Software arkitektur

Bachelorprojektet:
Real-time eye-tracking
Projektnummer: 15017

Version 1.0 20/04/2015
Studerende: Søren Vøgg Krabbe Lyster (SVL) 10920,
Martin Degn Kristensen (MDK) 10441
Studieretning: Elektro
Vejleder: Preben Kidmose

Revision History

Revision	Date	Author(s)	Description
1.0	20.04.15	SVL,MDK	Oprettet

Indhold

1 Indledning

For at oprette en software arkitektur der overholder de krav stillet i kravspecifikationen, er der gjort brug af UML (Unified Modelling Language) [1]. UML tillader en tilnærmelsesvis direkte omskrivning af krav opstillet som use cases, til UML-diagrammer der skitserer en software arkitektur. Da der i kravspecifikationen er gjort brug af UML til udarbejdelse af de forskellige use cases, har det derfor været muligt af skrive sekvensdiagrammer ud fra hver enkelte use case.

Sideløbende med udviklingen af sekvensdiagrammerne er de forskellige klasser blevet forfattet. Følgende afsnit beskriver grundlæggende tanker og argumentation for valget af klasser.

2 Klasser

2.1 UserInterface

Denne klasse foretager al kommunikation med aktøren *bruger* igennem et grafisk user interface (GUI). For at have en konstant opdatering af interfacet, forventes denne klasse at blive afviklet i egen tråd.

2.2 LogHandler

Denne klasse håndterer eye-tracking-programmets log-fil. Data fra eye-tracking bliver her pakket og gemt i en log-fil. Kommunikation af målings-relevant data til UserInterface-klassen foretages af denne klasse.

2.3 SessionHandler

Her håndteres al data tilknyttet opsætning af eye-tracking-session.

2.4 EyeTracking

Denne klasse håndterer kommunikation og instantiering af klassen ETAlgorithm.

2.5 ETAlgorithm

Denne klasse er kernen af eye-tracking-systemet. Her foretages alle udregninger med henblik på eye-tracking. Denne klasse ønskes isoleret så meget som muligt fra resten af systemet, således at der altid kan foretages ændringer i klassens interne kode, uden at det påvirker resten af systemet. Konsistente grænseflader for denne klasse er derfor essentielt. ETAlgorithm bliver håndteret af klassen EyeTracking, modtager et videosignal, og returnerer XY-koordinat samt trigger-niveau.

3 Miljø

Dette afsnit har til formål at klarlægge det valgte programmeringsmiljø, og argumenterne for valget af dette. Programmet kan med fordel deles op i tre typer klasser:

1. Grafisk bruger interface.
   Herunder er klassen UserInterface.

2. "Handlers"til håndtering af data og funktionskald.
   Herunder er klasserne LogHandler, SessionHandler og EyeTracking.

3. EyeTracking algoritmen.
   Herunder er klassen ETAlgorithm.

3.1 Python

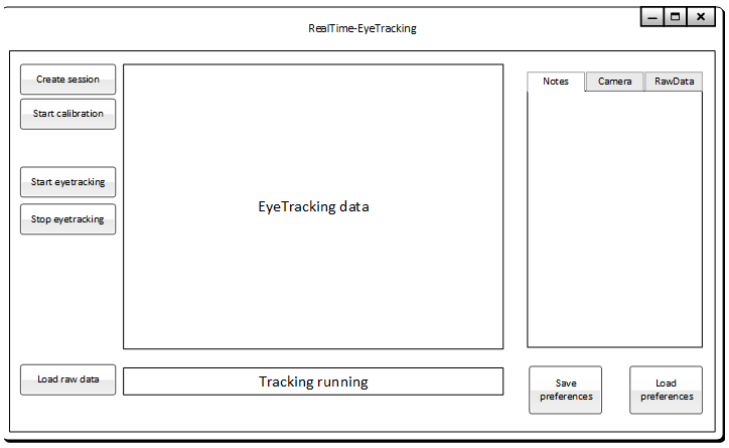
Matplotlib [2]

3.2 OpenCV

OpenCV [3] er en række "open source"biblioteker med kraftfulde billedbehandlingsalgoritmer. Herfra benyttes klassen VideoCapture, som giver adgang til et systems forskellige kamera-kilder.

3.3 Wrapping

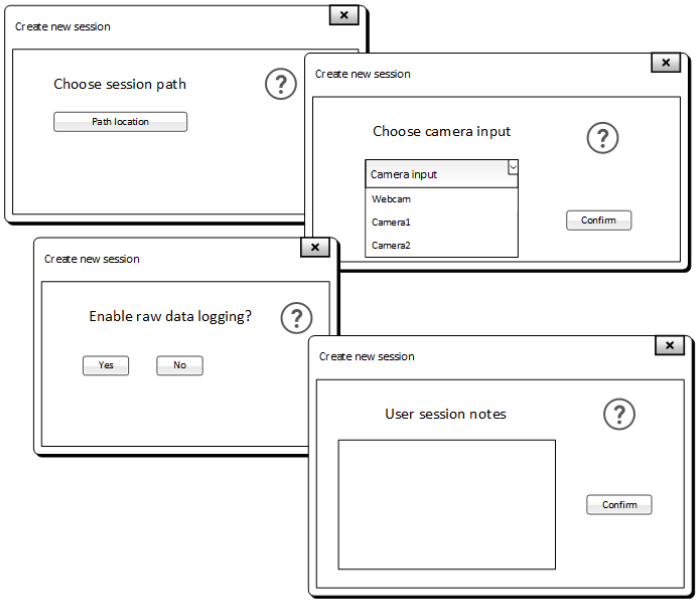
OpenCV tillader



Figur 1: Udkast til grafisk bruger interface

4 Grafisk bruger interface

Følgende figurer beskriver det forventede grafiske bruger interface. Interfaces er designet med henblik at overholde use case diagrammerne beskrevet i kravspecifikationen.



Figur 2: Udkast til menuer der guider bruger igennem opsætning

Litteratur

[1] O. M. Group, "Unified Modelling Language." `http://www.uml.org/`, 2015. [Online; sidst tilgået 20-04-2015].

[2] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[3] G. Bradski *Dr. Dobb's Journal of Software Tools*.