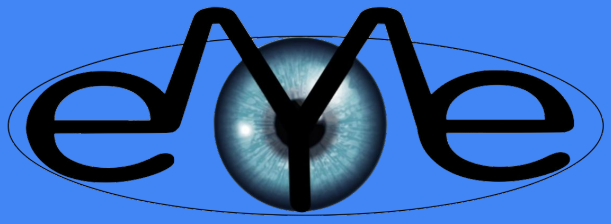
MyEye



Eye Capture Window Manager

Group 31

Intro

Contents

Introduction

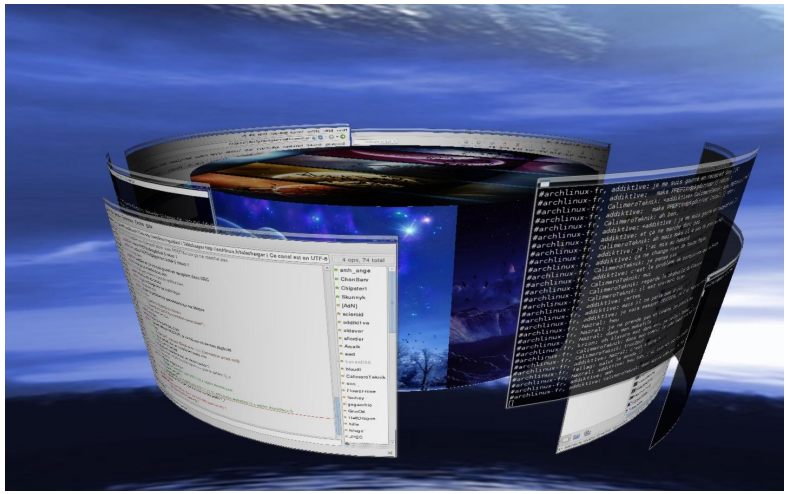
Basic Characteristics of MyEye

- MyEye is a window manager navigation tool. It uses the head position to change window focus.
- Make navigating windows on a Linux OS easier and faster.
- Runs in the background, but has a UI that can be used to utilize some settings.

Background - Window Management

- What is a window manager?
  - “System software that controls the placement and appearance of windows within a windowing system in a graphical user interface.” -https://en.wikipedia.org/wiki/Window_manager

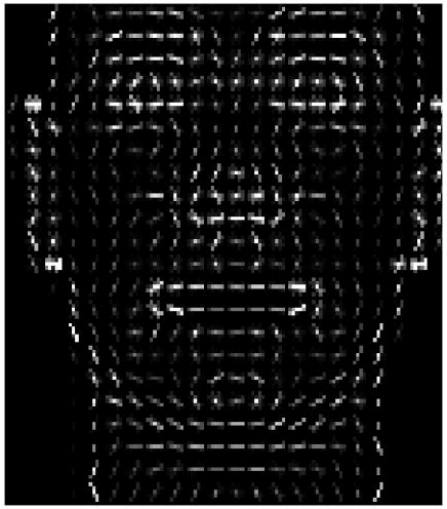
- X window system
  - Client-server model
  - Xlib



