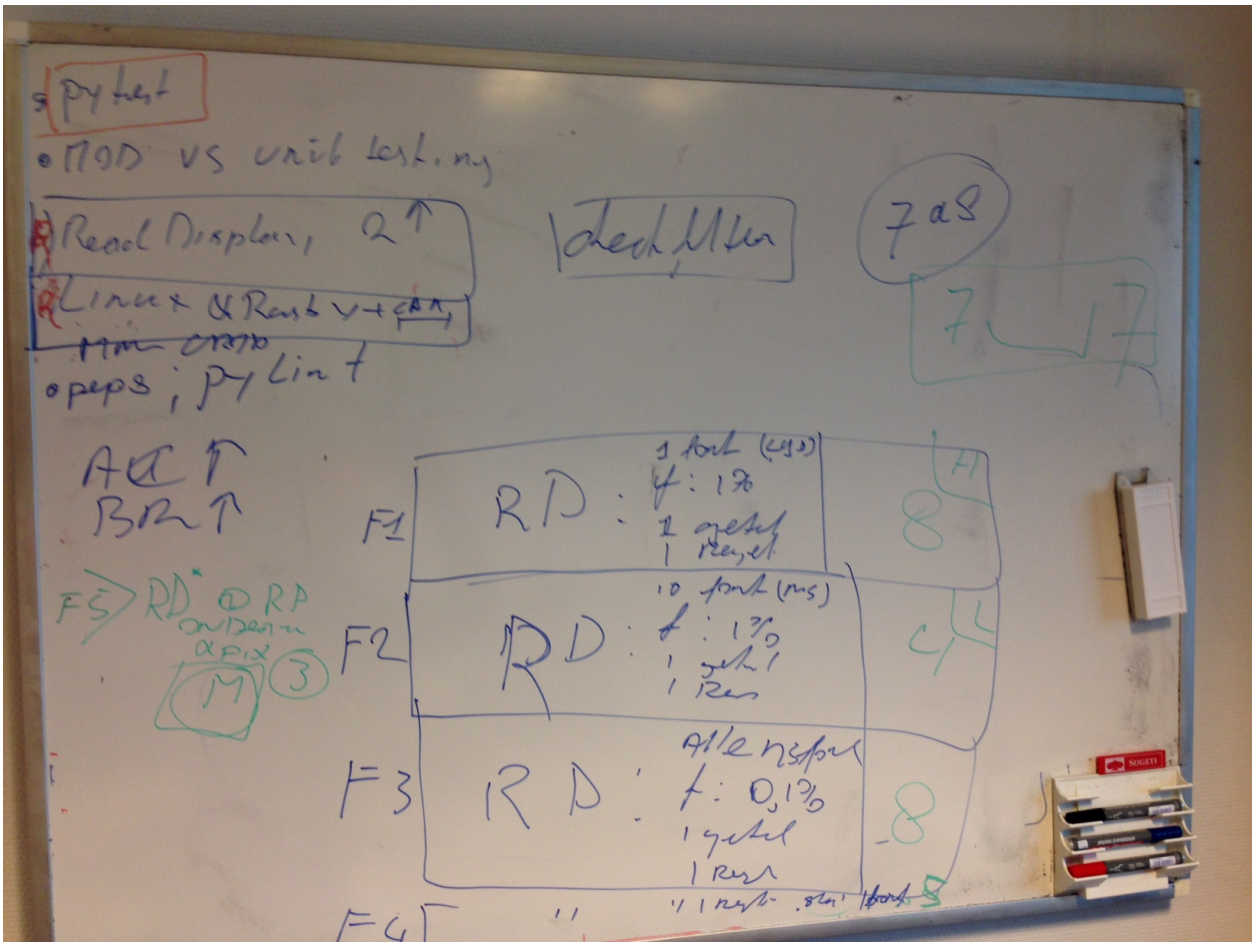


Fig. 10: Planning sprint 2 met bijbehorende features. Feature 1 is high priority, feature 5 is medium priority en feature 2 is low priority.

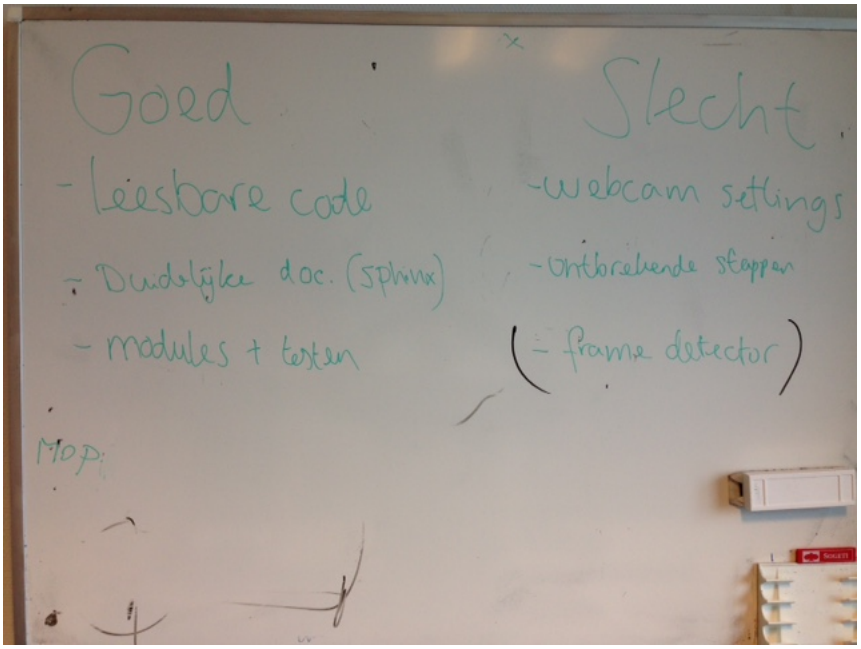


Fig. 11: Goede en minder goede dingen aan het werk van de pilot groepen.

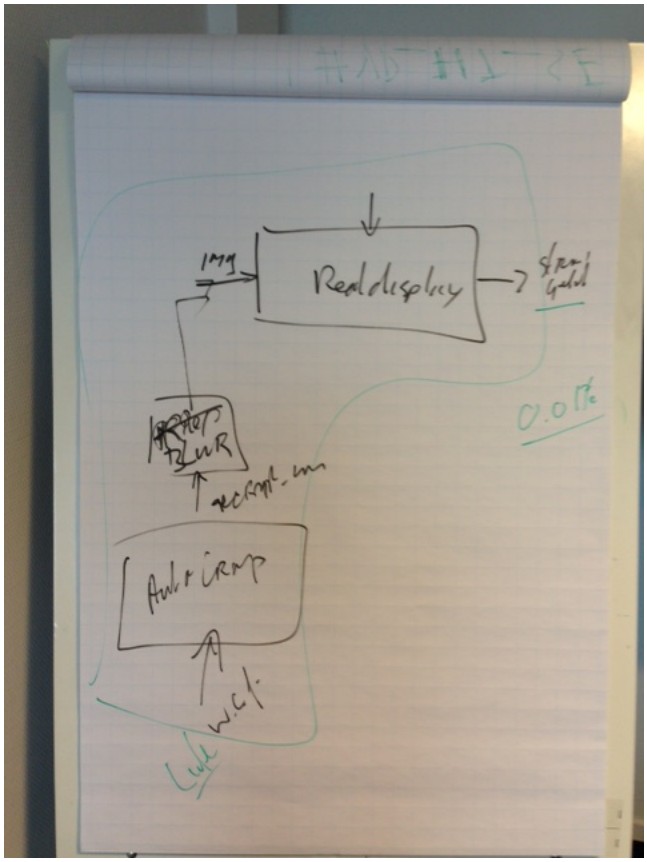


Fig. 12: Design dat beschrijft wat er met een input (plaatje of webcam frame).

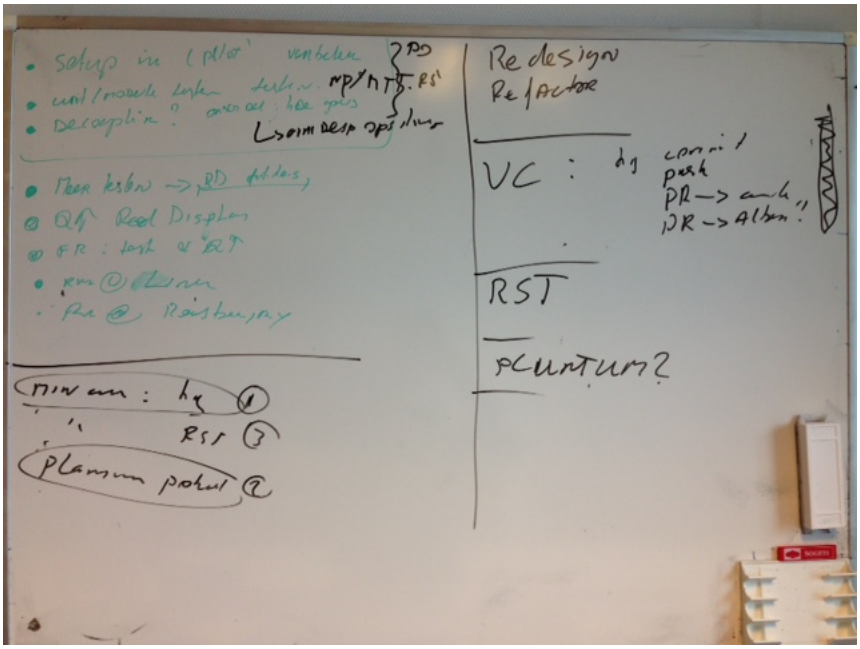


Fig. 13: Planning 1 (linksboven) en planning van de dag zelf (linksonder)

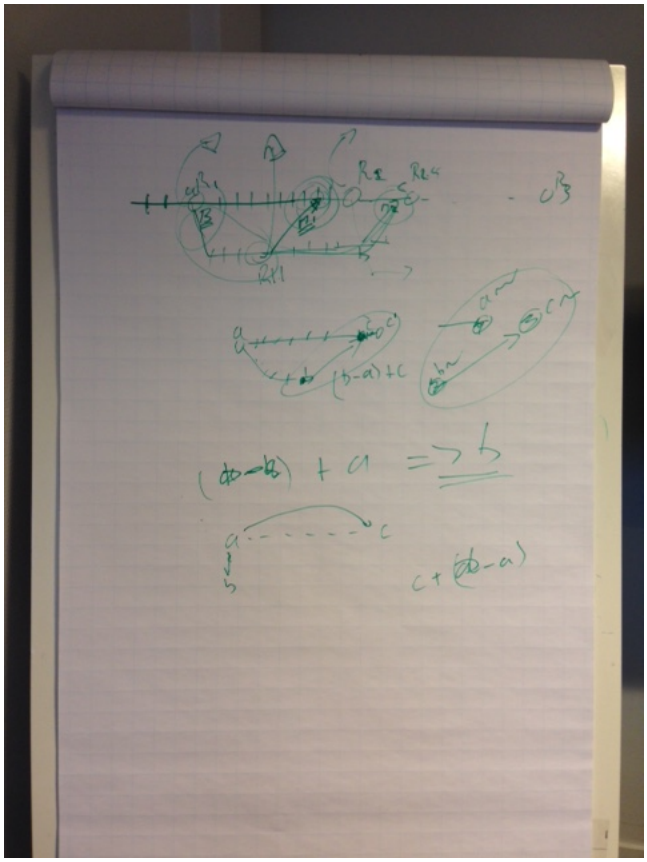


Fig. 14: Visualisatie van een merge.

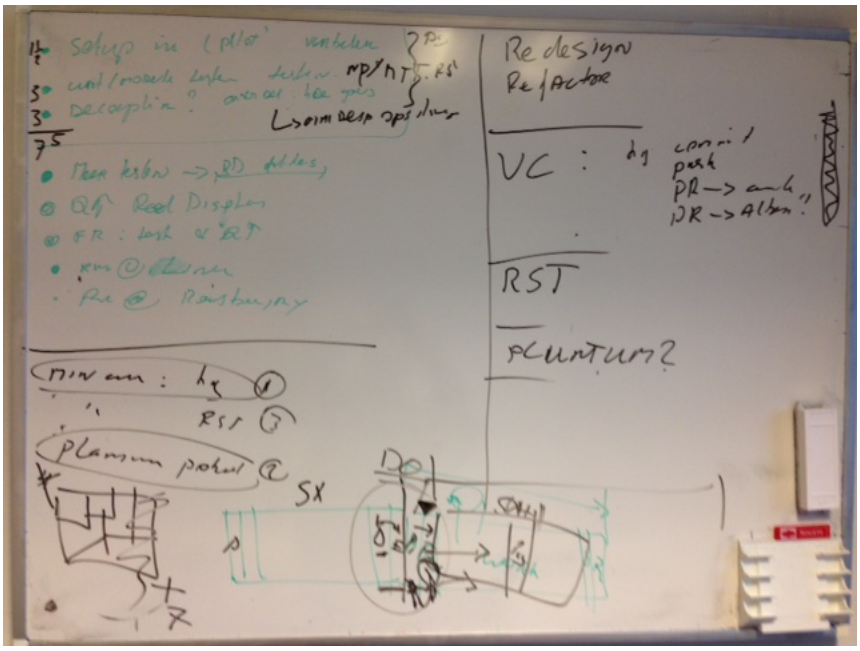


Fig. 15: Planning 1 en planning 2 uitgebeeld (zie onderaan). P1 heeft betrekking op de volgende periode, P2 alleen op de aankomende sprint.

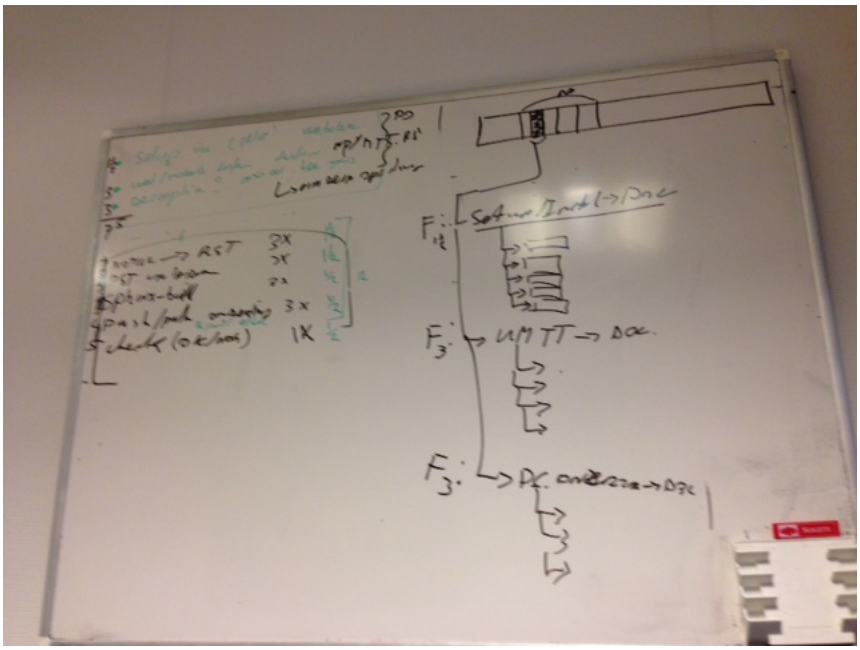


Fig. 16: Het onderscheid tussen features (wat lever je op?) en taken (wat ga je doen?).

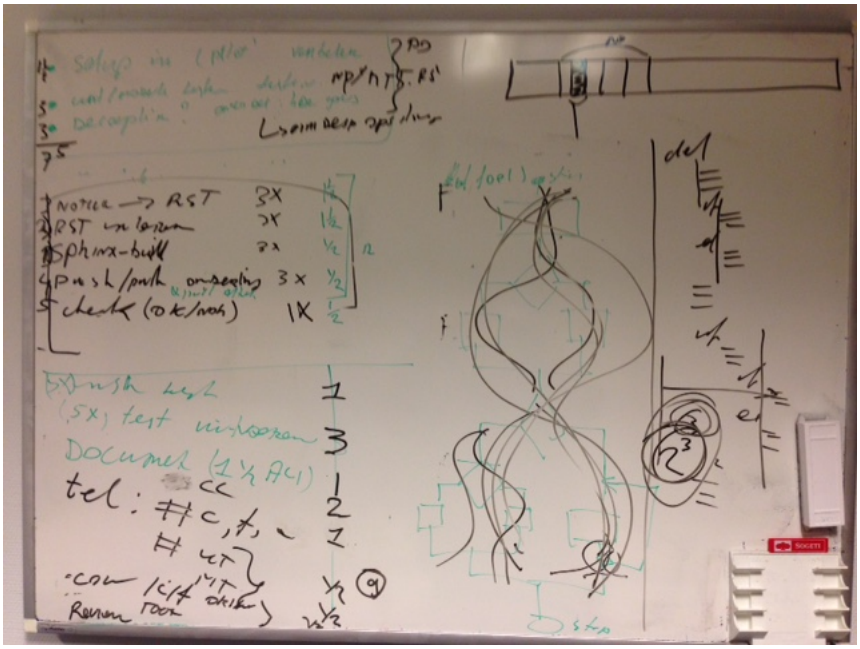


Fig. 17: Planning + schatting (links) en testingpaden (rechts).

Raspberry Pi

De huidige code werkt ook op de Raspberry Pi. Hier vind je een guide over het configureren van een Raspberry Pi om het up-and-running te krijgen.

Display Lezer voor de Raspberry Pi (First Time Setup)

Dit document dient als guide om een 'Out-of-the-box' Raspberry Pi te configureren voor de DisplayLezer. Deze stappen zijn al doorlopen en om het systeem snel op te configureren refereren wij door naar het document "Display Lezer voor de Raspberry Pi". Dit document dient als referentie.

Rasbian Jessie

Eerst installeren we de Rasbian OS voor de Raspberry Pi, dit is een Linux omgeving.

- Download NOOBS (New-Out-Of-the-Box-Software) van https://www.raspberrypi.org/downloads/noobs/.
- Plaats SD in de Raspberry Pi en connect een power supply, een display, een muis en een toetsenbord.
- Wanneer geprompt; installeer Rasbian Jessie vanuit NOOBS.

VNC (Virtual Network Computing)

Nu gaan we een VNC opzetten om de Raspberry Pi te besturen vanaf de computer.

- In Raspberry Pi: Open terminal.
- Voer het volgende in:

```
$ sudo raspi-config
```

- In het configuratie scherm: Zet 'Enable Camera' tot 'Yes' en zet in 'Advanced Options' -> 'SSH' en 'VNC' op 'Enable'.
- Klik op '<finish>' en voer het volgende in de terminal:

```
$ sudo apt-get update
$ sudo apt-get install tightvncserver
$ vncserver :1
```

- Je wordt gevraagd een password in te voeren. Dit moet een 8-cijferig password zijn.
- Download en installeer VNC van https://www.realvnc.com/download/vnc/
- Open VNC Viewer
- Start een nieuwe connection en voer als 'VNC-Server' [ip-address van de Raspberry]:1 (Het IP-Address is te verkrijgen door "ifconfig" in de terminal van de Raspberry Pi in te voeren).
- Voer een naam in en druk ok.

Vanaf nu kan je via VNC Viewer je Raspberry Pi besturen.

Nu moeten we nog instellen dat de VNC-Server draait vanaf boot zodat we in het vervolg alleen nog maar de raspberry pi nodig hebben.

- In de Raspberry Pi terminal, voer het volgende in:

```
$ cd /home/pi
$ cd .config

$ mkdir autostart
$ cd autostart
$ sudo nano tightvnc.desktop
```

- Voer in deze nieuwe file het volgende in:

```
[Desktop Entry]
Type=Application
Name=TightVNC
Exec=vncserver :1
StartupNotify=false
```

- Voer het volgende in de terminal in:

```
$ passwd

Password = raspberry
new password = rekenrobot
```

Nu hoeven we alleen nog een static IP in te stellen zodat we de VNC Viewer een profile kunnen aanleveren.

- Voer in de terminal het volgende in:

```
$ route -ne
```

- Noteer de Gateway IP.
- Voer nu in:

```
$ sudo nano /etc/resolv.conf
```

- Noteer de Domain Name Server IP's
- Voer nu in:

```
$ sudo nano /etc/dhcpcd.conf
```

- Voeg het volgende toe aan het eind van de file:

```
interface eth0
static ip_address=[De gateway IP van de route-ne maar verander het laatste getal naar␣
↪243]

static routers=[De Gateway IP van de route -ne]
static domain_name_servers=[De Domain name Servers van de resolv.conf gescheiden met␣
↪een spatie]
```

- Save deze file met Ctrl+O en sluit hem af met Ctrl+X
- Reboot de Raspberry Pi door het volgende in de terminal in te voeren:

```
$ sudo reboot
```

Vanaf nu heb je geen scherm, toetsenbord of muis meer nodig. Je kan bij het opstarten de VNC-Viewer opstarten met de informatie van de bovenstaande stappen. (VNC-Server = [static ip_address]:1)

OpenCV Installeren

Nu gaan we OpenCV installeren. Deze stap zal langer dan een uur duren, neem dit mee in je planning.

Eerst gaan we wat ruimte vrij maken.

- Voer het volgende in de terminal in:

```
$ sudo apt-get purge wolfram-engine
```

Hiermee verwijder je wolfram en maak je ~700 Mb vrij.

- Upgrade bestaande packages door de volgende commando's in de terminal in te voeren:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Dependancies installeren

- Voer de volgende regels in terminal in:

```
$ sudo apt-get install build-essential cmake pkg-config
$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
$ sudo apt-get install libgtk2.0-dev
$ sudo apt-get install libatlas-base-dev gfortran
$ sudo apt-get install python2.7-dev python3-dev
```

Nu gaan we OpenCV downloaden.

- Voer het volgende in de terminal in:

```
$ cd ~
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
$ unzip opencv.zip
$ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.1.0.
↪zip
$ unzip opencv_contrib.zip
```

Nu installeren we pip om python packages te installeren.

- Voer het volgende in de terminal in:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
```

Virtual Environment

Nu installeren we een virtual environment.

- Voer het volgende in de terminal in:

```
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/.cache/pip
```

- Open de '.profile' file door middel van het volgende commando.