Background - Eye Gaze and Face tracking

- Viola-Jones techniques

- Histogram of oriented gradients



Background - Motivation

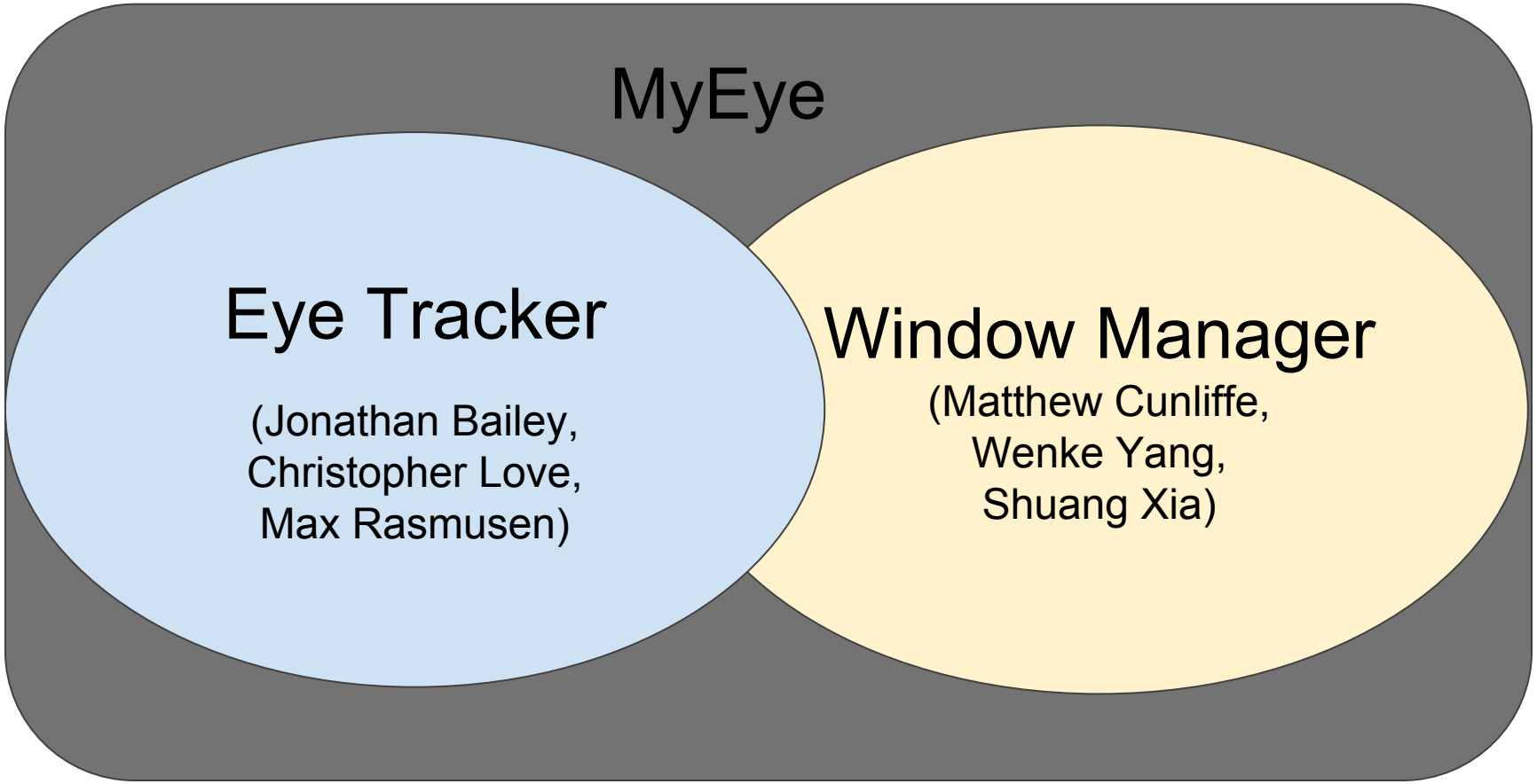
- Regular and Power Users
- Accessibility - provide a benefit to users who have trouble using the standard window focus method

Objectives

- Primary objective
  - switch focus
- Stretch goals
  - accuracy
  - precision
  - additional features
  - more screens
  - unrestricted configuration

Project Management

Work Allocation



Group Management

- Whole Group Meeting

  weekly in lab, add Kanban feature, review previous work, schedule sub-group meeting, meet supervisor

- Sub-group Meeting

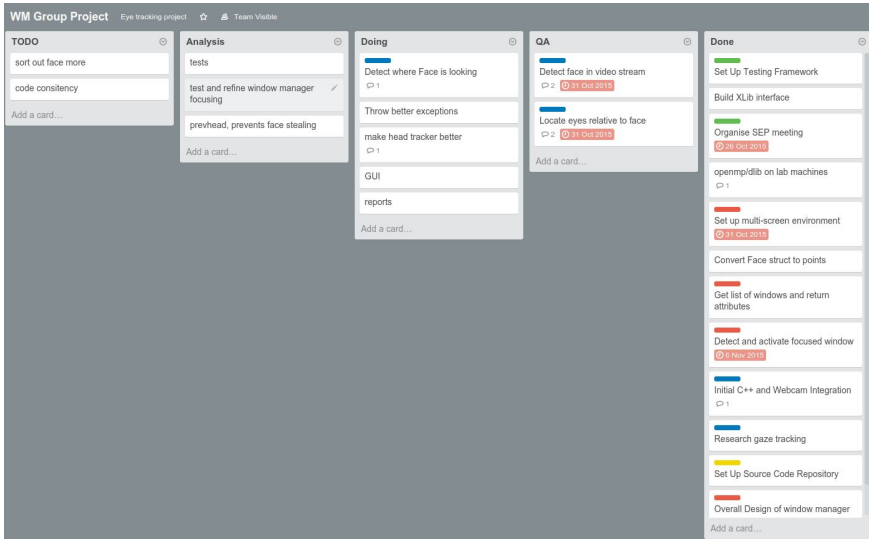
  more frequent, quicker to organise, collaborate when meet problem, in lab & online

- Supervisor Meeting

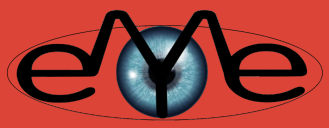
  every couples of weeks, show supervisor new features, discuss how to improve, receive advice

Kanban

- Just-in-time delivery, software slowly evolves over time
- Follow to-do, analysis, doing, QA, done cycle
- No set work period with certain feature
- Constant development, use Kanban board to track features through workflow
- Use Trello as electrical Kanban Board



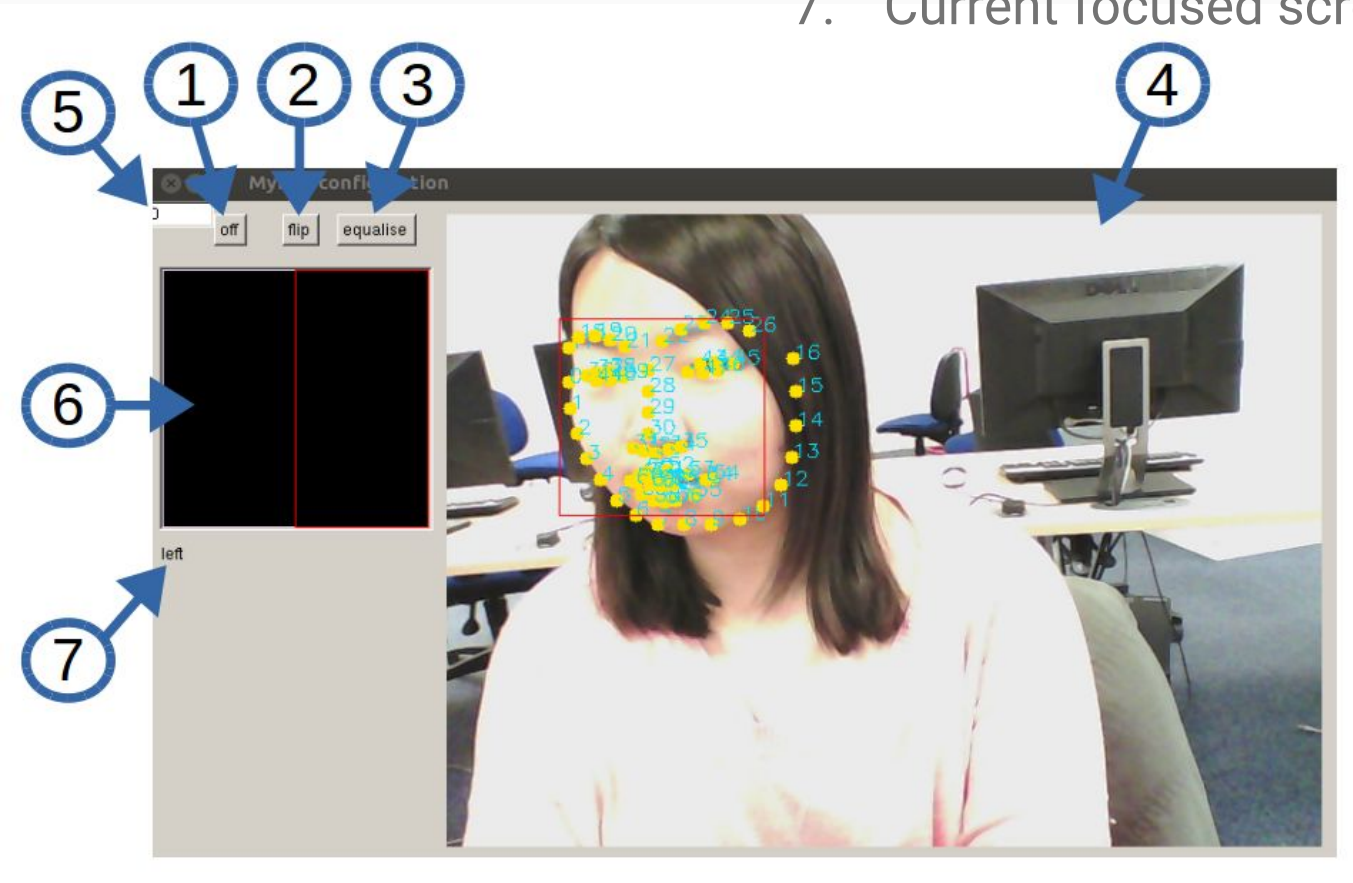
Team Collaborate Technology



Design

Front-End Design

1. ON/OFF button
2. Flip button
3. Equalise button
4. Live video
5. Camera selection
6. Screen setting
7. Current focused screen



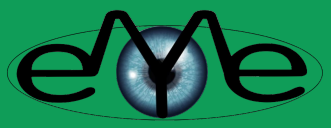
Back-End Design

Technical Challenges

- Users have preferred window manager
  - Interface with X server directly, instead of building on a specific window manager's API.
- Hard to detect correct direction of gazing
  - Switch from OpenCV's Viola-Jones technique to dlib's landmark detection algorithm.
- Software ran slowly
  - Minimise search area during feature detection

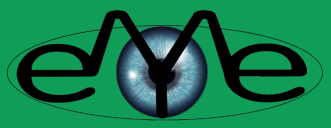
Implementation

Window Management



- `X Server`
  - Client - Server model
- `Xlib`
  - Attributes: information about the window
    - `XWindowAttributes` struct
  - Root Window: contains all other windows
- `Xrandr`
  - Resize and rotate X window extension