```
$ sudo nano ~/.profile
```

- Voeg aan het eind van deze file het volgende toe:

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

- Update de file door het volgende commando:

```
$ source ~/.profile
```

Attention: Dit commando moet gebruikt worden elke keer dat de terminal opnieuw opent!

- Voer het volgende in de terminal in:

```
$ mkvirtualenv rr -p python3
```

Attention: Om in de virtuele omgeving te werken gebruik je het commando:

```
$ workon rr
```

Je weet dat je in de virtuele omgeving zit door de aanduiding (rr) aan het begin van elke regel in terminal.

- Install numpy doormiddel van het volgende commando in terminal:

```
$ pip install numpy
```

Compileren en installeren van OpenCV

Attention: Dit onderdeel duurt 1h12 min op de Raspberry Pi 3 en 1h35 op de Raspberry Pi 2!

- Zorg dat je in de virtuele omgeving zit door het volgende commando te gebruiken:

```
$ workon rr
```

- Nu gaan we de build klaarzetten met de volgende commando's:

```
$ cd ~/opencv-3.1.0/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \ -D CMAKE_INSTALL_PREFIX=/usr/local \ -D INSTALL_
↪PYTHON_EXAMPLES=ON \ -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.1.0/modules \ -
↪D BUILD_EXAMPLES=ON ..
```

- Nu compilen we OpenCV met alle 4 kernen door het volgende commando:

```
$ make -j4
```

Dit is het gedeelte wat 1h12min in beslag neemt!

- Nu gaan we OpenCV installeren met de volgende commando's

```
$ sudo make install
$ sudo ldconfig
```

- Nu gaan we een filename veranderen om latere bugs te voorkomen. Voer het volgende in je terminal in:

```
$ cd /usr/local/lib/python3.4/site-packages/
$ sudo mv cv2.cpython-34m.so cv2.so
```

- En we sym-linken onze OpenCV met onze virtuale omgeving.

```
cd ~/.virtualenvs/rr/lib/python3.4/site-packages/
ln -s /usr/local/lib/python3.4/site-packages/cv2.so cv2.so
```

- Nu kun je de installatie testen door het volgende in de terminal in te voeren.

```
$ source ~/.profile
$ workon rr
$ python
>>> import cv2
>>> cv2.__version__
```

Als dit zonder errors '3.1.0' returned is de installatie succesvol.

Raspberry Pi Camera

- Connect de Raspberry Pi camera met de Raspberry Pi.

- Als je nog niet in de virtualenv zit, voer dan het volgende in de terminal in:

```
$ source ~/.profile
$ workon rr
```

- installeer de picamera module met de volgende commando's:

```
$ pip install "picamera[array]"
```

- Om te testen of je camera functioneert kun je het volgende script gebruiken:

```
# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2

# initialize the camera and grab a reference to the raw camera
camera = PiCamera()
raw_capture = PiRGBArray(camera)

# allow the camera to warmup
time.sleep(0.1)

# grab and image from the camera
camera.capture(raw_capture, format="bgr")
image = raw_capture.array

# display the image on screen and wait for a keypress
cv2.imshow("Image", image)
cv2.waitKey(0)
```

DisplayLezer Code

- Installeer mercurial met het volgende commando:

```
$ sudo apt-get install mercurial
```

- Clone je repository door naar de gewenste destination folder te gaan en het volgende commando te gebruiken:

```
$ hg clone [jouw repository link]
```

De main file is pathways-extensions-training/RekenRobot/Src/DisplayLezen/Demo/MainFileReadImagePi.py

Tesseract OCR

- Installeer Tesseract OCR met het volgende commando:

```
$ sudo apt-get install tesseract-ocr
```

- Installeer pytesseract met het volgende commando:

```
$ sudo pip install pytesseract
$ sudo pip3 install pytesseract
```

- Installeer python imaging:

```
$ sudo apt-get install python-imaging
```

- Installeer imutils:

```
$ sudo pip3 install imutils
```

- Installeer Pillow:

```
$ sudo pip3 install --upgrade pillow
```

- Herstart de Raspberry Pi.

```
$ sudo reboot
```

- Download letsgodigital.traineddata:

```
$ wget -O letsgodigital.traineddata "https://github.com/arturaugusto/display_ocr/blob/
↪master/letsgodigital/letsgodigital.traineddata?raw=true"
```

- Plaats letsgodigital.traineddata in de juiste directory:

```
$ sudo mv letsgodigital.traineddata /usr/share/tesseract-ocr/tessdata
```

Pydev

PyDev in Eclipse werkt momenteel niet op het Sogeti netwerk. Hoe je dat oplost vind je hier.

PyDev set-up

PyDev in Eclipse werkt op het Sogeti netwerk niet out-of-the-box. Om dit te fixen kun je de volgende stappen volgen!

- Stap 1: Open Eclipse
- Stap 2: Window -> Preferences -> General -> Network connections
- Stap 3: Bovenin zie je “Active Provider”. Verander deze naar “Direct”.

Dit hebben we gedaan op Eclipse IDE for Java Developers (Version: Neon.2 Release (4.6.2)).

Image Preprocessing

Input plaatjes worden voorbewerkt met als doel om het ruis eruit te halen. Hierdoor kan ReadImage de nummers in het plaatje beter herkennen. Hier wordt geillustreerd welke stappen het input plaatje doorloopt tijdens het voorbewerk proces. Let wel dat elke stap parameters bevat die getweaked kunnen worden, ten goede of ten slechte van de performance van ReadDisplay. De huidige selectie van preprocessors en parametersettings zijn geoptimaliseerd voor slechts 1 font.

Image Preprocessing

Input plaatjes worden gepreprocessed met als doel om het ruis eruit te halen. Hierdoor kan ReadImage de nummers in het plaatje beter herkennen.

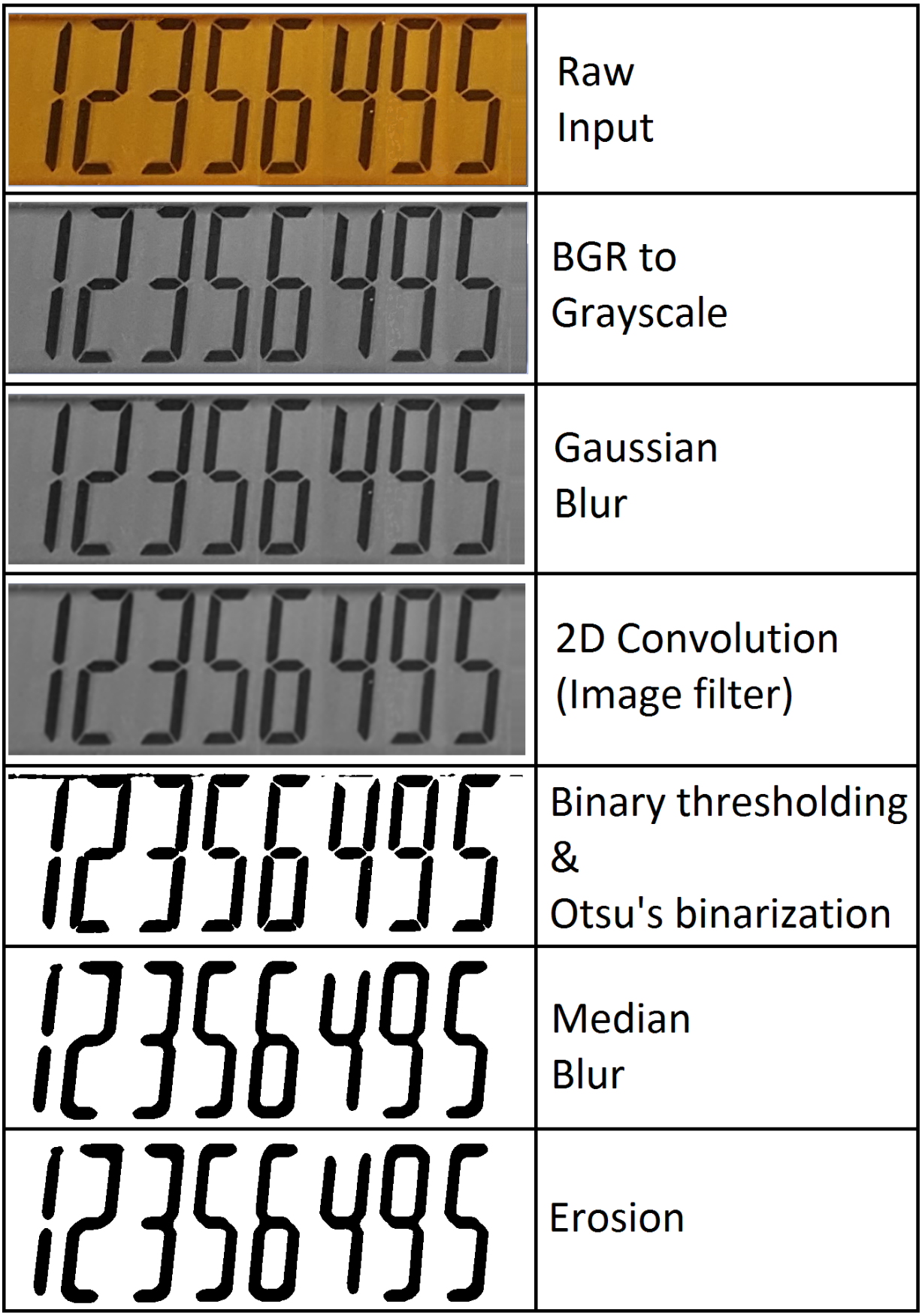


Fig. 18: De verschillende operaties die worden uitgevoerd op een input plaatje.

De blurs worden met name toegepast om de gaten tussen de segments op te vullen. Met binary thresholding heb je een threshold waarde; alle pixels onder deze waarde worden wit, alles daarboven worden zwart. Otsu's binarization helpt met het bepalen van die threshold waarde. De threshold waarde kan ook met de hand aangepast worden.

Het preprocessen van plaatjes gebeurt in de package Preprocessor (heette voorheen "Blur). Hierin zit de class Preprocess. Deze class bevat een functie preprocess_image (heette voorheen "blur"). Hierin gebeuren alle hierboven genoemde operaties op een input plaatje.

De effectiviteit van het preprocessen kan worden geevalueerd met het script in EvaluateReadDisplay.py (..RekenRobotSrcDisplayLezen). Deze script opent input-plaatjes uit een folder en haalt ze vervolgens door de Preprocessor en ReadImage heen. Testplaatjes kunnen worden gegenereerd met CombineImages.py (..RekenRobotSrcDisplayImageSplitterAndGenerator).

De volgende link bevat nadere uitleg en tutorials over het bewerken van plaatjes in OpenCV:

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_table_of_contents_imgproc/py_table_of_contents_imgproc.html#py-table-of-content-imgproc

Springplanning

Hier vind je de sprintplanningen.

Sprintplanning Week 1

Planning (Gemaakt op Ma 9 januari 2017)

Planning van de (korte) sprint

Wat?	Hoeveel feature points?
Notities (Sphinx/Rst) • Versiebeheer • Tooling leren	1.5
Unittesten testen • 1.5-2 A4 • Handleiding aanwezig? • Welke modules/units/testen zijn er? • Doen ze het?	3
Decoupling • Hoe goed? • Suggesties voor verbetering	3

Update RST docs

Geplande taken en duur:

Taak	Aantal keer	Aantal manuren
Notities van setup verwerken	3x	1.5 uur
RST inleren	3x	1.5 uur
Sphinx builden	3x	1.5 uur
PUSH/PULL onderling + Albert	3x	1.5 uur
Check documentatie (OK/Not OK)	1x	1.5 uur

Voortgang taken:

Taak	Andy	Pim	Christian
Notities van setup verwerken	x	x	x
RST inleren	x	x	x
Sphinx builden	x	x	x
PUSH/PULL onderling + Albert	x	x	x
Check documentatie (OK/Not OK)	x	x	x

(- = To Do, / = In progress, x = done)

Testen van de testen

Geplande taken en duur:

Taak	Aantal manuren
5x uitvoeren van de bestaande testen	3
Documentatie (1.5 A4)	1
5x uitvoeren van instructie tests	1
Review	0.5
Conclusie	0.5
CC berekenen	2
Aantal classes en functies tellen	1
Aantal UT en MT tellen	0.5

Voortgang taken:

Taak	Andy	Pim	Christian
5x uitvoeren van de bestaande testen	x	x	x
Documentatie (1.5 A4)	x	x	x
5x uitvoeren van instructie tests	x	x	x
Review	x	x	x
Conclusie	x	x	x
CC berekenen	x	x	x
Aantal classes en functies tellen	x	x	x
Aantal UT en MT tellen	x	x	x

(- = To Do, / = In progress, x = done)

Sprintplanning Week 2

Openstaande features

Wat?	Hoeveel feature points?
Kwaliteit verhogen ReadDisplay (High) • 1 font, 1 regel • 99% nauwkeurigheid	8
Kwaliteit verhogen ReadDisplay (Low) • 10 fonts, 1 regel • 99% nauwkeurigheid	4
Kwaliteit verhogen ReadDisplay • alle fonts, 1 regel • 99,9% nauwkeurigheid	8
Kwaliteit verhogen ReadDisplay • 1 font, 1+ regel • 99% nauwkeurigheid	5
readDisplay werkend op Raspberry Pi (Medium)	3

Sprintplanning

Kwaliteit verhogen ReadDisplay

Geplande taken en duur:

Taak	Aantal manuren
Uitbreiden van testdataset tot 1000	3
Optimaliseren dataset voor goede testresultaten	1
Aanpassen programma voor behalen goede kwaliteit	1

PlantUML

Met PlantUML kun je UML diagrammen genereren uit tekst.

Plant UML

PlantUML is een open-source tool dat UML diagrammen kan maken vanuit text. Met PlantUML in combinate met sphinx is het mogelijk om makkelijk UML diagrammen te verwerken binnen de documentatie.

Installatie

Conf.py is goed ingesteld om Plant UML direct te compileren, toch zijn er een aantal, eenmalige, stappen nodig om alles te installeren.

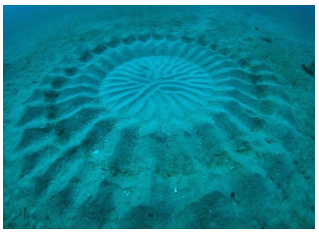
Toevoegen plantuml extensie bij Sphinx; In cmd het volgend command:

```
> pip install sphinxcontrib-plantuml
```

Installeren plantuml:

- Download plant uml via http://plantuml.com/download

3.1.3 SeaBroomPhi



date April 6th 2017

Welkom op de team-pagina van SeaBroomPhi.

Algemeen

Hier vind je algemene notities van het team. Het eerste stuk gaat over version control. Vervolgens beschrijven we de bevindingen van ons onderzoek naar decoupling van bestaande code en unit-testen van de pilotgroepen.

In de MOP gallerij tref je foto's aan die gemaakt zijn van aantekeningen op het bord tijdens de (sprintwissel) sessies.

Manual: From installation to demo

date 14 april 2017

This document contains a manual to install the Rekenrobot on a new Raspberri pi and to run a first Demo of the work from previous teams.

Tools

Verschillende tools worden gebruikt voor verschillende acties. Om te beginnen gebruik je Putty om een verbinding te maken met een RaspberryPi. Zo kan je commands uitvoeren op de RaspberryPi via je laptop. Ook kan je er voor kiezen om de interface van de RaspberryPi op je laptop te draaien, dit kan via VNC. Op deze manier hoef je niet de command line te volgen, en is het dus makkelijker om cammands uit te voeren. Gebruik deze tool aleen als dit nodig is.

Om file te delen met elkaar worden drie tools gebruikt: Bitbucket, HG Tortoise en Sphinx. Om van de main server een fork te maken, wordt bitbucket gebruikt. Om

van deze fork een clone of pull te maken wordt HG Tortoise gebruikt, en om van deze clone of pull een eigen working copy te maken wordt sphinx gebruikt.

De installatie van Bitbucket, HG Tortoise, Sphinx en VNC is goed uitgelegd via de pagina van NautulusPi. Voor de Instalatie van Putty zal op deze teampagina een instructie komen. *under construction*

teams/2017.2_SeaBroomPhi/IMG_3523.jpg

Run a demo

After proper installation of the tools and RekenRobot scripts on the Raspberri, a demo can be run as following.

- Open VNC to work from the Raspberri pi
- Open the terminal, enter the following commands:

```
$ source ~/.profile
↪     # allways update file for virtual
↪environment when opening terminal

$ workon rr
↪         # allways work in the virtual
↪environment when running a script

# for now the software only works when
↪it is located in the correct directory!
$ cd ~/pathways-extensions-
↪training/RekenRobot/Src/DisplayLezen/
↪Demo          # change to the
↪right directory of the python script

$ python MainfileReadimage.py
↪       # run the python script you like

# you might need to (re)install some
↪modules in the virtual environment,
↪ follow the onscreen instructions
↪(pillow, pytesseract and imutils)
```

rst plugin

Mop Gallerij

In deze pagina vind je algemene notities van het team SeaBroomPhi die gemaakt zijn van aantekeningen op het bord tijdens de (sprintwissel) sessies.

Sprint sessie 1

date woensdag 12 april 2017

teams/2017.2_SeaBroomPhi/IMG_3521.jpg

Fig. 19: Features uit sprint sessie 1.

date donderdag 13 april 2017

teams/2017.2_SeaBroomPhi/IMG_3524.jpg

Fig. 20: Overzicht van het systeem en waarop alle programmas werken.

teams/2017.2_SeaBroomPhi/IMG_3523.jpg

Fig. 21: Visueel plaatje van hoe de RaspberryPi via Putty connect.

Sprint sessie 2

date dinsdag 18 april 2017

Test Results

This section contains the results of the test scripts developed by Nautules Pi. The section below describes how the test results were obtained.

pytest	Runnable	Run	Result (x out of 5 times)
TestFixedDetect.py (from TestImageCropper)	yes	5	Passed (5/5)
TestPreprocessor.py (from TestImageCropper)	yes	5	Passed (5/5)
TestAutoDetect.py (from TestImageCropper)	yes	5	Failed (4/5)
TestManualDetect.py (from TestImageCropper)	yes	5	Passed (5/5)
TestPreProcessor.py	yes	5	Passed (5/5)
TestReadDisplay.py	yes	5	Failed (5/5)
test_AutoCrop.py	No: error	5	n.v.t.

Run a singel test

```
$ pip install pytest
```

This will install the pytest libraries. Now you can run your test:

Fig. 22: Features uit sprint sessie 2.

Fig. 23: Aantekeningen, DisplayLezen module uit meerdere modules.

```
$ cd path/to/project
$ pytest 'filename.py'
```

Repeat a test

pip install pytest repeat

Use the –count command line option to specify how many times you want your test, or tests, to be run:

```
$ py.test -
↪-count = 'choose a number' test_file.py
```

Each test collected by py.test will be run count times.

Run all tests in once

```
$ pip install nose
```

This will install the nose libraries, as well as the nosetests script, which you can use to automatically discover and run tests. Now you can run all tests in a specified folder in once, using only one command:

```
$ cd path/to/project
$ nosetests -vv
```

Let's meet SeaBroomPhi

Help

In case of emergency, call hackerman..

Fig. 24: Aantekeningen, voorbeel test script.

teams/2017.2_SeaBroomPhi/IMG_3557.jpg

Fig. 25: Nieuwe Index voor Pathways.

Using RST

http://www.sphinx-doc.org/en/stable/index.html

3.1.4 DuoPInotti

date June 8th 2017

Welkom op de team-pagina van DuoPInotti. Better goed gejat dan slecht bedacht.

Algemeen

Hier vind je algemene notities van het team.

Sprints

In deze sectie zijn alle sprints en notities opgenomen.

Taken voor Sprint 1

1fp = 1 werkdag

ToDo:

-

InProgress:

-

Done:

Demo Pi (1fp)

- (her)installatie Rasberry Pi (2x)
- packages installaties zoals omschreven in de overdracht
- code clone op Pi
- verbeteren demo

- aanpassen documentatie

Demo Docs + installatie tools (1fp)

- **Sphinx**
  - installatie
  - run
  - aanmaken teampage
  - leren RST
- **Mercurial**
  - Installatie
  - clonen

Beschrijving Demo 'dicht laptop' (1fp)

- **Opstelling nabouwen**
  - Foto maken
- **beschrijving demo, met daarin aandacht voor:**
  - noodzaak 're-manualCrop'
  - tips + trucks opstelling

ReManualCrop (RMC) (2fp)

- Inlezen code
- Redesign code
- implement code
- test code
- debug code
- demo code
- document code + demo

Bij werken document 'handy Rules' (2fp)

- **installatie op 'schone laptop'**
  - requirements (qua versies)
  - redenatie achter programma's en plugins
  - mogelijke fouten

Taken voor Sprint 2

1fp = 1 werkdag

ToDo:

Verbeteren OCR (3fp)

- Inlezen hoe de OCR werkt
- Bestaande werking testen
- Verbetering OCR ontwerpen
- Programmeren
- Testen
- Documenteren

InProgress:

Normalizeren plaatjes (3fp)

- Inlezen hoe de normalizering werkt
- Controleren hoe de normalizering aan te passen is
- Ontwerpen hoe de normalizering te verbeteren is zou moeten gaan (ook procedure)
- Programmeren
- Testen
- Documenteren

Done:

Afmaken Documentatie (1fp)

- Keuzes toelichten in van de demo
- Doctree updaten
- Op de voorgeschreven manier bouwen.

180 graden draaien van plaatje (2fp)

- Inlezen hoe de draaiing nu werkt
- Ontwerpen hoe de draaing zou moeten gaan (ook procedure)
- Programmeren
- Testen
- Documenteren

Pull request (1fp)

- **Schone forks maken**
  - Nieuwe clones maken
- Werk herstellen
- Pullen van elkaar
- Reviewen

Taken voor Sprint 3

1fp = 1 werkdag

ToDo:

Verbeteren OCR

- Inlezen hoe de OCR werkt
- Bestaande werking testen
- Verbetering OCR ontwerpen
- Programmeren
- Testen
- Documenteren

InProgress:

Done:

Pull request (1fp)

- Werk herstellen
- Pullen van elkaar
- Reviewen

180 graden draaien van plaatje - de luxe versie (klikken)

- Inlezen hoe de draaiing nu werkt
- Ontwerpen hoe de draaing zou moeten gaan (ook procedure)
- Programmeren
- Testen
- Documenteren

Normalizeren plaatjes

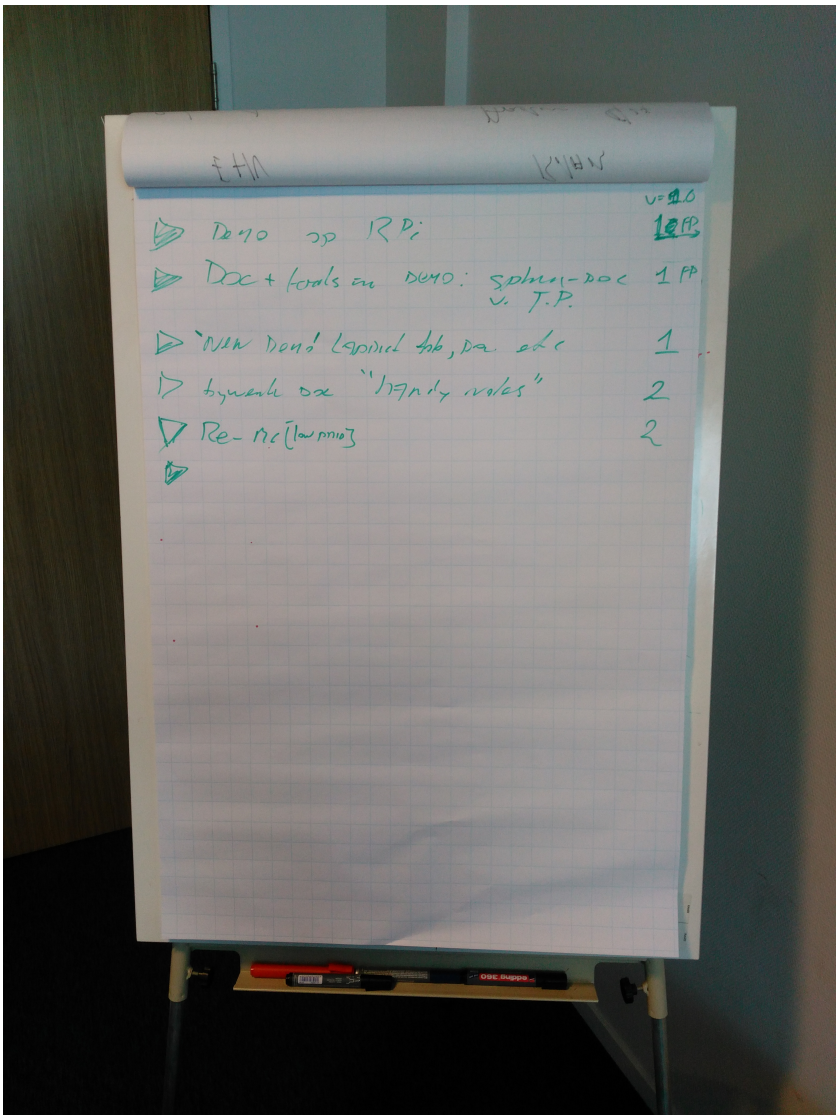
- Inlezen hoe de normalizering werkt
- Controleren hoe de normalizering aan te passen is
- Ontwerpen hoe de normalizering te verbeteren is zou moeten gaan (ook procedure)
- Programmeren
- Testen
- Documenteren

Mop Gallerij

Een overzicht van alle bord foto's gemaakt tijdens de sprintwisselingen door DuoPInotti.

Sprint sessie 1

date woensdag 9 juni 2017

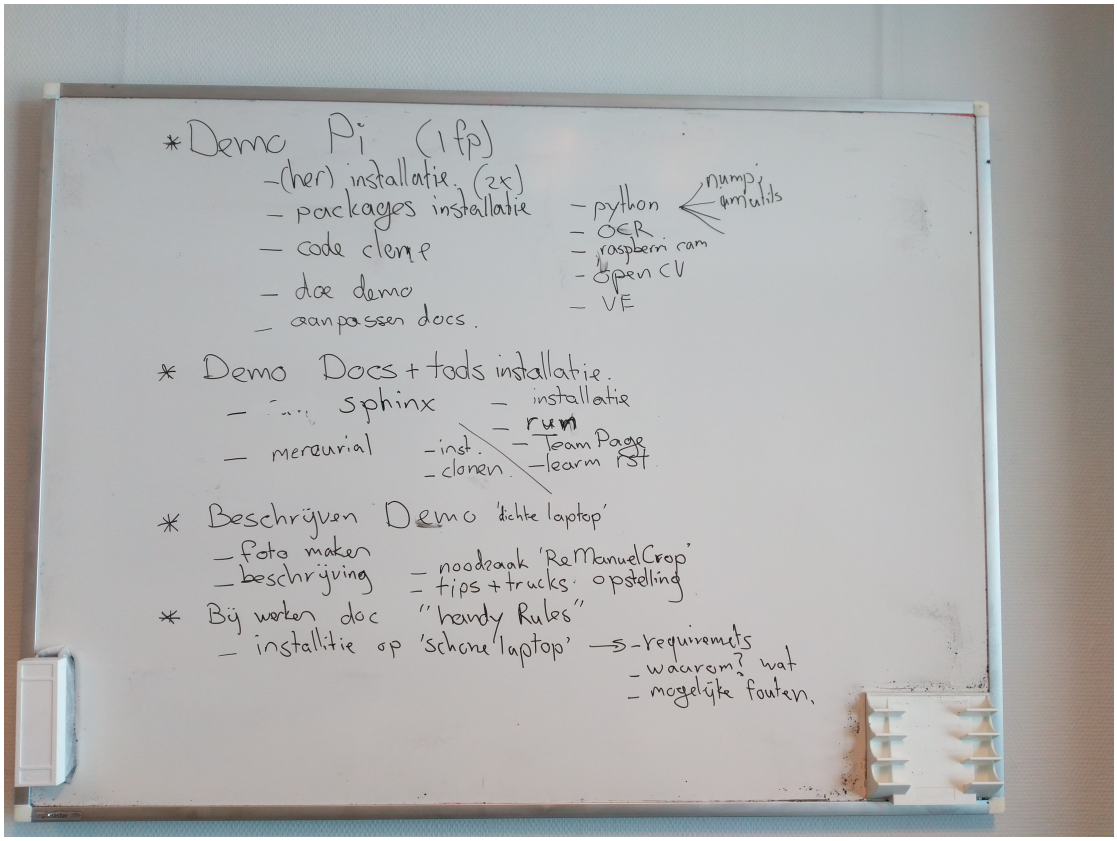


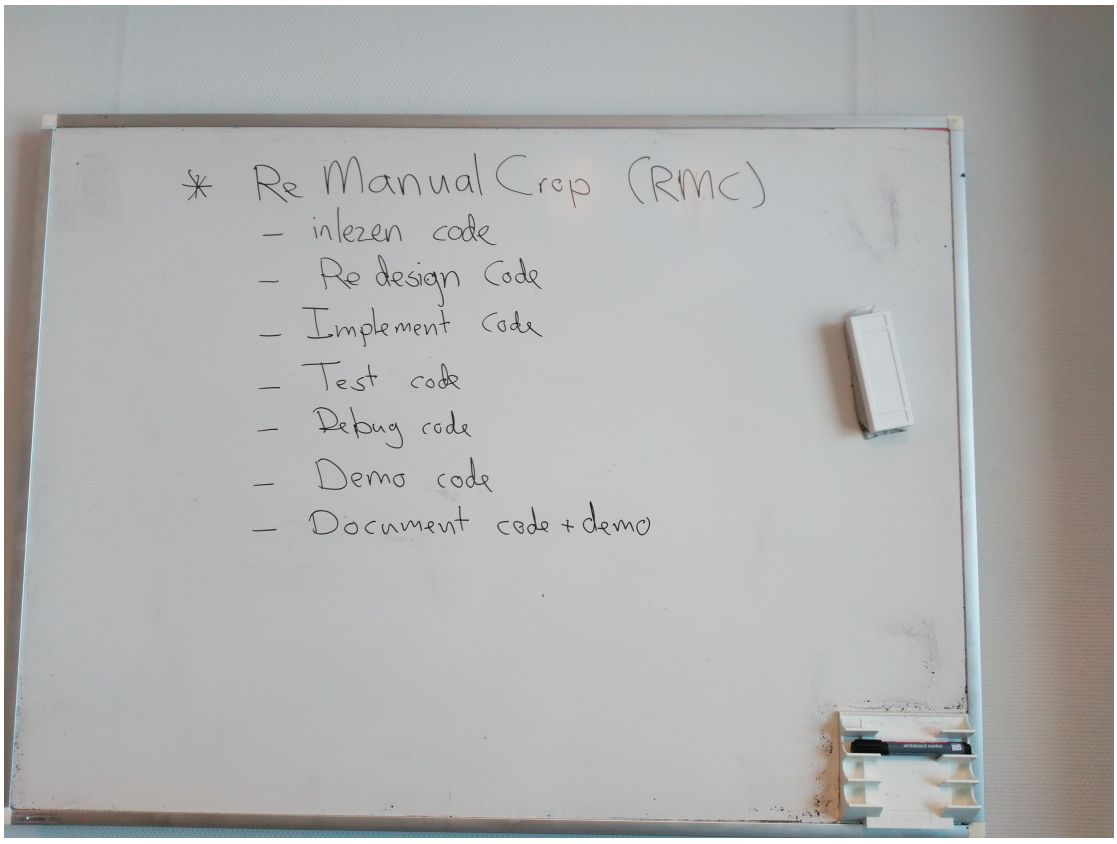
Sprint sessie 2

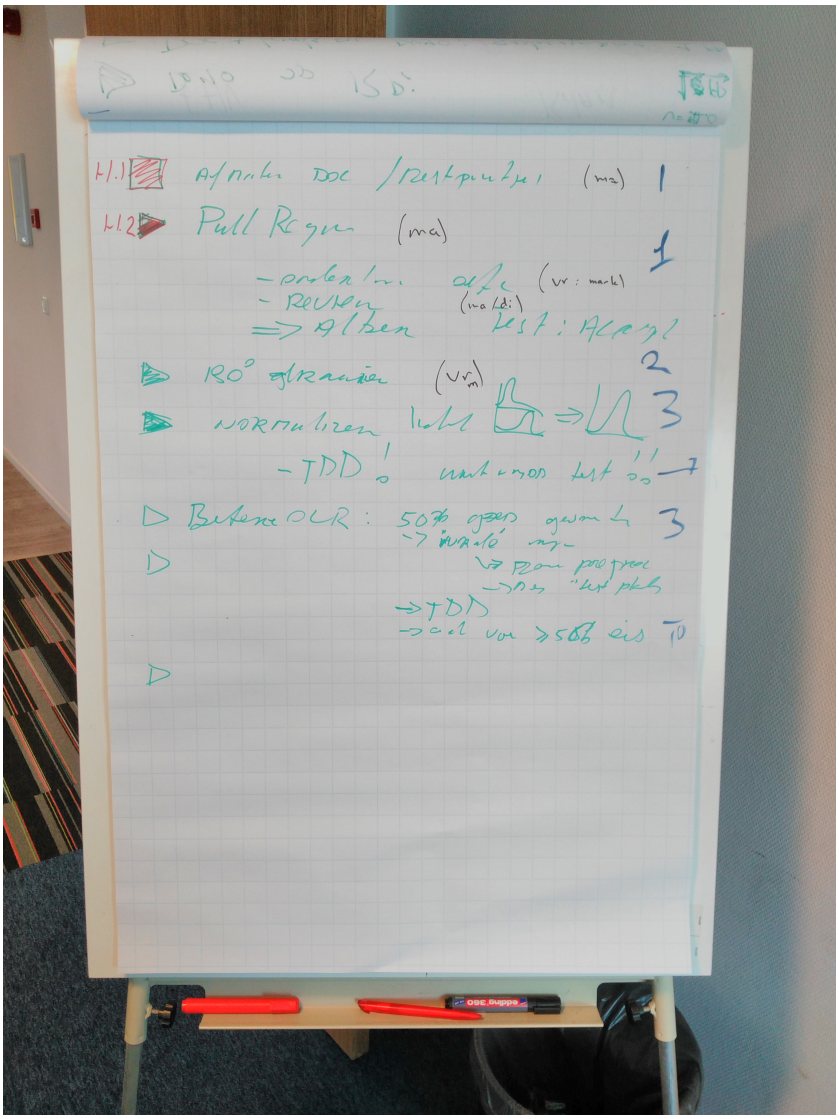
Demo: Project documentation bouwen

datum 12/06/2017

Deze pagina bevat de instructie om de documentatie voor dit project op te bouwen. Het is in de vorm van een demonstratie zodat je kan laten zien hoe je de documentatie opbouwt. Aan het einde van de (korte) demonstratie kunt u een RekenRobot-teampage maken en aanpassen, met behulp van Sphinx.


* Demo Pi (1 fp)
-(her) installatie. (2x)
- packages installatie
- code clone
- doe demo
- aanpassen docs.
- python
- OCR
- raspberri cam
- Open CV
- VE
numpy
matplotlib
* Demo Docs + tools installatie.
- ... sphinx
- mercurial
-inst.
-clonen
- installatie
- rum
- Team Page
-learn rst
* Beschrijven Demo 'dichte laptop'
- foto maken
- beschrijving
- noodzaak 'ReManuelCrop'
- tips + trucks: opstelling
* Bij werken doc "handy Rules"
- installitie op 'schone laptop' -> -requirements
- waarom? wat
- mogelijke fouten.


* ReManualCrop (RMC)
- inlezen code
- Re design Code
- Implement code
- Test code
- Debug code
- Demo code
- Document code + demo



Hulpmiddelen

Een beschrijving van de plugins, en de installatie daarvan, die je nodig hebt voor het maken van de documentatie vind je onder *Version Control*, *Sphinx-doc* en *Setup on laptop* van de training pagina.

In het kort heb je voor deze demo de volgende installaties nodig:

- Python 3.6 (64-bits)
- Sphinx
- Mercurial 4.2.1 (64-bits)
- Een clone van de repository

Op de bovenstaande instructiepagina vind ook een globale uitleg hoe je de documentatie moet opbouwen. Deze instructie geeft specifiek aan hoe je de teampagina aanmaakt en wijzigingen doet.

Opzet demo

Na het opzetten van de demo kan je praktisch alles wat je tijdens de demo wilt laten zien. Pas na vaker gebruik krijg je ook handigheid in het opmaken via rst.

Stap 1 - bestaande documentatie op te bouwen met Sphinx

Zorg dat je een clone van de repository lokaal op je PC hebt staan. Alle files voor de documentatie staat in de map .pathways-extensions-trainingdocs. Voer het onderstaande commando uit in cmd om de pagina te bouwen:

```
sphinx-build -c doc -b html  -d tmp/sphinx_cache  doc/ __result/html
```

Dit zorgt ervoor dat er een mapje __results in de map doc komt met daarin het bestand index.html. Als je hier dubbel op klikt zou de Read the Docs pagina lokaal moeten openen.

Stap 2 - eigen teampagina aan te maken en bouwen

Ga naar de map pathways-extensions-trainingdocsdocteams en maak een nieuwe map aan, conform het template jaaral.volgnummer_teamnaam. Kopieer het index.rst bestand van een vorig team naar de map. Pas de inhoud van dit bestand aan voor jullie team en sla het bestand op. Bouw de pagina op met het Sphinx commando wat in de readme staat. Door het Sphinx commando te draaien wordt de map __results in de docs map gecreerd. In deze map staan alle html bestanden die zijn opgebouwd uit de rst code. Vind het bestand wat je gemaakt hebt en je zult zien dat jullie teampagina erbij staat. Eigenlijk heb je nu al het eerste gedeelte van de demo laten zien.

Demo

Elke Teampagina heeft een MopGallerij! Maak een nieuw bestand aan in de teammap en noem die MopGallerij.rst. Pas ook de index.rst aan door ‘MopGallerij’ onder te toctree te zetten run het build commando. De demo is geslaagd als de pagina lokaal geopend kan worden. De pagina kan je openen door een ()

Demo: DisplayLezer op de Laptop, verbeterde versie

Tijdens onze testen zijn we op een aardige methode gekomen om de displayLezer functionaliteit te testen. Hieronder een beschrijving van hoe we dat gedaan hebben.

De opstelling

Door de rekenmachine neer te leggen om het muispad en de laptop half dicht te doen, ligt de rekenmachine op een stabiele ondergrond en wijst de webcam van de laptop richting de rekenmachine. De noodzaak is wel dat het beeldscherm is gedubliceerd naar een extern beeldscherm anders kan je niet ziet wat je doet om de programmateur te starten en de manualCrop uit te voeren.

Fig. 26: De opstelling zoals hierboven beschreven maar een plaatje zegt (ongeveer) een duizend woorden. Zorg er voor dat er voldoende daglicht in de ruimte is waar je je bevindt. TL verlichting kan te fel zijn, dus zet dit uit als dit het geval is.

De voordelen

- Stabiele rekenmachine positie ten opzichte van de webcam. Dus geen onhandig gedoe met de rekenmachine stil proberen te houden voor de webcam.
- Geen overige materialen nodig anders dan laptop en rekenmachine (op een extra scherm na dan).

De nadelen

- De lichtinval en intensiteit is nogal variabel door dat de rekenmachine in de schaduw ligt van het laptopscherm.
- Doordat de laptop half dicht is, is de besturing van de laptop niet mogelijk, tenzij de laptop op een extern scherm is aangesloten met extra muis en toetsenbord.
- Besturing van de rekenmachine is bemoeilijkt door dat het laptop scherm er boven zit.

Voortschrijnend inzicht

In de opstelling beschreven hierboven is er behoefte ontstaan voor een nieuwe feature. Wij maakte namelijk veel gebruik van de ManualCrop feature omdat die zeer stabiel is als de opstelling niet veranderd. Echter bij een her-invoer van een getal in de rekenmachine kan deze verschoven zijn dus moet er opnieuwe geCropt worden. Om dit process te versoepelen moet er een ReManualCrop komen die te activeren is door (bijvoorbeeld) een toetsindruk.

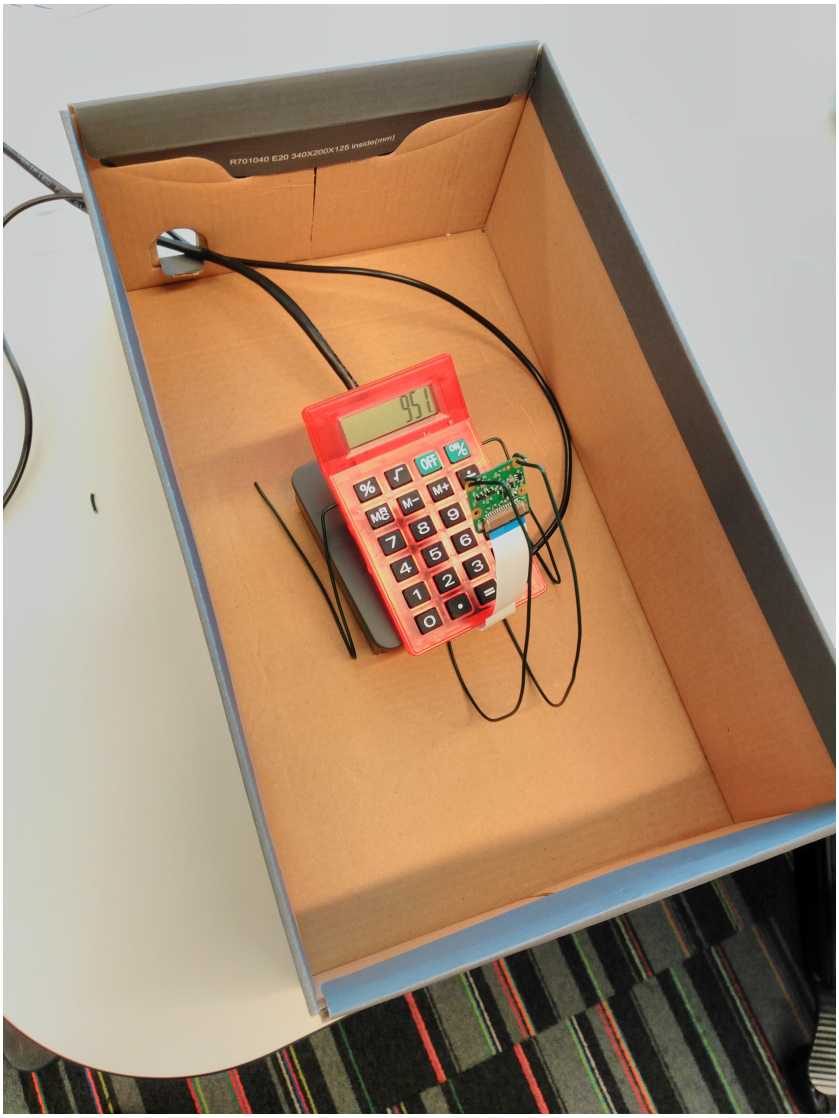
Demo: DisplayLezer op de RaspberryPi

Tijdens onze testen de testmethode om de displayLezer functionaliteit te testen op een RaspberryPi verbeterd. Hieronder een beschrijving van hoe we dat gedaan hebben. Deze demo kan alleen uitgevoerd worden als de RaspberryPi correct is geïnstalleerd en de rekenmachine repository er op staat. De installatie handleiding is te vinden op de *Raspberry Pi*.

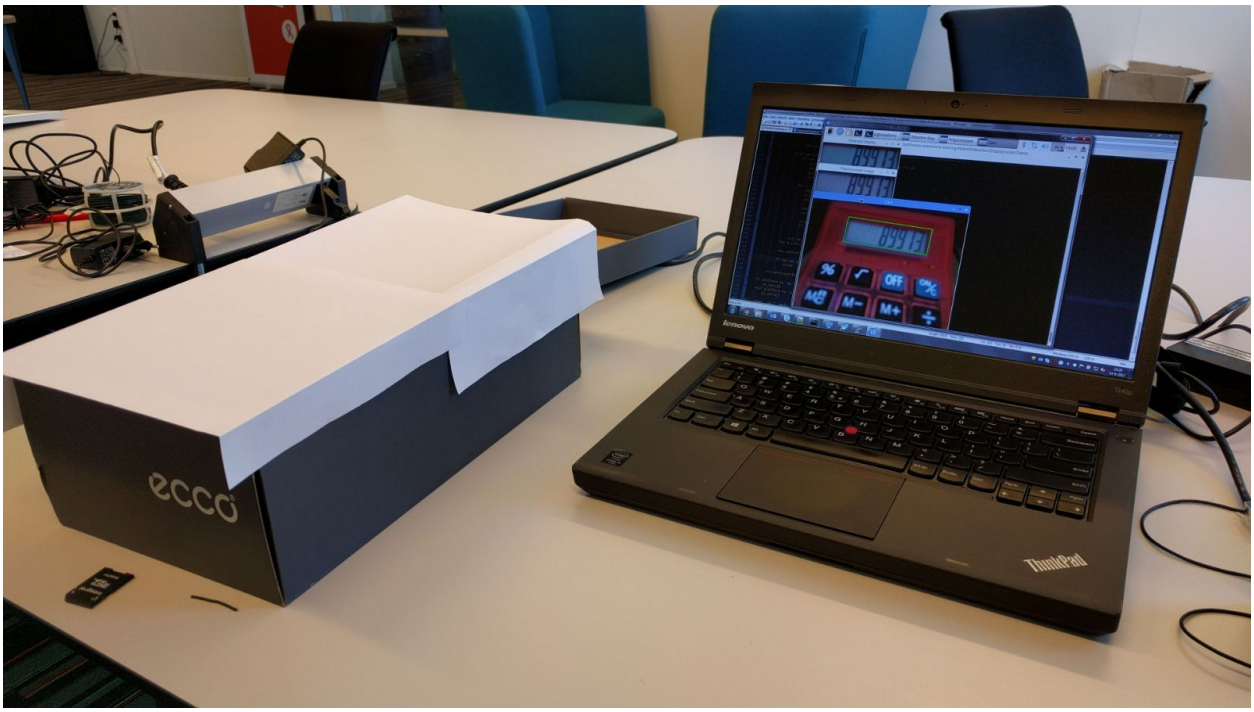
De opstelling

Met een simpel gemaakte opstelling m.b.v. ijzerdraadjes, witte A4tjes en een schoenendoos, hebben we een manier gevonden om externe factoren zoveel mogelijk uit te sluiten. Uitsluiten van externe factoren is nodig omdat we merkte dat zaken als schaduw en fel licht de uitkomsten beïnvloede van het testen. Na de installatie van de RaspberryPi kun je, bijvoorbeeld via VNC viewer, de projectbestanden clonen op de Pi. Om te zorgen dat de RaspberryPi camera de rekenmachine goed kan lezen moet je de volgende opstelling bouwen:

Om te de zorgen dat je de externe factoren moet je de bovenstaande opstelling in een (schoenen)doos zetten.


R701040 E20 340X200X125 inside(mm)

Door er wat wat A4tjes overheen te leggen zorg je er voor dat er difuus licht op de opstelling komt. Zorg er dus wel voor dat de doos in een goed verlichte ruimte staat! Veel TL-licht is prima, direct zonlicht is beter. De opstelling komt er dan uiteindelijk zo uit te zien:



De demo

Zet de rekenmachine aan en druk wat willekeurige nummers in. Door de onderstaande code uit te voeren op de Raspberry Pi, kun je de demo draaien.