Window Management



- Determining screen sizes
- Use Xrandr to get screen sizes
  - `XRRScreenResources, XRRGetCrtcInfo`
  - Unable to do this through `Xlib`
    - Has no concept of physical screens
  - Can calculate centres of screens
    - Used to switch between screens

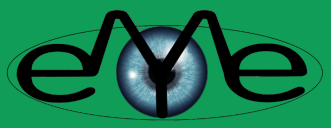
Window Management

- Determining Client Windows
  - root window
    - Tree of windows is queried and returned
      - list of windows
      - size and locations of them (Attributes)

Window Management

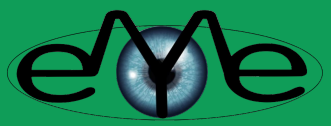
- Switching Between Client Windows
  - Message Events
    - Send message to root window of the display
  - Message specifies:
    - Window
    - `_NET_ACTIVE_WINDOW`
    - Masks showing type of event

Window Management



- Focuses topmost window at centre of desired screen
  - `map_state == IsViewable`
    - Only viewable windows
- Focus window using `XSendEvent` to send an event to the X server.
  - X Events conflict with permissions from other window managers.
  - Focus our own window to circumvent permissions.

Vision



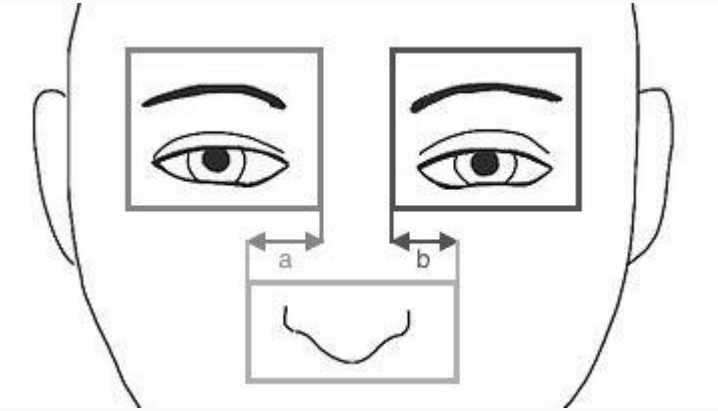
- Input
  - `FileInput`, `CameraInput` inherit from abstract class `Input`
  - `getFrame` function: returns struct `Frame`
- `Frame` contains the OpenCV matrix and has methods including overloaded casting operators so it can be easily used in dlib or OpenCV functions