```
$ source ~/.profile          # always update file for virtual environment when␣
↪opening terminal
$ workon rr                  # always work in the virtual environment when running a␣
↪script on the raspberryPi
$ cd ~/pathways-extensions-training/RekenRobot/Src/DisplayLezen/Demo         #␣
↪naviagate to the right directory
$ python 'namefile'.py       # run the script of the demo
```

Bij het draaien van de demo kun je een aantal keuzes maken:

- Versie van de demo –> Op het moment van schrijven werkt optie 3: ManualDetect het beste
- Manier van preprocessen –> Hiermee wordt het plaatje klaargemaakt voor de OCR. Kies optie 1 om een snelle demo te laten zien, kies andere opties om het verschil te laten zien (je kunt er meerdere tegelijk kiezen, gescheiden door een “, “).
- **Maak nu een vierhoek om het rekenmachine schermpje, het liefst net iets buiten de schermranden.**
  - Druk op “r” als de crop is mislukt om het croppen te herstarten
  - Druk op “c” om verder te gaan
  - Druk op “q” om de demo te stoppen.

- **De demo laat nu live zien welke getallen er in het venster van de rekenmachine staan.**
  - Druk op "r" om de crop te herstarten (omdat de rekenmachine scheef is gaan liggen)
  - Druk op "q" om de demo te stoppen.

De voordelen

- Stabiele rekenmachine positie ten opzichte van de Raspberry Camera. Dus geen onhandig gedoe met de rekenmachine en camera stil proberen te houden.
- Beeldkwaliteit van het display hoog. Schaduws een weerspiegelingen zijn zoveel mogelijk geminimaliseerd.

De nadelen

- De lichtinval en intensiteit is nogal variabel door dat de rekenmachine in de schaduw ligt van het laptopscherm.
- Voor de besturing van de rekenmachine de A4tjes verwijderd worden.

Voortschrijnend inzicht

Buiten sluiten van externe factoren kan veel beter! Denk aan het opstelling waarmee je helemaal niet meer afhankelijk bent van externe factoren. Dan kun je d.m.v. een goede lamp met difuus licht

Uiteindelijk zou het mooi zijn als de DisplayLezer in elke situatie kan lezen. Maar om goed te kunnen begrijpen wat de modules en programma's doen, is dit voor nu goede oplossing.

Code Standard

Attention: Dit document is een schets en zeer incompleet, en misschien wel fout.

Een van de belangrijkste aspected van goede bron code is het de leestbaarheid van de code. Om het even extreen te zeggen, Goede code heeft geen documentatie nodig. echter is goede code zeer moeilijk om te maken dus moet er goed gedocumenteerd worden.

Toch moet men, om de code aan te passen, zich in de code zelf verdiepen. Hierbij helpt het als deze goed geformarteerd is. Hieronder een lijst (met voorgestelde) afspraken om de code goed leesbaar te houden en maken.

- Geef variable begrijpelijke namen die een omschrijving geven van hun type data.
- Gebruikt de juist indentatie, al is dat verplicht voor Pythen dus niet zo probleem in dit project. Wel gebruik 4 spaties per indentatie en NOOIT tab sympolen (check je editor)
- Kluster het aanmaken van variable zo ver mogelijk bovenin de functie en/of class. Sorteer hierin ook in vergelijkbare type. (int, float, etc)
- Zet '=' tekens zo ver mogelijk op 1 lijn door gebruik van spaties. Dit is vriendelijk voor de oog en veel beter leesbaar. (en we schrijven niet meer in fortran67 met een regellengthe limit ;))

Comment stijl; hierin zijn er een paar richtlijnen. Terug grijpen op "goede code heeft geen documentatie nodig", denk als je een comment wilt toevoegen, "kan ik de code anders maken dat er geen comment nodig is?".

- Zo ja, doe dit en je code is (zeer waarschijnlijk) veel beter.

- Zo nee, zet je comment neer waarin je verteld niet alleen wat er gebeurt maar ook waarom. Vooral dit laatste is erg handig voor je opvolgers.

Normalisatie module

Tijdens het experimenteren met de Demo op de Raspberry Pi bleek het omgevingslicht veel invloed te hebben op de correctheid van de OCR-module. Het bleek dat veranderd omgevingslicht voor schaduwen en andere ruis te zorgen, zowel als verminderd contrast op het scherm. Dit was hardware-matig simpel op te lossen door de Raspberry Pi in een doos te plaatsen met witte wanden en LED-strips voor een homogeen lichtveld.

Bij de demo bleek dat de product owner graag de doos vervangen zien worden door een software oplossing. Deze feature zou dan onderdeel worden van de preproccesor als module genaamd 'normalisatie'.

Het onderzoek

De manier waarop de normalisatie gedaan moest worden was vooraf nog niet goed omgeschreven en wat de mogelijkheden zijn was ook niet duidelijk.

Om een overzicht te krijgen van wat normaliseren inhoud is er eerst gekeken naar het spectrum van de grijswaarden van een plaatje. Dit is een overzicht van de frequentie dat elke grijswaarde voorkomt.

Hieronder een overzicht van de onderzoek resultaten behaalt tijdens het experimenteren met de contrast in de plaatjes. Na een korte speurtocht leek een contrast verhogende functie uit de OpenCV bibliotheek een goede kandidaat hiervoor. Er zullen echter ook andere methoden zijn dan contrast verhogen, die hebben we niet bekeken.

Het onderzoek is opgebouwd uit de volgende stappen: Eerst de analyse van een enkele cijfer, in dit geval de nummer 3. Eerste een vergelijking tussen een gegenereerde 3 op witte achtergrond met behulp van cijfer generator en een gegenereerde 3 maar dan met een achtergrond vergelijkbaar met het display van de rekenmachine. Het resultaat is redelijke voorspelbaar en een leuke inleiding in het onderzoek.

Todo: Documentatie nummergenerator maken.

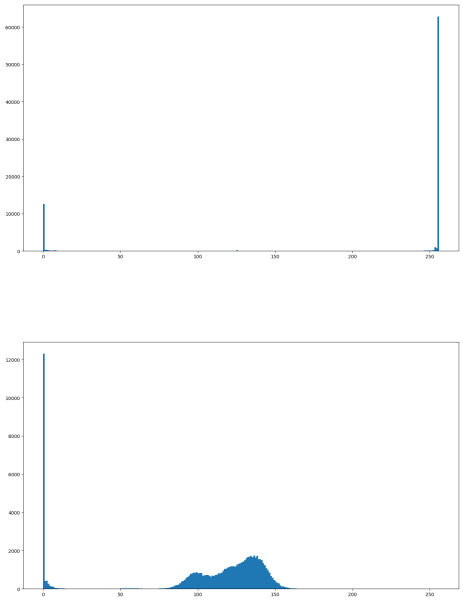
In het tweede deel zijn er 3 verschillende 3'en bekeken. De gegenereerde 3 op rekenmachine achtergrond, een foto van een 3 met goede verlichting condities en een foto van een 3 onder slechte verlichting.

In het derde deel is het onderzoek uitgebreid naar het gehele display. Hierin zijn meerdere versie bekeken. Het kleinst mogelijk getal qua segmenten en het grootste mogelijk getal qua aantal segment.

Introductie: Bronplaatjes

Dit zijn de twee plaatjes die zijn geanalyseerd.

Introductie: Grijswaarde vergelijking



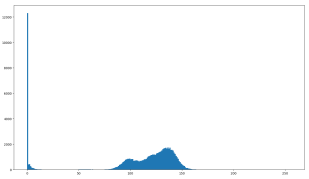
links de grijswaarde histogram van de 3 met witte achter grond en recht met solide zwart 3 met rekenmachine (korrelige) achtergrond. Hier het verschil tussen de voorgrond (de letter) en de achtgrond goed zichtbaar in beide plaatjes.

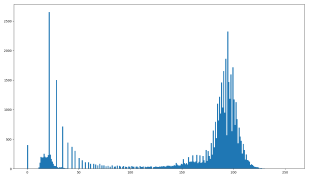
Introductie: Vergelijking met echte foto

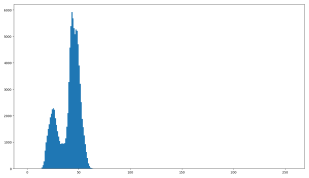
Drie varianten:

- links: rekenmachine achtergrond, ideaal zwarte karakter,
- midden: foto van rekenmachine onder goed verlichte omstandigheden,
- rechts: foto van rekenmachine in minder ideale verlichting dus zeer laag contrast en moeilijk leesbaar voor de mens.

Introductie: Histogrammen







Te zien is dat er zelfs bij weinig contrast nog steeds goed onderscheid gemaakt kan worden tussen voorgrond echt achtergrond pixels. Dit resulteert in het volgende plaatje:

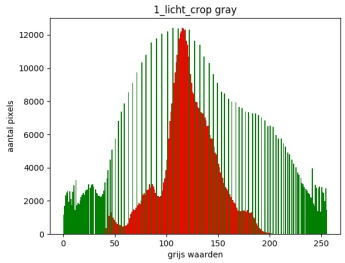
Gehele beeldscherm (enkel getal)

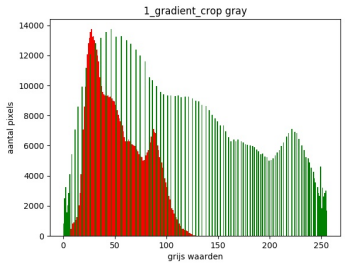
Uit de analyse hierboven bleek dat het contrast tussen de voorgrond (het cijfer) en de achtergrond met de computer goed te bepalen.

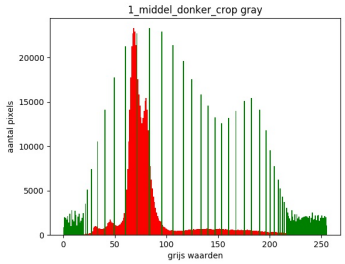
Uit de volgende analyse moet blijven of dat met het gehele display ook kan.

Het bekijken van een enkele cijfer is zoals hierboven is redelijk recht vooruit.

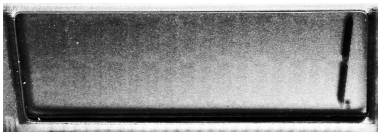
Nu kunnen we er de equalizeHist overheen halen van OpenCV. Dan veranderen de histogrammen naar dit:

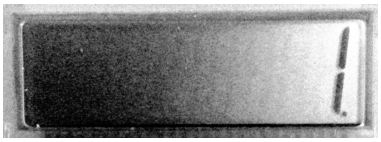


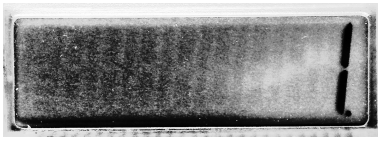




De bijbehorende plaatjes zijn dit:





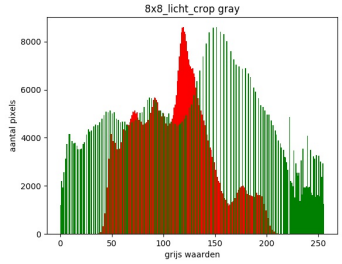


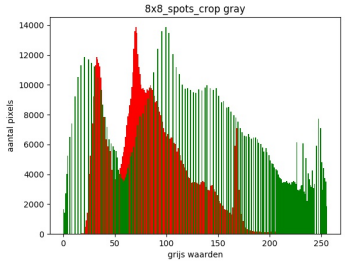
Uit deze figuren kunnen we halen dat het middelste plaatje onbruikbaar is zoals het nu is. Het zwart van het cijfer is niet anders dan het zwart links op het scherm.

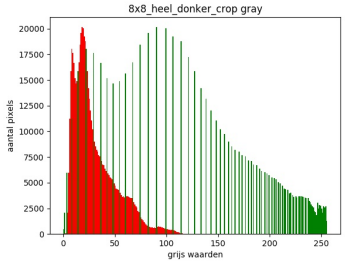
Gehele beeldscherm (meerdere getallen)

Het bekijken van een reeks 8 op het beeldscherm

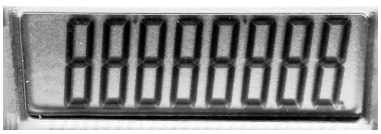
Nu kunnen we er de equalizeHist overheen halen van OpenCV. Dan veranderen de histogrammen naar dit:

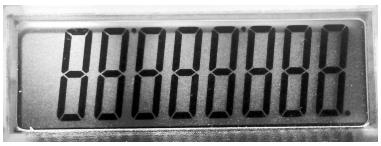


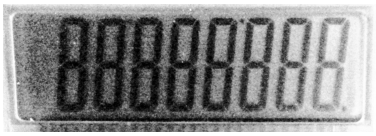




De bijbehorende plaatjes zijn dit:





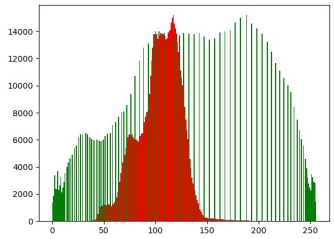


Plaatjes vanuit de doos

Een foto van het display in de box met homogene verlichting.

Tesseract: .3 1 1 5 . . . . 2214..21 1 11 53 19. . 15

Histogram van voor en na de ‘normalisatie’.



Het resultaat van de normalisatie:

Tesseract: 5 2 3 1 1 47 21 2 1 1 4 1

Het resultaat van een ‘goede’ threshold.

Tesseract: 2 7 4 3 33 545 2 9 1

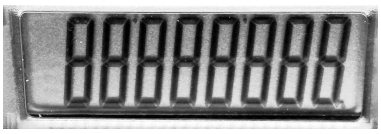
Het resultaat van een ‘goede’ threshold op een genormaliseerd beeld.

Tesseract: 1 4 7 3 5 5 2 2

Normalisatie vs Goede threshold functie

Input

Output van bovenstaande plaatje door de normalisatie module



Tesseract resultaat met normalisatie: 5 2 9

Output 'goede' threshold, nu nog met de handbepaald dus dit moet nog geautomatiseerd worden:

Tesseract resultaat met threshold: 33 5 7288 38873 3 3

Het 'goede threshold' plaatje met wat extra poetsen, dit was ook nog handmatig, De randen van de het scherm zijn verwijderd verder niets.

Tesseract resultaat met threshold (opgepoetst): 99989995

Conclusie

Op de manier hierboven beschreven kunnen we niet concluderen dat de geïmplementeerde normalisatie significant de uitkomst van de OCR beter maakt (of slechter). De output uit de OCR van de verschillende plaatjes, met of zonder normalisatie en met of zonder threshold geven allemaal verkeerde resultaten.

De moeilijkheid zit in het bepalen van een 'normaal' spectrum, dus welke vorm moet de histogram hebben na de normalisatie. Hierboven staan er meerdere spectra en geen van deze lijken transformeerbaar naar een 'normaal' spectra.

Een mogelijkheid voor een volgende groep is dit onderzoek op te pakken door de normalisatie te combineren met de andere bewerkingen die in de preproccesor zijn verwerkt.

Dan nog een opmerking om de normalisatie in het geheel te plaatsen. Het doel van de preproccesor is een input plaatje om te zetten naar een plaatje waarvan de OCR het slecht 1 op de 100 000 keren het foute antwoord leest. De normalisatie zou hierbij de variatie in lichtinval moeten compenseren.

Aanbevelingen

Iets waar we wel mee bezig zijn geweest maar verder niets concreets mee hebben bereikt is de analyse van de RGB kanalen apart. Een voorbeeld kan zijn dat alle puur rode pixels als achtergrond gezien kan worden omdat deze onderdeel uitmaken van plastic materiaal van de rekenmachine.

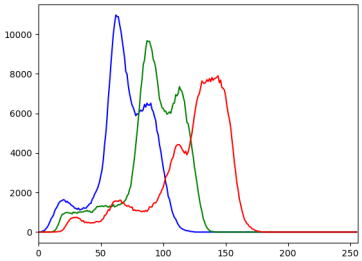
Een tweede toevoeging aan deze test zou zijn om nog te kijken naar andere methode om te normaliseren. Denk hierbij aan het platslaan van de grafiek (ipv witwaardes toevoegen), of andere manieren om te egaliseren. OpenCV heeft ook nog een andere veelbelovende functie om te 'normaliseren', namelijk CLAHE. Hiervan staat een test scriptje in de TryOut map, compare_equalize.py, maar er was tijdens dit project niet genoeg tijd om hier verdere tests mee te doen.

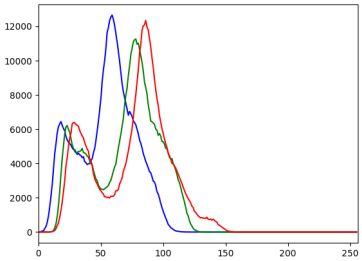
Vergelijken plaatjes

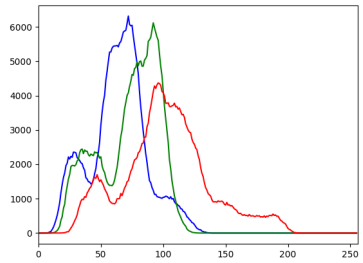
Bij het bekijken van de beelden kwamen we er achter dat er een verschil zit tussen de zijkanten van een plaatje en het middelste gedeelte. Een manier te vinden hoe we de plaatjes beter konden normaliseren, hebben we geprobeerd om de plaatjes op te delen in drie delen. Het grote midden stuk, en de twee zijkanten.

Resultaten

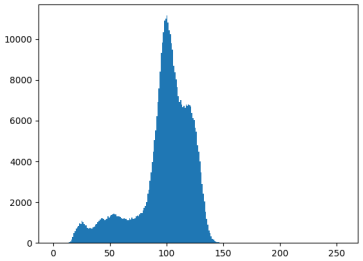
In kleur

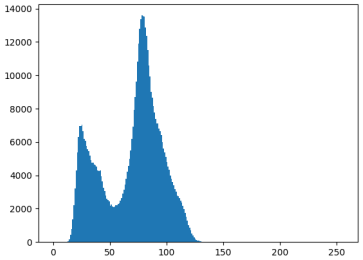


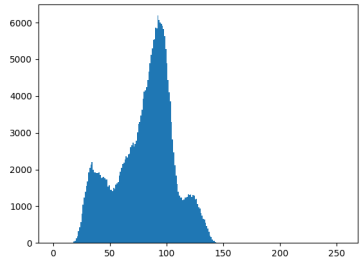




Grijs







ManualCrop - *Improved*

Introductie

Bij het begin van ons project, waren er enkele grote bugs in de manuele crop functie:

- Als de rekenmachine op zijn kop voor de camera lag, werd het beeld niet 180 graden gedraait na het croppen;

- Bij een hoek van 90 graden, werd het gekropte beeld een dun lijntje, wat niet meer leesbaar was voor mens en computer;
- Bij hoeken tussen de 0 en 90 graden, misvormde het gecropte plaatje buiten de grenzen van de gekozen vlakte.

Doel

Er was duidelijk iets gaande met de draaiing tijdens het manueel croppen. Het doel was om de bovenstaande problemen op te lossen, met een vernieuwde manuele crop functionaliteit. Hierbij moest met name de bovenstaande punten 1 en 2 worden opgelost.

Aanpak

De aanpak vroeg om een fikse aanpassing van de huidige functie.

- Door de klikactie van de gebruiker wilde we dat het demo programma wist wat de boven- en onderkant was van het gecropte plaatje, ongeacht de hoek.
- Door het programma zelf de vierhoek te laten berekenen en maken, krijg je geen vervormingen meer in de crop. Als de crop niet goed is, kun je opnieuw croppen door op 'r' te drukken.
- Het volledige plaatje (geschoten door de webcam/PiCamera) moet draaien om het middelpunt van het te croppen deel, en dan pas croppen. Dit zorgt ervoor dat de pixels recht worden gecropt.

Met deze ideeën in het achterhoofd zijn de testen aangepast (TDD) en vervolgens is de code aangepast.

Testen

Alle testen uit het testbestand TestManualDetect.py zijn aangescherpt aan de hand van de nieuwe criteria, en vervolgens geslaag

- Te weinig muisklikken (manuel_detect functie)
- Te veel muisklikken (click_and_detect functie)
- Een vierhoek tekenen buiten het gebied van het plaatje (dit is nu mogelijk omdat punt 3 en 4 uitgerekend worden aan de hand van een muisklik binnen het plaatje)

Conclusie

De feature is succesvol aangepast en getest. Er hoeft nu nog maar drie keer geklikt te worden door de gebruiker: In de bovenhoeken en op de onderrand (maakt niet uit waar) van het rekenmachinedisplay, zo weet het programma waar de onderkant van het display is. Dit werkt bij 45, 90, 180 graden, en alles wat daar tussen zit. Als een vierhoek buiten de grenzen van het plaatje is, zegt het programma dat de selectie niet goed is.

Aanbevelingen

Er moet nog gekeken worden naar de implementatie van de automatische crop functie. Die werkt nu niet meer altijd. Dit heeft te maken met de berekening van de constanten (b).

Tips en aanbevelingen voor opvolgers

Deze pagina bevat de aanbevelingen voor toekomstige deelnemers aan de basis software opleiding.

OCR

In de zoektocht naar betere resultaten zijn we aan het einde van ons project gestuit op een wellicht betere OCR dan die we nu gebruiken. Nu wordt er gebruik gemaakt van tesseract, maar is een Python binding die gebruik kan worden, zodat tesseract beter werkt. Deze heet PyOCR en gemakkelijk te installeren via pip install. Een probeersel is gemaakt ter vervanging van het huidige Readimage.py programma. De resultaten zien er hoopvol uit, maar er is niet genoeg mee getest om te kunnen stellen dat deze beter is. Het bestand om te proberen is te vinden in de map `.\\RekenRobot/TryOut/Python/ReadImagePyOCR.py`. Voor meer informatie over OCR kijk op de Pythontips pagina.

Version Control

Na het installeren van Mercurial kun je in het bestand .hghgrc de adressen neerzetten van de forks die jullie gebruiken. Onder het kopje paths kun je de onderstaande regel toevoegen om het pushen en pullen makkelijker te maken.