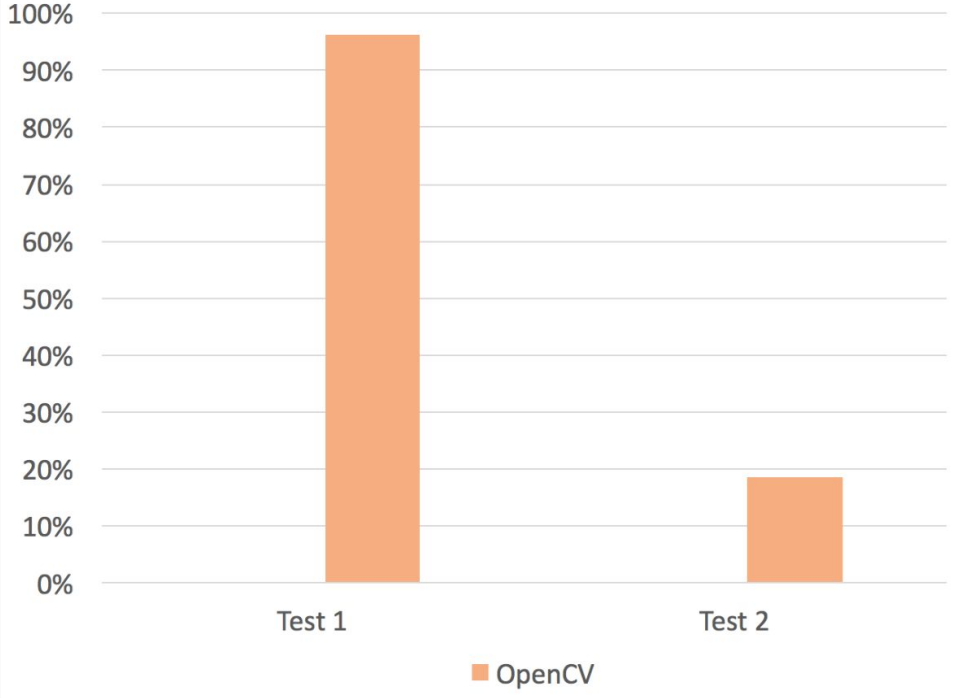
Vision - Feature Detection

`FeatureTracker` class
`getFeatures` method returns struct: `Face`

Initially: (OpenCV)

- Viola-Jones technique
  - Haar cascades
- Face struct:
  - Rects for eyes, nose and face



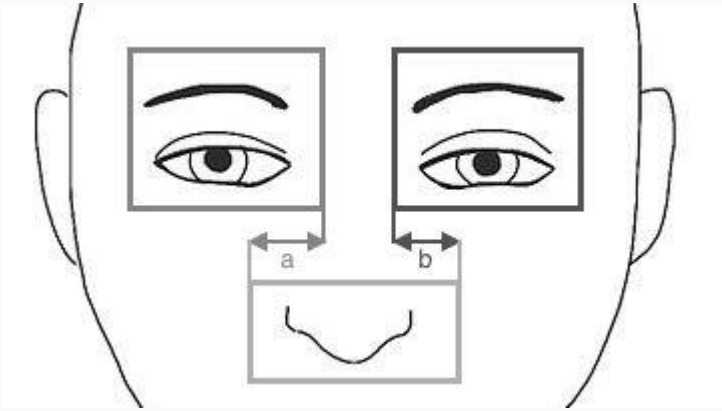


Vision - Feature Detection

`FeatureTracker` class
`getFeatures` method returns struct: `Face`

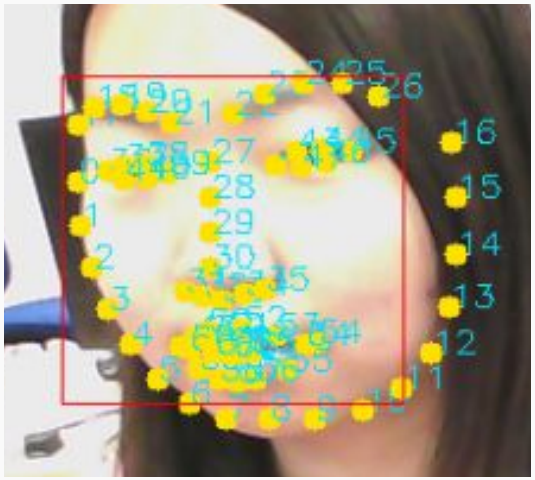
Initially: (OpenCV)