```
[naam] = [bitbucket link]

#bijvoorbeeld:
guus = https://GuusvanBohemen@bitbucket.org/GuusvanBohemen/pathways-extensions-
↪training
```

Als je nu een hg commando geeft naar een bepaalde link, kun je de naam gebruiken, ipv de link te kopiëren.

Documentatie

RST

Beter slecht gedocumenteerd als niet gedocumenteerd. Het is makkelijker om snel iets te typen en later weer weg te gooien dan dat je het uiteindelijke nergens hebt staan. Neem even de tijd om jezelf bekend te maken met de .rst syntax. Een aantal handige functies zijn:

```
'.. todo::'                 - Om automatische todo's op te geven, die zich op de todo␣
↪pagina verzamelen.
'.. code-block::'           - Om code op een nette manier aan te geven.
'.. figure:: [path\figure]' - Om een plaatje te laten zien (werkt ook met "image",␣
↪maar heeft dan weer andere functies).
':doc: [path\document]'     - Om te verwijzen naar een intern document.
```

Let ook goed op inspringingen en enters bij het aanroepen van functies!

Overige

Maak .bat bestanden aan met specifieke cmd regels, zet die op een logische plek, en voeg deze locatie to aan je path (Environmer

- **Navigeren naar de demo map: Het commando 'cddemo' navigeerde meteen naar de demo map van dit project. Het .**
  cd /d D:\Users\%username%\Documents\pathways-extensions-training\RekenRobot\Src\DisplayLezen\Demo

- **Bouwen van de documentatie: Het commando 'sphinx' bouwde automatisch de documentatie op. Het .bat bestand 's**
  sphinx-build -c doc -b html -d tmp/sphinx_cache doc/ __result/html
- **Open van de teampagina: Het commando 'team' opende direct onze teampagina in IE. Het .bat bestand 'team.bat' h**
  start iexplore

  D:\Users\gbohemen\Documents\pathways-extensions-training\docs__result\html\teams\2017.3_DuoPiNotti\index.htm

Let's meet DuoPInotti

He's an industial designer and the other is a thermo nuclear physicist. Both are here to save the world.

Help

Bill Nye tries to save the world, so do we. One line of code at the time.

Our method

programming is easy with the correct tools. So do not kill a fly with a flamethrower or sink a submarine with a bathduck.

So our keyboard:

3.1.5 CherryPi

date
August
7th
2017

Welcome to the team-page of CherryPi.

General

Here you'll find the general notes of the team.

Sprints

In deze sectie zijn alle sprints en notities opgenomen.

Tasks for Sprint 1

ToDo:

F1 - Project documentation (½fp)

- Finish team page
- Document installation bugs
- Create a sprint page

F2 - Demo Pi (½fp)

- Finish installation on Raspberry Pi
- Create a back-up image of each Pi.
- Ensure the Raspberry remotely accessible
- Test the new manual to see if it covers all bugs in the manual

F3 - Report found bugs (½fp)

- Create report for bug in AutoDetect
- Create bug report for FixedDetect

F4 - Fix bugs (3fp)

- Learn about pytest and writing test scripts
- **Fix bug #2**
  - Update test readdisplay
  - Discuss with Albert what to do with remaining bugs
- Fix bug #3
- Fix bug #4

F5 - Test current effectiveness preproces

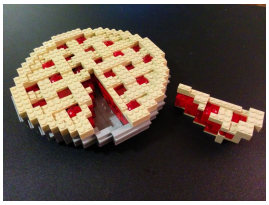
- Create different test setups
- Get more test images
- Test effectiveness without preprocessor with 10 images
- Test effectiveness with preprocessor with 10 images
- **Write test scripts to test the effectiveness**
  - preprocessor
  - OCR
  - preprocessor + OCR
- Write test report on process and findings after testing on more images
- Run the test on Pi

F6 - Improve current preprocessor OCR

- Describe possible improvements

- Design improved preprocessor
- Test improved preprocessor
- Write test report on process and findings
- Run the test on Pi

Tasks for Sprint 2



ToDo:

F1 - Improve the ImageGenerator (4fp)

- Make ImageGenerator can select the number of digits we want
- Include more backgrounds
- Make it possible to add noise and deform
- Include more backgrounds

F2 - Complete the tests of the whole system including preprocessor and OCR (3fp)

- Test the random factor which effects the accuracy and write the report
- Test all individual digits and write the report
- Test a set of combination of digits and write the report
- Draw a graph showing the test result

F3 - Set up and show the demo on the Pi (2fp)

F4 - Make pull request of Sprint 1 (1fp)

- Clean up
- Review the pull request

F5 - Make pull request of Sprint 2 (1fp)

- Clean up
- Review the pull request

F6 - Research and improve the preprocessor (4fp)

- Design improved preprocessor
- Test improved preprocessor
- Write test report on process and findings
- Run the test on Pi

Mop Gallery

This page shows all the pictures taken of the board notes.

Session 1

date Thursday August 3rd 2017

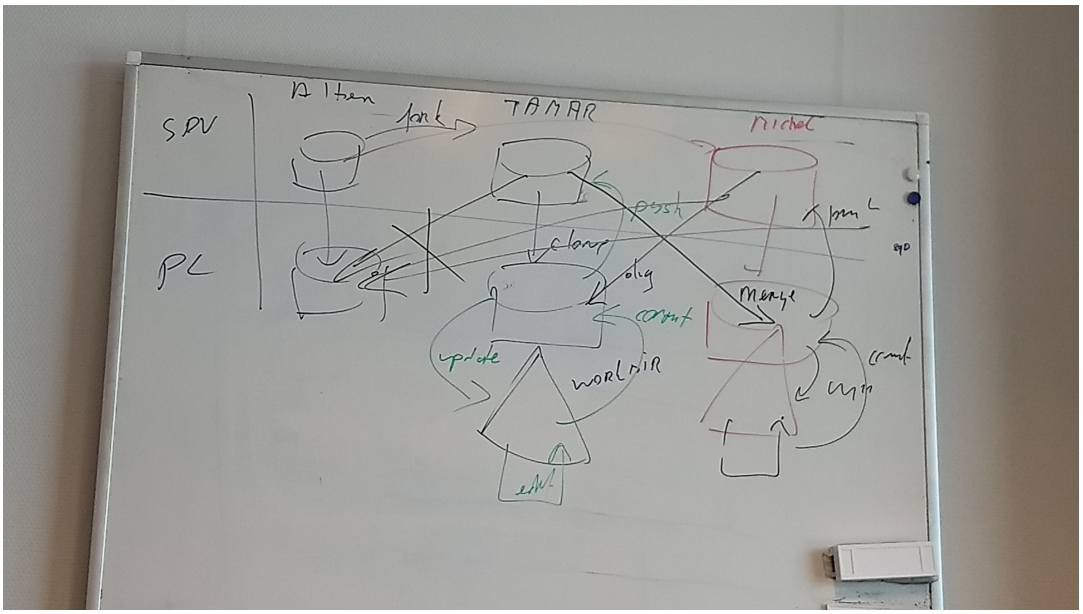


Fig. 27: Explanation of bitbucket

Session 2

date Thursday August 10th 2017

Tips and tricks for the next team

This page contains tips and tricks for future participants of the basic training.

Branching with Mercurial

In our trial and error to make branches without clodding the repository we came across an article that explains how branching works with Mercurial.

http://stevelosh.com/blog/2009/08/a-guide-to-branching-in-mercurial/#branching-with-named-branches

---

Note: Do not branch!

It's a good article, however there is often *no need to branch*; its even **forbidden** in this repro

Remember, making a fork (server-side clone) is simular: a split in development, to be joined later.

Quick CMD access

If you want to reach a directory fast in CMD, you can simple go to the folder in Explorer and type "cmd" in the address bar.

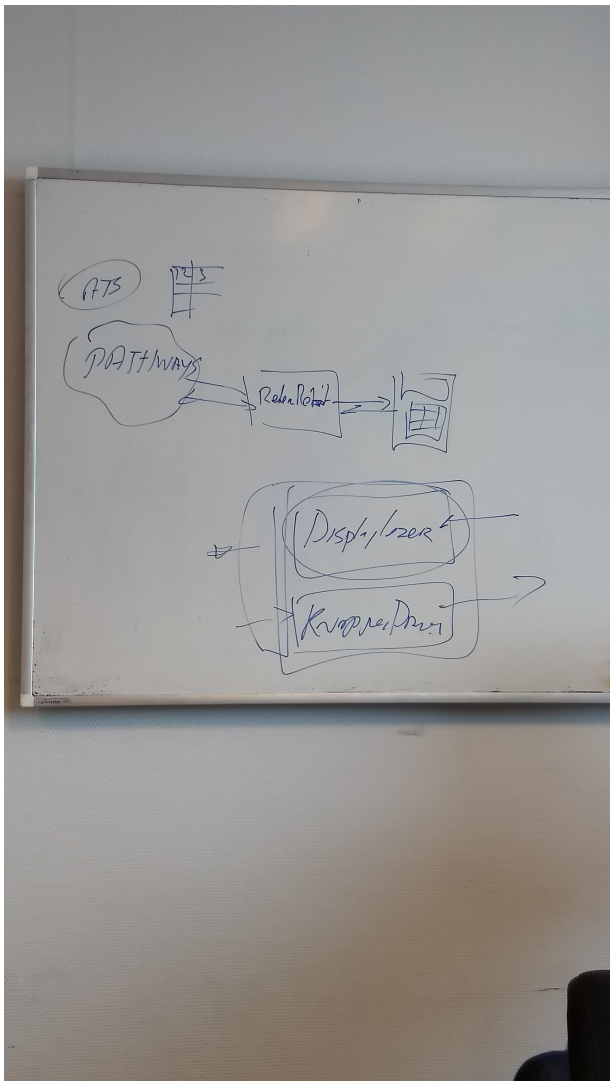


Fig. 28: The RekenRobot

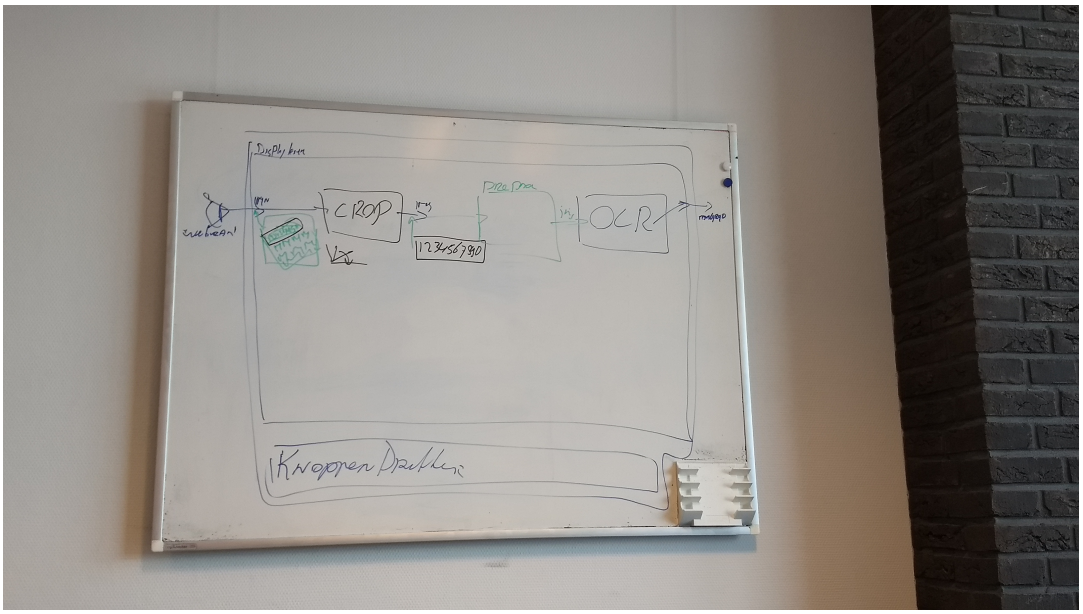


Fig. 29: Different components of the displayreader

Let's meet CherryPi

CherryPi consists out of three people, Hanrong, Tamar and Michel. Hanrong and Michel studied embedded software engineering and Tamar has a masters in biomedical engineering.

Image preprocessor

The preprocessor aims to remove noise and improve contrast, to aid the number recognition. We aimed to improve the existing preprocessor.

Here we present the results of our preprocessor tests. As it is hard to test the output of the preprocessor objectively, we tested whether the preprocessor improve the efficiency of the OCR module.

Effectiveness preprocessor

Test preprocessor effect

date Thursday August 17th 2017

Setup:

We generated 1000 images of the calculator-display, with different numbers and a 'normal' background (normal lighting and noise). In our first test, these numbers were used as input for the OCR (so without any preprocessing). In our second test, the default preprocessing was applied, executing the functions "gray_scale", "gaussian_blur", "filter2d", "binary_threshold", "median_blur" and "erosion". The preprocessed image was further analyzed in the OCR module. The results of the OCR (a string of numbers) was compared with the actual number and the fraction of correct outputs was assessed.

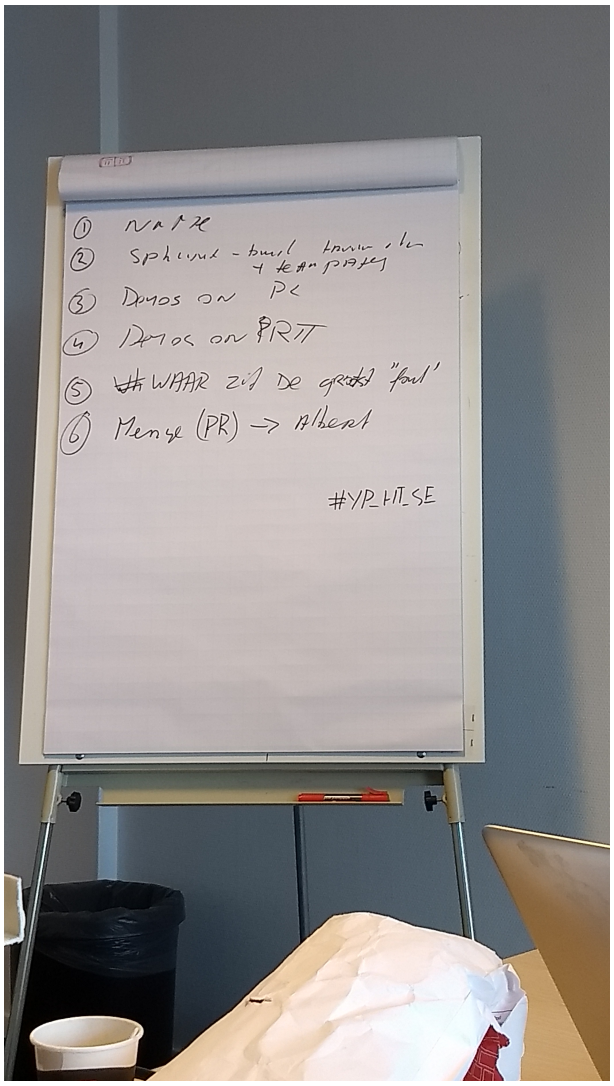

1 NAP
2 Sphinx - tool
4 team
3 Demos on PC
4 Demos on PIRT
5 WHAR zijn de
6 Merge (PR) → Albert
#YP_HTSE

Fig. 30: To do

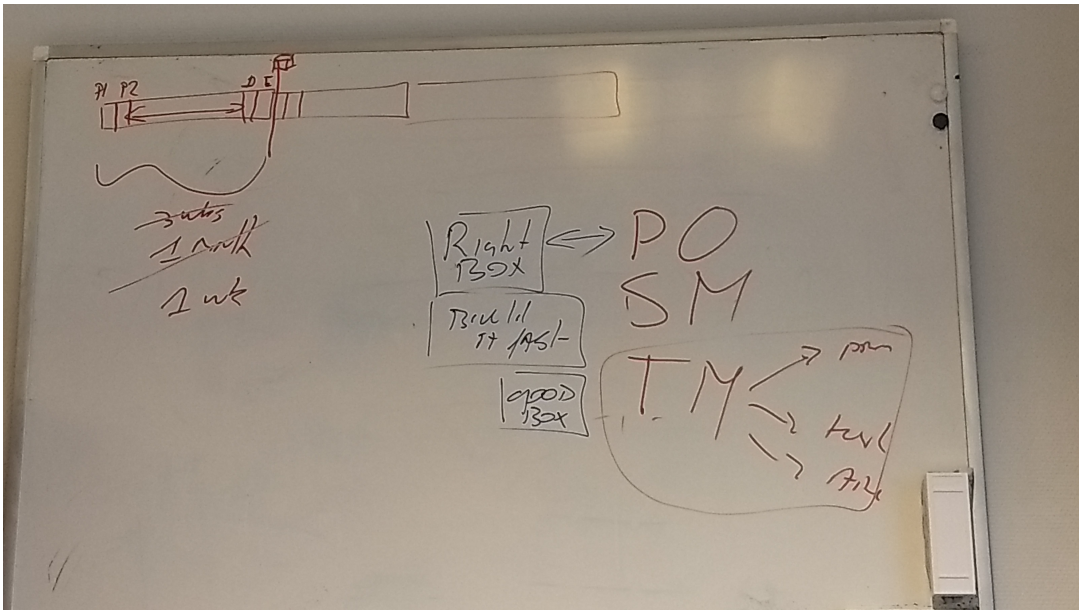


Fig. 31: Different tasks in scrumteam: product owner (PO), scrum master (SM), and team members (TM)

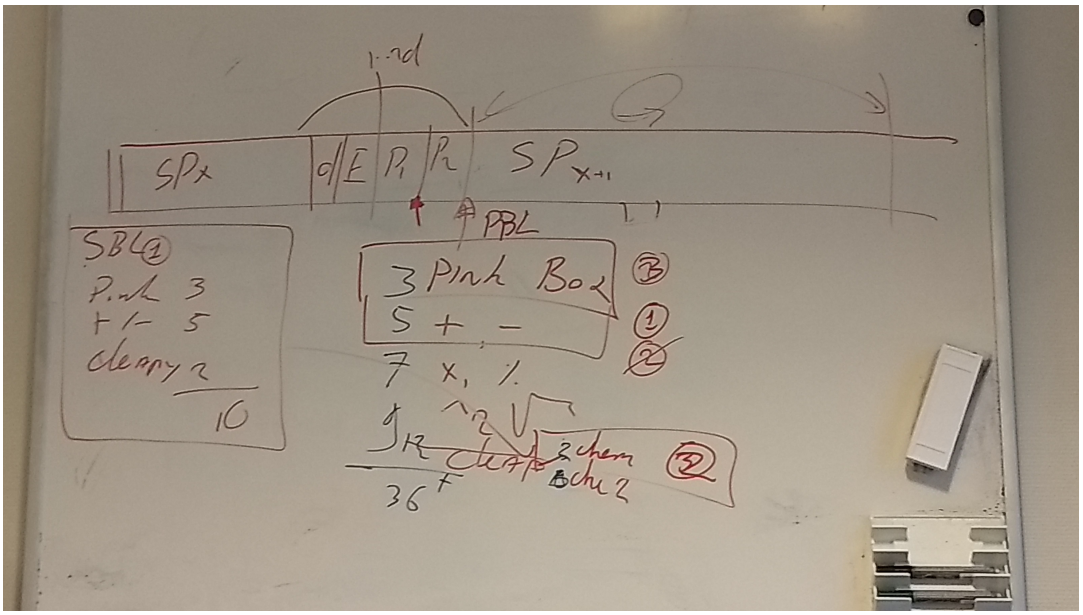


Fig. 32: Planning. p1: planning session 1 (what features will be planned this sprint), p2: planning session 2 (which tasks should be done, who is doing what), sp: sprint, d: demo's, E: evaluation

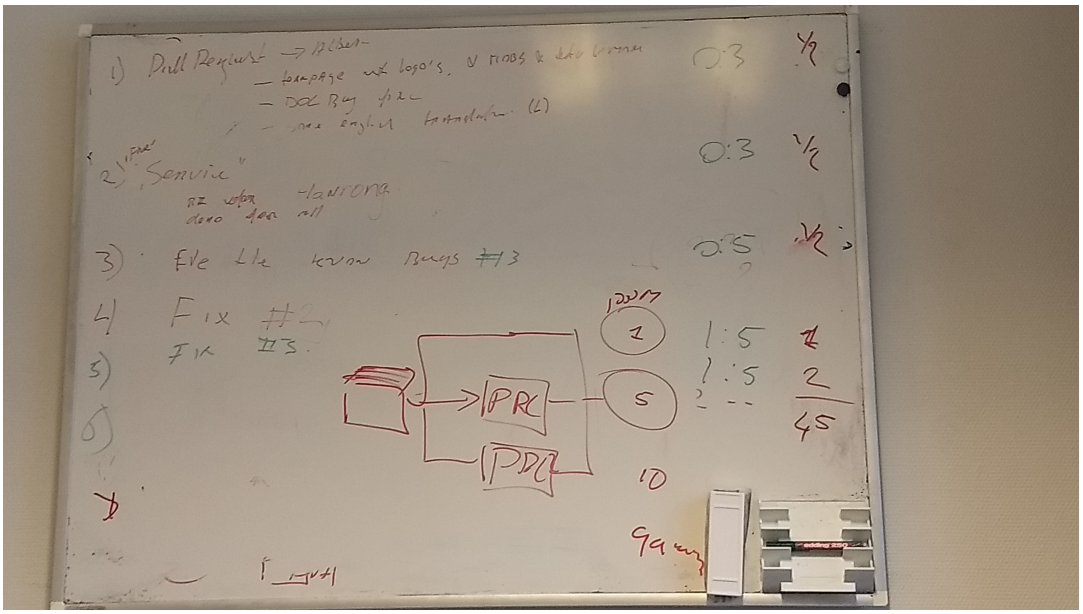


Fig. 33: To do (part 1)

Results:

Without preprocessing the OCR was able to recognize the numbers of 0 of the 1000 images (0%). With preprocessing, the OCR was able to recognize the numbers of 172 of the 1000 images (17%). This number does depend on the test images (which were randomly created and therefore differed between computers), with another set test images the success rate was 24%.

Conclusion:

The current preprocessing improves the efficiency of the OCR.

Test backgrounds

date Tuesday 29th August 2017

Setup:

We generated images on four different backgrounds using the DSEG7Classic font. We generated images with 2 digits, 3 digits and 4 digits with each background. The numbers were used to test the OCR with and without the use of the preprocessor. For the preprocessor we used the following configuration: “gray_scale”, “median_blur”, “binary_threshold” and “erosion”, in that order.

Results:

Without preprocessor, it seems to be able to recognize some with this font, but most are ignored.(No numbers excluded)

- **DSEG7Classic-Regular - clean_bg:**
  - 2Digits: Accuracy: 0.0%, correct: 0, number of images: 100
  - 3Digits: Accuracy: 0.0%, correct: 0, number of images: 1000
  - 4Digits: Accuracy: 0.0%, correct: 0, number of images: 951

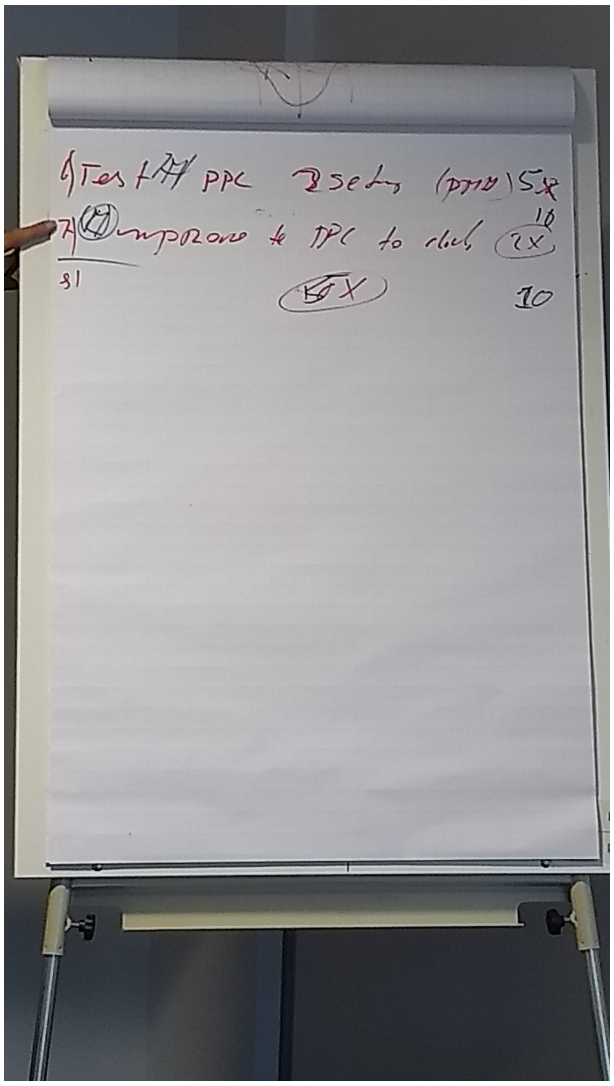


Fig. 34: To do (part 2)

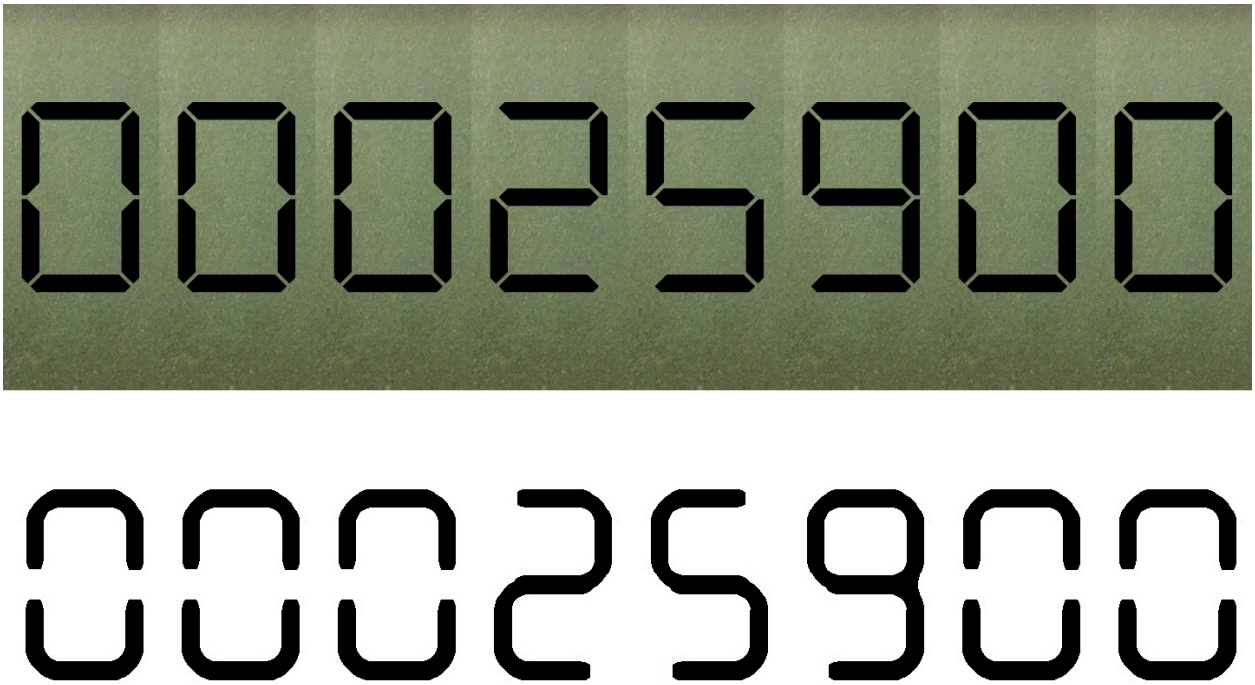
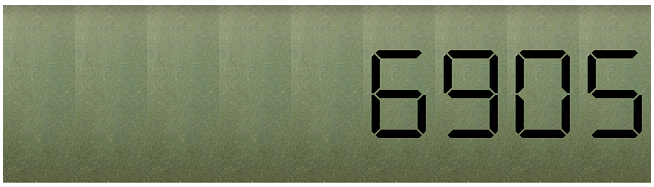


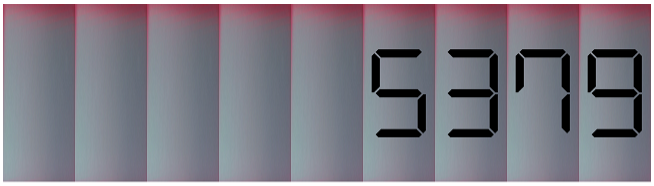
Fig. 35: Display before (top) and after (bottom) default preprocessing

- **DSEG7Classic-Regular - easy_bg:**
  - 2Digits: Accuracy: 0.0%, correct: 0, number of images: 100
  - 3Digits: Accuracy: 0.0%, correct: 0, number of images: 1000
  - 4Digits: Accuracy: 0.0%, correct: 0, number of images: 943
- **DSEG7Classic-Regular - normal_bg:**
  - 2Digits: Accuracy: 0.0%, correct: 0, number of images: 100
  - 3Digits: Accuracy: 0.0%, correct: 0, number of images: 1000
  - 4Digits: Accuracy: 0.52%, correct: 5, number of images: 958
- **DSEG7Classic-Regular - hard_bg:**
  - 2Digits: Accuracy: 0.0%, correct: 0, number of images: 100
  - 3Digits: Accuracy: 0.0%, correct: 0, number of images: 1000
  - 4Digits: Accuracy: 0.0%, correct: 0, number of images: 954

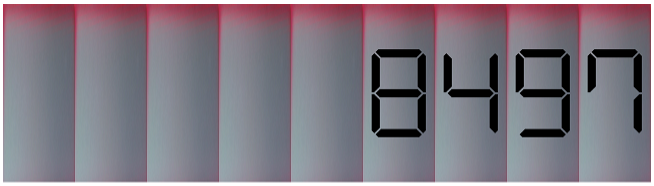
  With preprocessor, it recognizes around 45% of the images.(No numbers excluded)
- **DSEG7Classic-Regular - clean_bg:**
  - 2Digits: Accuracy: 53.0%, correct: 53, number of images: 100
  - 3Digits: Accuracy: 50.0%, correct: 500, number of images: 1000
  - 4Digits: Accuracy: 42.8%, correct: 407, number of images: 951
- **DSEG7Classic-Regular - easy_bg:**



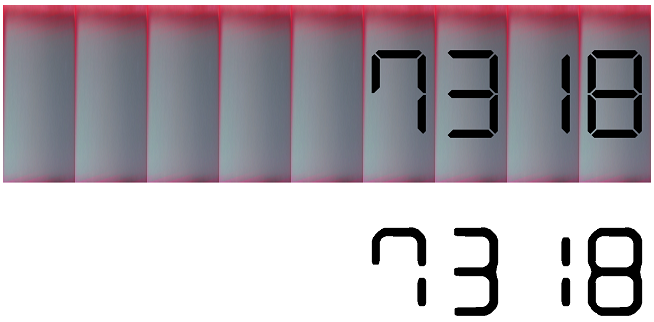
  - 2Digits: Accuracy: 47.0%, correct: 47, number of images: 100
  - 3Digits: Accuracy: 51.3%, correct: 513, number of images: 1000
  - 4Digits: Accuracy: 41.89%, correct: 395, number of images: 943



- **DSEG7Classic-Regular - normal_bg:**
  - 2Digits: Accuracy: 47.0%, correct: 47, number of images: 100
  - 3Digits: Accuracy: 49.7%, correct: 497, number of images: 1000
  - 4Digits: Accuracy: 40.29%, correct: 386, number of images: 958



- **DSEG7Classic-Regular - hard_bg:**
  - 2Digits: Accuracy: 35.0%, correct: 35, number of images: 100
  - 3Digits: Accuracy: 50.0%, correct: 500, number of images: 1000
  - 4Digits: Accuracy: 42.03%, correct: 401, number of images: 954



Conclusion:

The different backgrounds seem to have little effect on the percentage. Although the amount does differ. A lot of the incorrect numbers are due to the '0' and the '8'. In all of the test images not one '0' was recognized out of the 2846. I case of the '8', 220 were recognized in correct numbers out of the 2485. There are incorrect numbers where an '8' is recognized, but failed due to a '0' or an '8'. Results of those numbers are for example:

- Predicted: 8636, Actual: 086.
- Predicted: 8637, Actual: 087.
- Predicted: 86363, Actual: 088.
- Predicted: 1488, Actual: 1480. <- '8' is correct, failed due to '0'
- Predicted: 89633, Actual: 0983.
- Predicted: 89635, Actual: 0985.
- Predicted: 896363, Actual: 0988.

Tips for next test:

In case of the '8' the occurrence is a lot less than the '0', but often enough to have a big effect on the total percentage. Tests without the use of those numbers should show a massive difference in accuracy. The resolution is a lot higher than the cropping module returns. This should be about the same for higher realistic accuracies. Also smaller images seem to be easier to read by the OCR module, but tests should prove this statement.

Randomness

date Monday August 21st 2017

Aim:

To test whether there is some randomness in recognizing the numbers. If the same image is processed 100 times, it should fail 100 times or succeed 100 times, otherwise there is some randomness in the preprocessing/OCR module.

Setup:

Five test images, generated without any adaptations (e.g. no noise added) were tested 100 times (11312555, 11799773, 31195166, 75888955, 89925729). The default preprocessing was applied, executing the functions "gray_scale", "gaussian_blur", "filter2d", "binary_threshold", "median_blur" and "erosion" and the images were further analyzed in

the OCR module. The results of the OCR were compared with the actual numbers on the image.

Results:

The numbers 11312555, 11799773 and 31195166 were recognized 100 times, while the numbers 75888955 and 89925729 were never recognized.

Conclusion:

There is no indication for a randomness in the preprocessing or OCR module.

Effectiveness with different images

date Thursday August 24th 2017

Aim:

To test the effectiveness of the preprocessor and OCR, but with more variety in images.

Setup:

We created different options in the Image Generation:

1. Different number of digits
2. Option to add noise, blur and deform the image, to make it look more realistic
3. Different backgrounds
4. Possibility to exclude some numbers - not done yet

Each option was assessed with the default preprocessing option ("gray_scale", "gaussian_blur", "filter2d", "binary_threshold", "median_blur"). With each option, we created maximum 1000 random images (or, if the number of digits is <3, the total possible number, i.e. 10 with 1 digit)

Results:

1. **Accuracy depending on the number of digits (# good/total #)**
   (a) 0% (0/10)
   (b) 47% (47/100)
   (c) 49% (493/1000)
   (d) 42% (398/959)
   (e) 31% (304/991)
   (f) 24% (244/1000)
   (g) 20% (200/1000)
   (h) 15% (149/1000)
2. **Accuracy depending on blurring etc. (tested on 1000 3-digit images)**
   - If the figure is only blurred: accuracy = 0% (0/1000)
   - If only the size of the figure is decreased: accuracy = 0% (0/1000)
   - If the figure is only deformed: accuracy = 0% (0/1000)
   - If figure is blurred, deformed and with noise: accuracy = 0% (0/1000)

3. **Accuracy depending on different backgrounds (tested on 200 3-digit images)**
    - default background: accuracy = 50.55% (92/182)
    - background 1: accuracy = 65.61% (124/189)
    - background 2: accuracy = 42.39% (78/184)
    - background 3: accuracy = 56.18% (100/178)
    - background 4: accuracy = 0% (0/180)

Preprocessor

Current functions

Different OpenCV functions are applied in the preprocessing, which are explained at http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/

Currently, the following functions are implemented in the RekenRobot:

- grey-scale
- gaussian_blur: low-pass filtering, to remove noise
- filter2d: 2d convolution(increases amount of pixels?), kind of low-pass filtering, to remove noise
- binary_threshold (convert to binary image)
- normalization: increase contrast, by stretching the histogram to the whole range.
- white_boarder: puts a white boarder around image (does not seem to be usefull. . .)
- median_blur: to remove salt-and-pepper noise
- erosion: erodes white image. As the numbers are black on a white background, the numbers actually get dilated with the erosion function.

  All of these functions can be used, although the default preprocessor does not apply the normalization and white border.

Order of preprocessing functions

Current default order: 1. Grayscale 2. Gaussian blur 3. 2d convolution 4. Binary threshold 5. Median blur 6. Erosion

- Grayscale should be performed before thresholding.
- Gaussian blur, 2d convolution and median blur can be performed both before and after grayscale.
- Gaussian blur and 2d convolution can be done after thresholding, but might introducing new grey-values.
- Erosion should be performed on binary image.
- Normalization should be performed after grey-scaling, but before thresholding (currently not in default preprocessing).

Possible improvements

Roughly sorted from most to least promising.

- Binary treshold: At this moment, a global threshold is applied. This threshold can be made adaptive, in order to better handle varying contrasts. (see http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html#thresholding)
- Improve order of functions, and check if all functions improve the preprocessing.
- Perspective transformation: might be added to correct for the perspective (i.e. the deformation created by the camera).
- At this moment, the normalization is not working properly, while this might improve the preprocessing. Further looking into the normalization might therefore be usefull. Adaptive histogram equalization might be used (see http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_histograms/py_histogram_equalization/py_histogram_equalization.html#py-histogram-equalization), but I'm not sure if that would improve the normalization in our case.
- Gaussian filtering might be replaced by bilateral filtering. With bilateral filtering, a gaussian filter is used as well but it also considers intensity differences. This should keep edges sharper than with gaussian filtering, but it might be slower. (see http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_filtering/py_filtering.html#filtering)
- High pass filtering: as the contrast between the numbers and background is large, high-pass filtering might be used to detect the numbers as well (using laplacian derivatives, http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_gradients/py_gradients.html#gradients)

- **Erosion: Erosion and dilatation can be applied to remove noise as wel:**
  - first erode, then dilate will remove noise in the numbers (black area)
  - first dilate, then erode will remove noise in the background (white area)

Added functionalities

We implemented both the adaptive treshold and the image deformation (perspective_transform) as preprocessor step (with as name binary_adaptive_threshold). The results of this is highly dependent on the input image:

The image deformation does exactly the opposite of the deformations applied to the test images. Therefore, this works perfectly on the test images, but worse on real images. If tested using the webcam, it does not appear to have much added value.

Using a test set of 200 images (4 digits, 0 and 8 excluded, including blurring, noise and deformation, clean background), we compared the default preprocession (gray_scale, gaussian_blur, filter2d, binary_threshold, median_blur, erosion) with the new preprocessing (gray_scale, perspective_transform, filter2d, binary_adaptive_threshold). The succes rate of the default preprocessing was 0%, and of the improved preprocessing 74%.

If testing with actual webcam images, the "improved" preprocessing (gray_scale, filter2d, binary_adaptive_threshold and median blur) shows much better results than after the default preprocessing. However, the border shadows sometimes appear black in the image, resulting in additional recognized numbers by the OCR module.

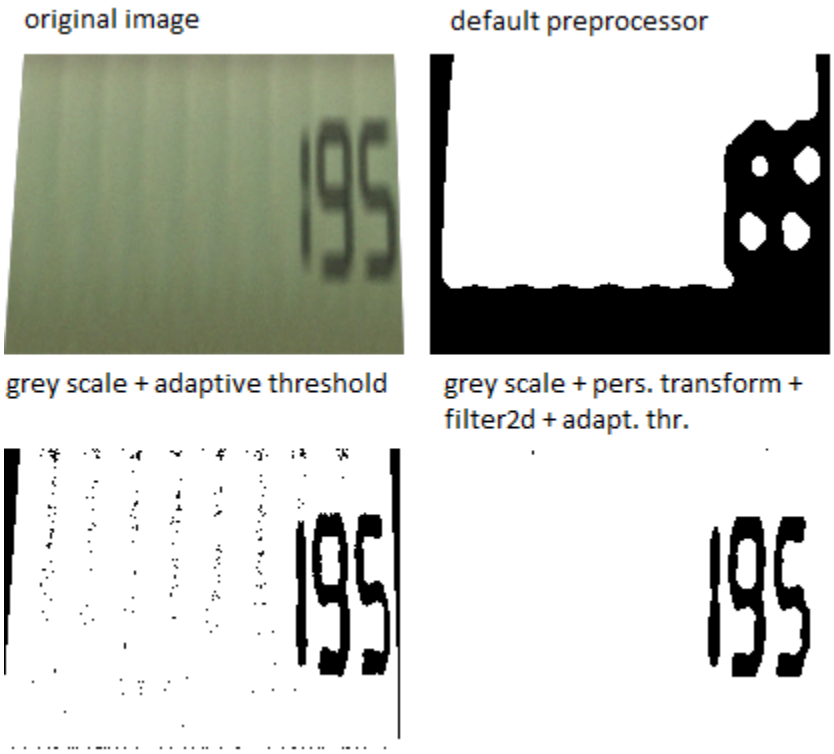


Fig. 36: Original "real image" and after different preprocessing options

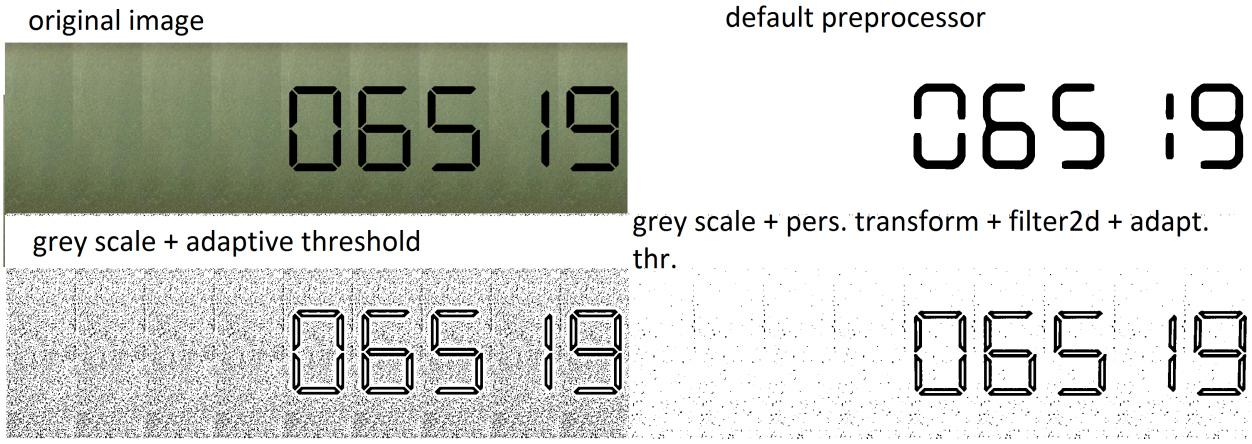


Fig. 37: Original "good image" and after different preprocessing options (scaled at 25%)

Fig. 38: Real calculator image with default and improved preprocessing.

Help

Bill Nye tries to save the world, so do we. One line of code at the time.

Our method

Persistence is key, maybe cry a little every now and then, but stay persistent!

3.1.6 sPYinkenHof

date 5 feb 2018

Welcome to the team page of the team sPYnkenhof.

General

Here you'll find the notes of the team.

Sprints

This index contains a summery of our sprints.

Tasks for Sprint 1

ToDo

- **raspberry pi demo (8 FeaturePoints)**
  - Installing software, getting started
  - Reading previous team documentation
  - Testing Raspberry -> VNC, static ip,
  - Running demo on pc's
- **pathways-extensions-training (5 FeaturePoints)**
  - Install Sphinx
  - Learn rst format
  - Create team page
  - Create a sprint page
- **pull request (3 FeaturePoints)**
  - Create clean forks
  - Join rst files
  - Commit
  - Pull request

Mop Gallery

This page shows all the pictures taken of the board notes.

Session 1

date Thursday Februari 8th 2018

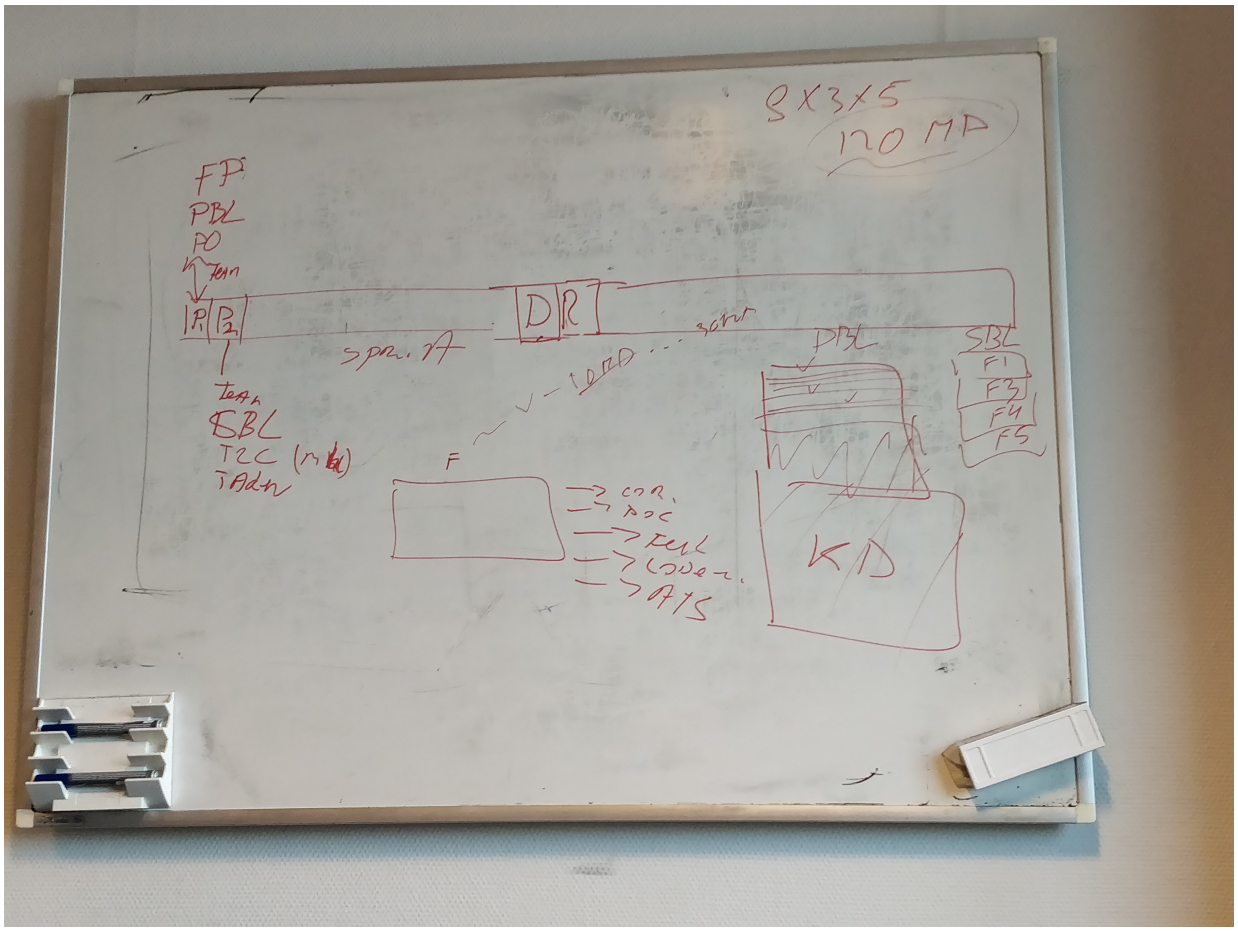


Fig. 39: Overview of tasks activities during a sprint

Session 2

date Thursday Februari 15th 2018

Session 3

date Thursday Februari 22nd 2018

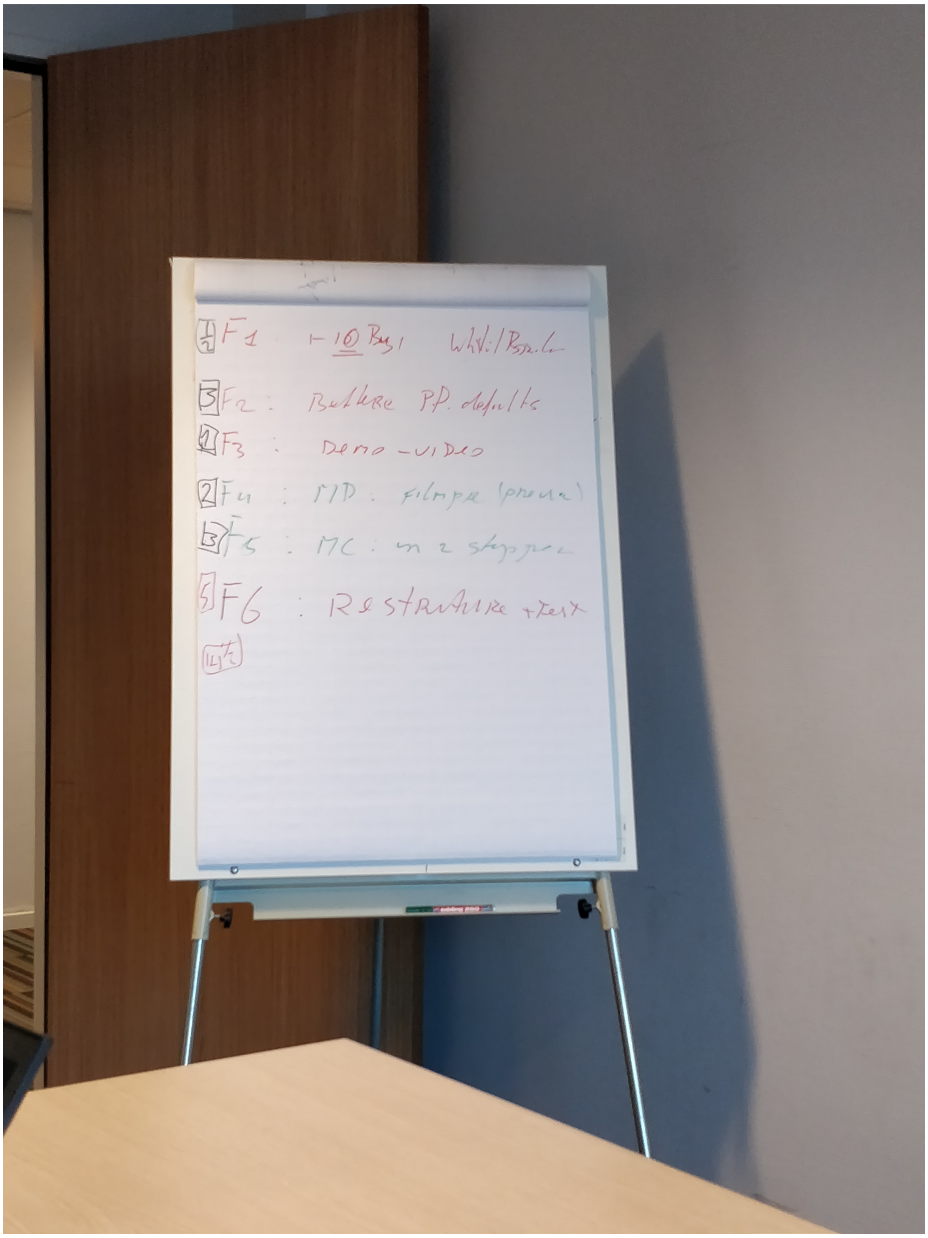


Fig. 40: List of features for sprint 2

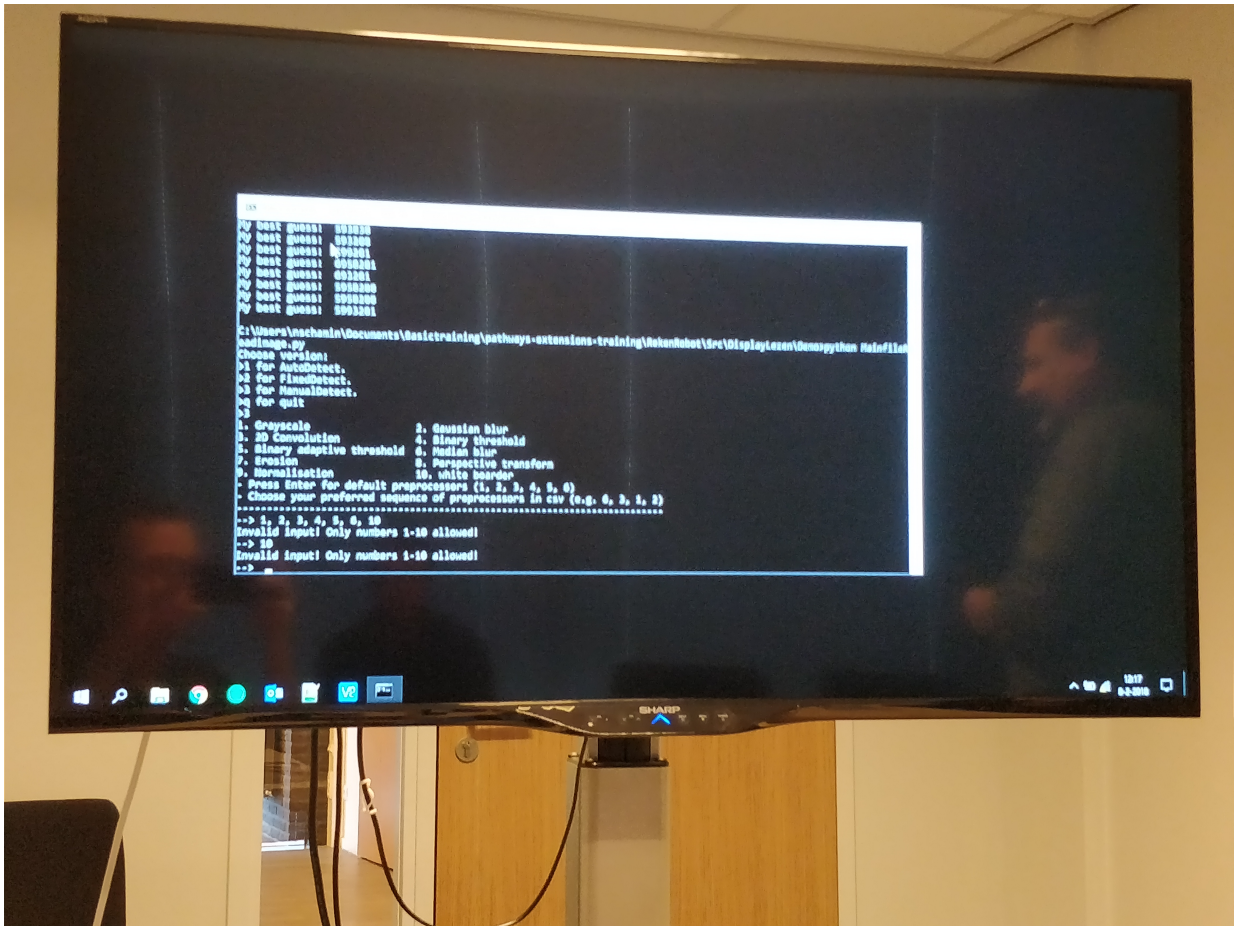


Fig. 41: Located a bug in the filter select menu

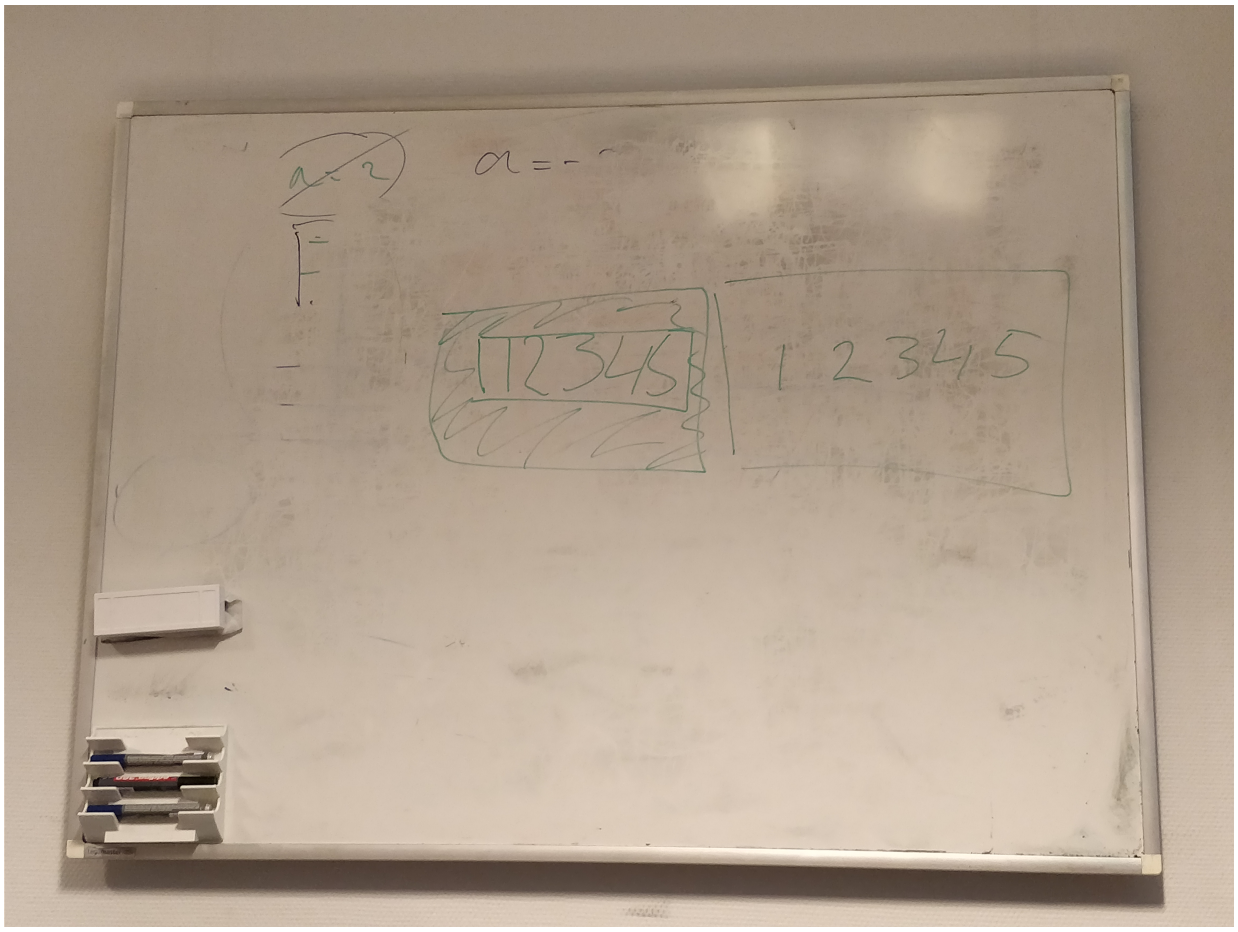


Fig. 42: Feature 1 explanation

Fig. 43: Work in progress

Fig. 44: Work in progress

Session 4

date Thursday March 1st 2018

Fig. 45: Work in progress

Session 5

date Thursday March 8th 2018(??)

Fig. 46: Work in progress

Let's meet sPYnkenhof

sPYinkenHof consists out of four people, Bert, Ian, Niels and Thom. Bert and Ian studied embedded software engineering, Thom studied Business IT & management and Niels studied Physics.

3.1.7 PiSolo

date June 6th 2018

Welkom op de PiSolo, one man army team, pagina

General

Notities van het project worden hier bijgehouden

Notes

author Bram

date 8-6-2018

Hier worden notities bijgehouden over uitgevoerde werkzaamheden. Wat is er gedaan, wat ging er mis, wat gaat er nog gebeuren? Dat zijn de vragen die beantwoord worden per dag.

Notitie 6-6-2018

Vandaag gedaan bitbucket aangemaakt fork gemaakt fork gecloond eerste commit en pull request gedaan om te testen python geinstalleerd pi aangezet team page aangemaakt en aangepast. begonnen met documentatie doorlezen eerste demo gestart op pc en pi

Problemen tegen gekomen:

Probleem: Kon de source code niet vinden. Oplossing: https://bitbucket.org/ALbert_Mietus/pathways-extensions-training

Probleem readthedocs steeds weggeklikt en nou al drie keer naar mn mail moeten gaan. oplossing: bookmark maken van: http://pathways-extensions-training.readthedocs.io/en/latest/teams/2018.1_sPYnkenHof/index.html

Probleem wachtwoord van de pi was verandert: oplossing: sluit monitor, toetsenbord en muis aan op pi en typ in terminal: sudo passwd pi typ het nieuwe wachtwoord van pi dat je wilt gebruiken.

Probleem Eerste wat ik tegen kwam was het aanmaken van de locale clone van de fork. Je kon ergens kiezen tussen bitbucket server of een atlassian account source van je fork. oplossing: hierbij moet je atlassian account kiezen waarvan ik ff niet meer kan vinden hoe het heet.

Probleem Ik wist niet hoe ik een pull request moest doen oplossing: youtube

Probleem Weet niet hoe schoenendoos spullen in elkaar gezet moeten worden oplossing: nog onbekend

Dingen nog te doen:

documentatie lezen

Niet vergeten: bedenk tips aan het einde van het project en zet op readthedocs

Kijken ik de schoenen doos in elkaar kan zetten

kijken of ik een betere opencv functie kan vinden voor text detectie

Notitie 7-6-2018

Vandaag gedaan

Met Albert een hele hoop besproken.

Problemen tegen gekomen

Dingen te doen

Voor morgen

SMART doelen

paar minuten Documenteren Foto's online zetten van notities bijhouden van opmerkingen door de dag heen

Twee uur lees getting started lees Python docstrings lees extensions Neem de

drie uur eclipse customizen pep8 controle trailing white spaces sphinx auto build remove functie voor rst files verwijderen

een uur PlantUML installeren test maken met PlantUML

twee uur To Do lijst maken met taken die nodig zijn om te beginnen met het maken van de meekijker

Notitie 8-6-2018

Vandaag gedaan Documentatie doorgelezen(duurde anderhalf uur ipv twee uur)

Problemen tegen gekomen

Probleem Zoek functie werkt niet goed op readthedocs oplossing Hij doet het ineens.

Probleem plantUML werkt niet ook niet na pip install plantUML daarnaast moet er ergens een java file neergezet worden maar waar?

nog te doen

Notitie 11-6-2018

Vandaag doen:

plantUML werkend krijgen met nieuwe inzichten Verder gaan met het design van de image-relay

Problemen tegen gekomen

Probleem De sphinx build external tool is niet meer zichtbaar. Volgens mij komt dit doordat ik een keer eclipse twee keer tegelijk heb opgestart. **oplossing** Instellingen opnieuw toepassen en niet eclipse twee keer opstarten.

Notitie 12-6-2018

Vandaag doen:

plantUML werkend krijgen met nieuwe inzichten

Problemen tegen gekomen

Probleem

plantuml werkte niet ook na aanpassing van config file.

Oplossing

ik had als verwijzing naar de directory van plantuml.jar file gebruikt maar je moet zoals in linux / gebruiken anders snapt ie het niet.

DocumentatieNotities

date 8-6-2018

Sphinx-doc link doet het niet meer(hij zegt access denied) Onlogische volgorde, eerst zie ik sphinx en dan pas over python terwijl python nodig is voor sphinx? Albert is allergisch voor notedpad++ mmaar toch staat het in de docs waar staat conf.py? —> zoeken en aanpassen

Raspberry pi Ik zou vnc niet gebruiken.

Tesseract OCR(rpi) dollar tekens zijn irritant beetje vaag hoe tesseract letters moet leren.

OpenCV(rpi) Uuuh $VirtualEnvironment.rst is er niet meer ofzo?

OpenCV(laptop) Pip install opencv meerdere keren? 3.4.0 is er niet meer opencv_python-3.4.1-cp36-cp36m-win_amd64.whl is not a supported wheel on this platform.

Version control Is dit voor windows of op de pi?

Python docstrings builden Is dit gewoon een slechter gedocumenteerde versie van Sphinx-doc?

De opstelling De opstelling met de schoenendoos staat beschreven in de team pagina van duopinotti. Dit moet ergens anders komen te staan.

Sprints

Hier worden de sprints beschreven.

Sprint1

Sprint2

Eclipse

EclipseCustomizations

author Bram

date 8-6-2018

Deze pagina beschrijft de plugins en dergelijke die ik geinstalleerd heb voor eclipse.

PyDev

PyDev maakt het mogelijk python scripts te draaien in eclipse. Doe het volgende om PyDev te installeren:

- ga naar help–>install new software
- selecteer add
- Name: PyDev
- locations: http://pydev.org/updates
- druk op "ok" (of enter)
- selecteer PyDev in de lijst(niet mylyn)
- druk op next
- druk op finish

  Voordat PyDev gebruikt kan worden moet er nog wat geconfigureerd worden.
- ga naar window–>preferences
- Ga naar PyDev–>interpreters–>python interpreter
- klik op "new"
- Python name: python3.X(afhankelijk van je versie is de X anders)
- Interpreter executable: gebruik browse om python.exe te vinden(Staat waarschijnlijk in C:Program Files (x86)Python36-32)

  bron: https://www.youtube.com/watch?v=4ZO0-AOSLwA

pep8 controle voor python

pep8 is een opmaak standaard voor python. Door gebruik te maken van de plugin zie je altijd wanneer je niet goed aan deze standaard voldoet.

- Open Window–>Preferences
- Ga naar Pydev–>Editor–>CodeAnalysis–>pycodestyle.py(pep8)
- Selecteer Error, Warning of Info afhankelijk van hoe prominent het aanwezig moet zijn dat je pep8 niet goed toepast

Trailing whitespaces

Om trailing whitespaces tegen te gaan in eclipse:

White spaces laten zien:

- ga naar Window–>General–>Editors–>Text Editors
- Selecteer het vakje met “show whitespace characters”

Sphinx in Eclipse

Om makkelijk vanuit eclipse sphinx documentatie te generen kan het volgende gedaan worden:

- maak een .bat file met sphinx-build -b <pathways-extensions-trainingdocsdoc locatie> <plek waar het resultaat moet komen>
- ga naar run–>external tools–>External tools configuration
- klik met rechtermuisknop op program en klik new
- geef bovenaan een naam(bijvoorbeeld sphinx-build) en vul bij location de locatie van de batch file in

  Wat dit doet is simpelweg het .bat file runnen als je op run external tool drukt. Voorbeeld inhoud van bat file: sphinx-build -b html C:UsersbmartensDocumentspathways-extensions-trainingdocsdoc C:UsersbmartensDocumentsbuild

  Andere mogelijkheid is PeST instaleren:

- Help–>Eclipse Marketplace–>find: pest
- Klik op install bij rest editor

  Note: Dit voegt bij “Run As” de optie Sphinx as PDF en Sphinx as html toe. Het geeft echter bij mij ‘make.bat’ is not recognized as an internal or external command error.

  Hot key instellen voor sphinx-build:

- Windows–>Preference–>General–>keys
- Zoek command: Run last used external tool
- Hier bij binding kan je aangeven welke knop je ervoor wilt gebruiken

plantUML

plantUML installeren

De installatie van plantUML in sphinx en windows wordt hier besproken.

Doorloop de volgende stappen om plantUML te installeren:

- Download plantuml.jar bij http://plantuml.com/download[1]
- Plaats plantuml.jar in C:/temp/plantUML[2]

> **Warning:** This is incorrect!
>
> - One should **never** change a global setting
> - One should **never** adapt a project-setting (like `conf.py`) with **personal** vallues
>
> **See also:**
>
> *Tips about plantUML*

- Voeg plantUML ondersteuning aan sphinx toe het met volgende commando:

```
pip install sphinxcontrib-plantuml
```

Met deze stappen zou het mogelijk moeten zijn plantUML te installeren en te gebruiken. Zelf krijg ik nog wel de volgende waarschuwing: Currently, RtFD does not support plantUML drawings. Maar ondanks deze waarschuwing werkt het wel.

Les van Albert

Les van Albert

Uitleg zeventig dertig regel

Uitleg diagram dat communicatie tussen de modules door de tijd laat zien.

Schets van meekijker ontwerp

Soort to do lijst voor de onderdelen die nodig zijn voor de meekijker

3.1.8 ALbert Trips & Tricks

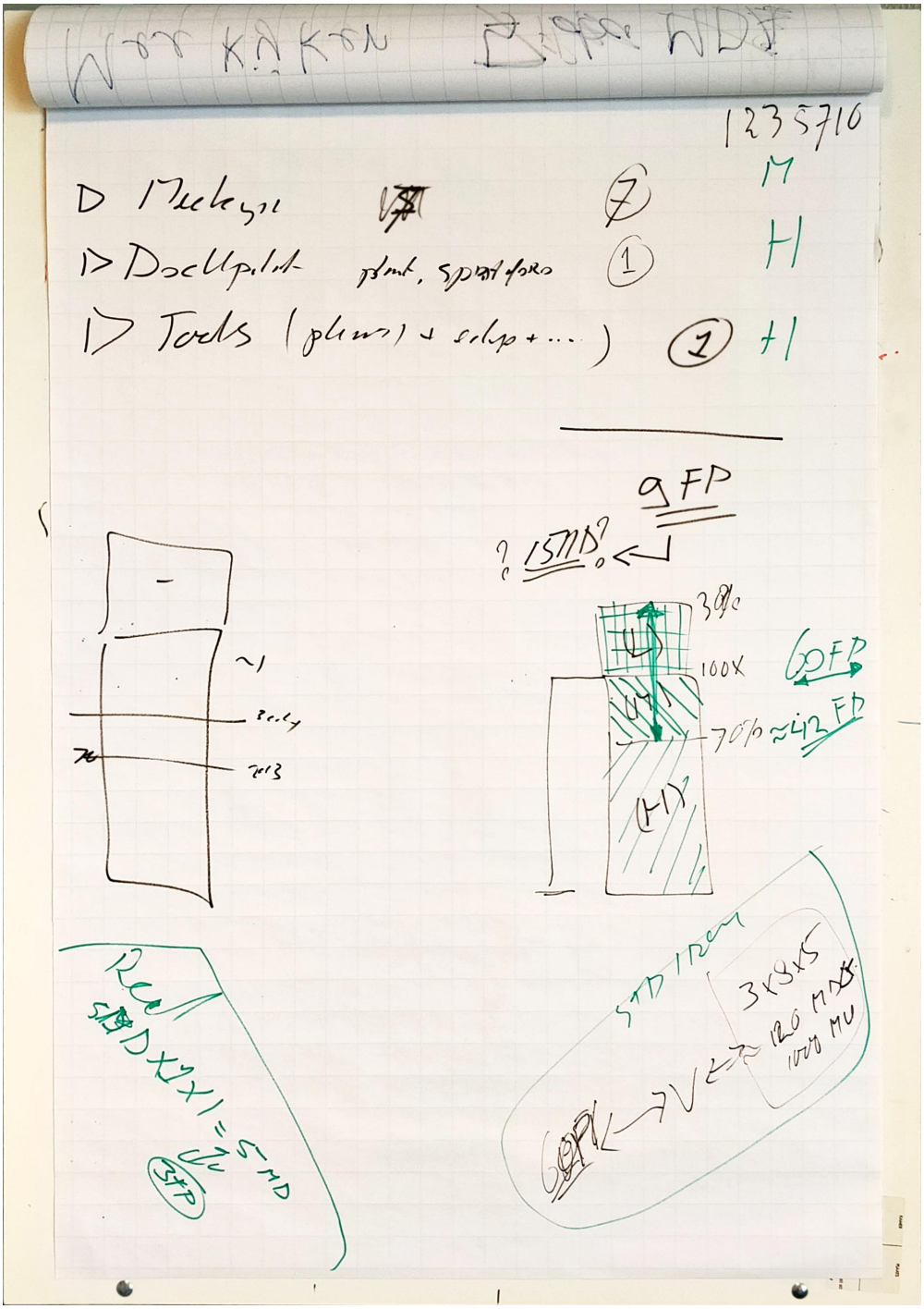
author Albert

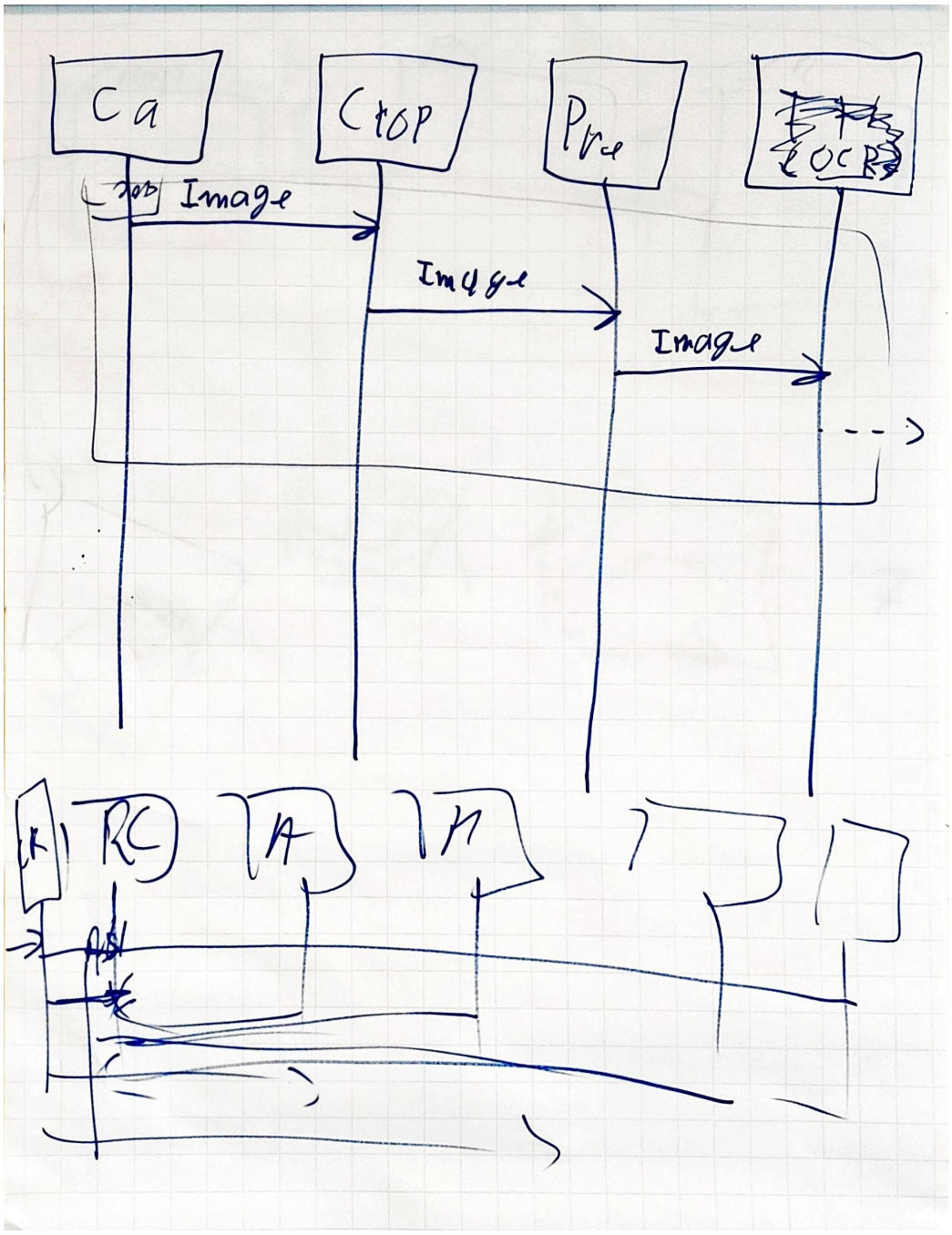
Tips about plantUML

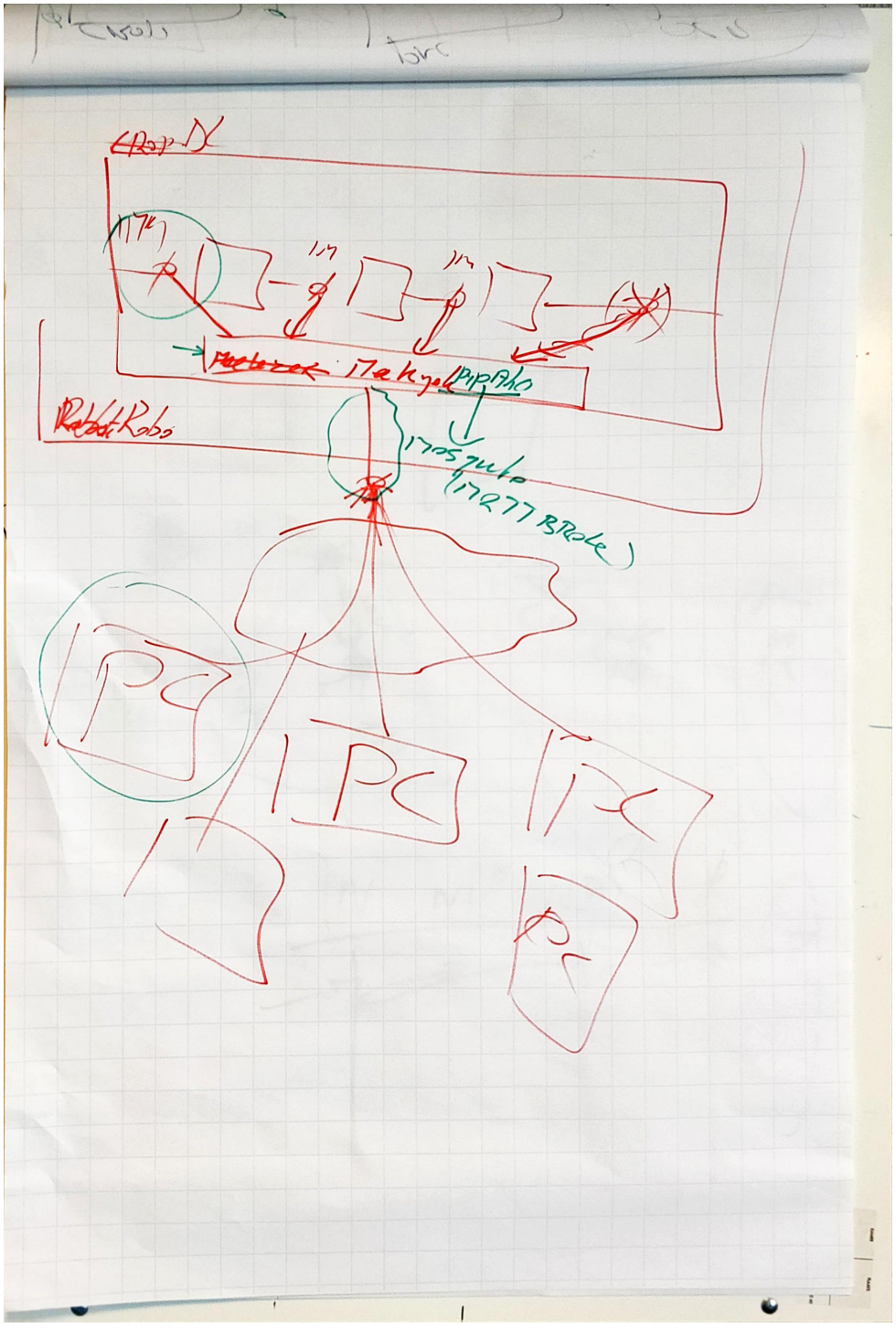
Tip: Use a bash/batch stript to connect . . .

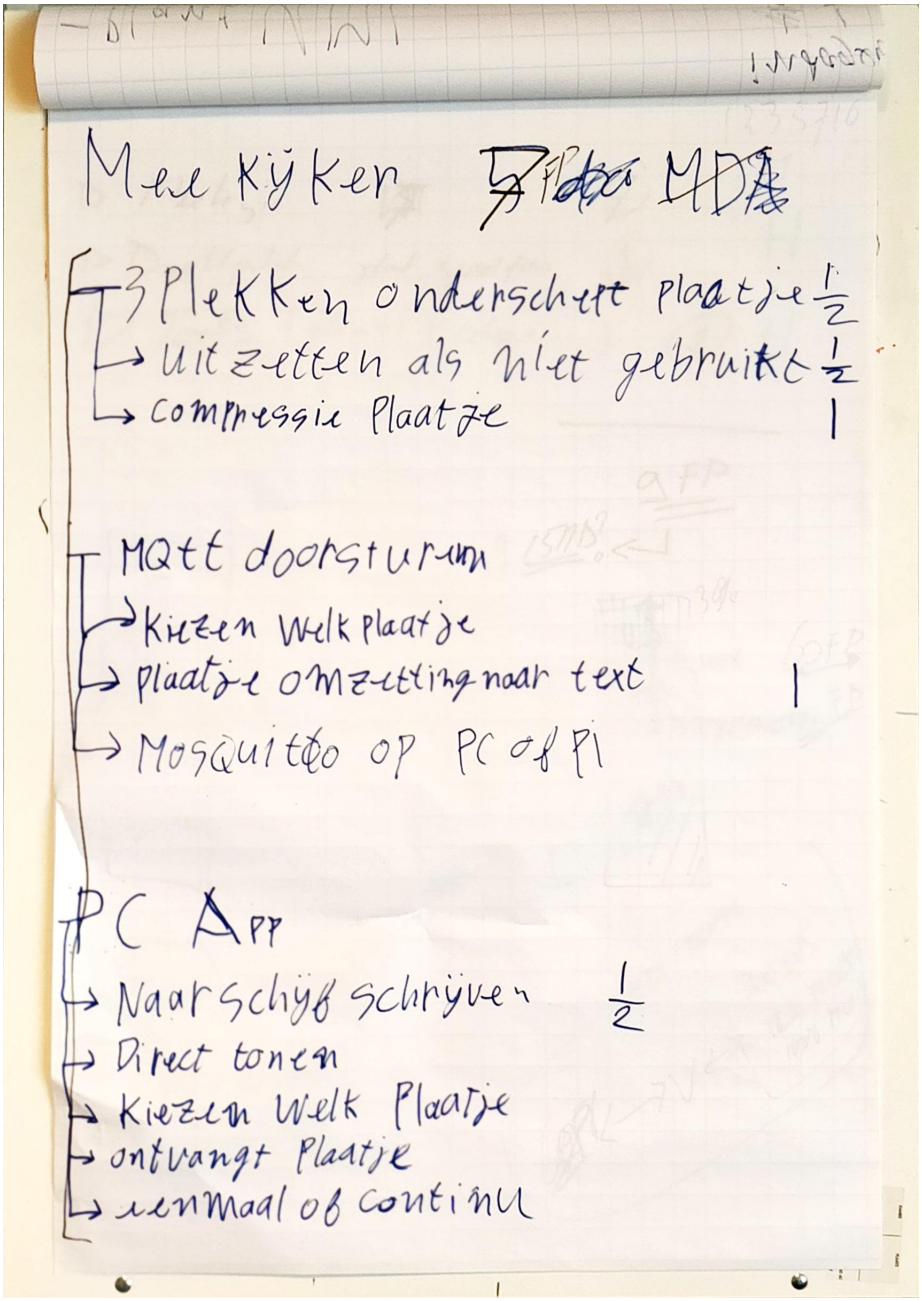
By default plantUML will call the executable `plantuml` (without extensions!). So, make sure you have a small bash/batch file with that name somewhere in the *$PATH* (*%PATH%* in windows).

[1] Door plantuml.jar op te starten en een text bestand met een voorbeeld op te starten is het mogelijk te kijken of het zo werkt.
[2] Deze locatie is te veranderen in conf.py


1235710
M
H
H
9 FP
60 FP
~42 FP
100%
70%
3x3x5
120 MB
100% MU


Ca
Crop
Pre
OCR
Image
Image
Image




Mee kijker
3 plekken onderscheft plaatje 1/2
uitzetten als niet gebruikt 1/2
compressie plaatje 1
MQtt doorsturen
Kiezen welk plaatje
plaatje omzetting naar text 1
Mosquitto op PC of Pi
PC App
Naar schijf schrijven 1/2
Direct tonen
Kiezen welk Plaatje
ontvangt Plaatje
eenmaal of continu

That file, which should *NOT* be part in the project (read: do not check-in), will be called; by sphinx (`sphinx-build`). And should call the correct *real* programm. In this case the java (executable) with the correct jar-file (here: plantuml.jar)

DO not forget the pass **all** parameters that sphinx give to the java-programm!

QuickLinks

- *Things to do:*

3.2 Pathways glossary

Tip: This is a **Copy** (do not edit)

It's a copy of the pathways-file: glossary.html

See also:

http://pathways-rd.readthedocs.io/en/dev_ats.fixtures/glossary.html

---

ATS

ATSes

Automatic Test Script

Automated Test Script An *ATS* is a scripted test to *'prove'* (or *disprove*) a part of the developed *PUT* works correctly. As such it delivers only **one bit** of information: verified of fail. This implies it does verify (check) the results of test.

An *ATS* typically consists of a number of steps, like starting a session, doing several operations, and verifying the results. Each step, not only the verification once, have to be successful to get an 'OK'.

ATSfile

ATStest As a single **ATSfile** can contain several **ATStest**s and both are typically called *ATS*, the suffixes *file* and *test* are sometime used distinguish them.

TsTbrick

brick A (test) **brick** is the basic, *reusable*, unit to build (all) *ATS*es. A *brick* offers a conceptual, *human oriented* set of commands to the *ATS*; so the upper-interface of the brick is *independend* of the technology of the PUT.

Typically, the brick's implementation should also not depend on (PUT-implementation & -interfacing) technology depending; it makes the *brick* better reusable.

There are some brick-like unit for that; see: *cobblestone*, *gate*

gate During the executing an *ATS* all kind of commands and results are passed to and from the *PUT*, by an interface called the *gate*. The implementation of a *gate* is typically specific to the technology of that that *PUT*. At the same time, the role of the *gate* is to hide that technology. Such that *cobblestone*s are less depending on that specific *PUT* and *brick*s not at all.

interface Deprecated since version pre-alfa: Old name for *gate*; do not use anymore

cobblestone A *rough* kind of brick, for special cases. By example to extend a `pathways.puts.put`

A real brick has a functional interface; with cobblestone a second, *lower-level* way to interact with the PUT possible. Which isn't used in most tests, but can be convenient in exceptional cases. It better to have a test, that scripted with to many details, then having no option to automate it.

This is best explained with a simple web-calculator. A (human) user can control it by clicking buttons, reading or entering some text, etc. With a cobblestone that level of control is available for a script. Although, a good test isn't build out of this kind of commands. They use more functional commands, like: 'add two numbers'; a typical brick-command.

With a cobblestone those smaller *steps*, like *clicking*, entering test and reading values are possible. Just for in case they are needed to test the exception.

Secondly, cobblestones are often used to script real bricks. By example the 'add two numbers' bricks-command can be implemented by entering a some strings, clicking a few buttons, and reading the result.

PUT

Product Under Test Depending on the context, the term **put** can refer to the actual product that is tested, or to the proxy-object that represents it.

The proxy-object is typically a subclass of `pathways.puts.put`, or one of it subclasses in `pathways.puts`, which can be extended with *cobblestone*s

feed The **feed** part of a test-vector are used as input for the *PUT*.

expected

expected results The **expected results** (or shortly: **expected**) part of a test-vector specify the values that the *PUT* should return. They are (smartly) compared with the *actual results*.

actuals

actual results This are the values the *PUT* outputs and are compared to the *expected results* to verify correctness.

Note that the compare has to be *smart*; not all parts of the output are relevant. By example an (generated) XML-file typically contains a datestamp; which will vary each run. So that part is typically filtered out

conditions New in version Future: Currently no support is available for conditions. Only *feed* and *expected* can be used now.

Some tests may need more complex validations then comparing the *expected results* with the *actual results* after giving the *PUT* an input *feed*. By example, real-time demands may specify a maximum delay. This kind of conditions can be given in the **condition** part of a test-vector.

TstVector

Todo: TstVector

fixture

fixtures

Todo: fixture

3.3 Things to do:

Todo:

- re.match("^[1-9,]*$", pp_sequence) is line 48 allows all number above 1!

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/pathways-extensions-training/checkouts/default/RekenRobot/Src/DisplayLezen/FlexPreprocessInitializer/InitializeFlexPreprocessor.py:docstring of FlexPreprocessInitializer.InitializeFlexPreprocessor.InitializeFlexPreprocessor.create_preprocessor_sequence, line 5.)

Todo:

- **Create the option to generate:**
  - images of all backgrounds in one call
  - all digit ranges in one call
  - images of all possible fonts in one call
  - all possibilities in one call
- Refactor code in sub functions

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/pathways-extensions-training/checkouts/default/RekenRobot/Src/DisplayLezen/TestEvaluateOcr/ImageGenerator.py:docstring of TestEvaluateOcr.ImageGenerator.main, line 7.)

Todo:

- Print complete numbers on 1 background, instead of combining individual numbers + backgrounds.
- Ensure a 0 can't be placed in front of any other number
- Create a folder for processed and non-processed images

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/pathways-extensions-training/checkouts/default/RekenRobot/Src/DisplayLezen/TestEvaluateOcr/ImageGenerator.py:docstring of TestEvaluateOcr.ImageGenerator.Digits, line 11.)

Todo:

- Make different steps more flexible.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/pathways-extensions-training/checkouts/default/RekenRobot/Src/DisplayLezen/TestEvaluateOcr/ImageGenerator.py:docstring of TestEvaluateOcr.ImageGenerator.adaptImage, line 6.)

Todo:

- Allow the fonts to be used without being installed.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/pathways-extensions-training/checkouts/default/RekenRobot/Src/DisplayLezen/TestEvaluateOcr/DigNumberGenerator.py:docstring of TestEvaluateOcr.DigNumberGenerator.main, line 6.)

Todo: Later in te vullen

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/pathways-extensions-training/checkouts/default/docs/doc/extensions/RekenRobot/opdracht/KnoppenDrukker.rst, line 21.)

Todo: TstVector

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/pathways-extensions-training/checkouts/default/docs/doc/glossaries/pathways.rst, line 116.)

Todo: fixture

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/pathways-extensions-training/checkouts/default/docs/doc/glossaries/pathways.rst, line 120.)

Todo: Documentatie nummergenerator maken.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/pathways-extensions-training/checkouts/default/docs/doc/teams/2017.3_DuoPiNotti/Normalisation.rst, line 20.)

Python Module Index

Index

W