- Viola-Jones technique
  - Haar cascades
- Face struct:
  - Rects for eyes, nose and face



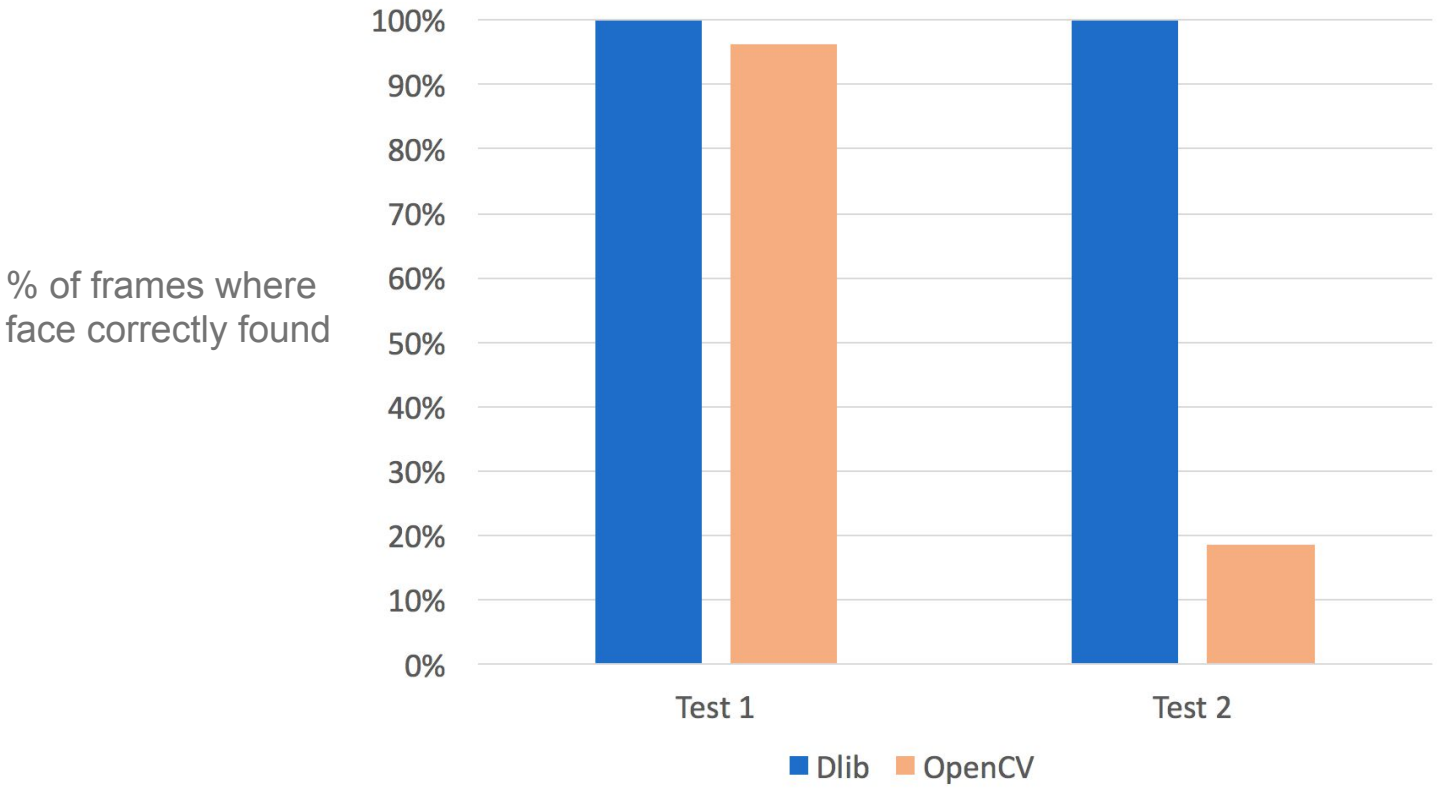
Now: (dlib)

- Histogram of Oriented Gradients
- Face struct:
  - 68 facial landmarks (points)
  - Rect for face



Vision - Feature Detection

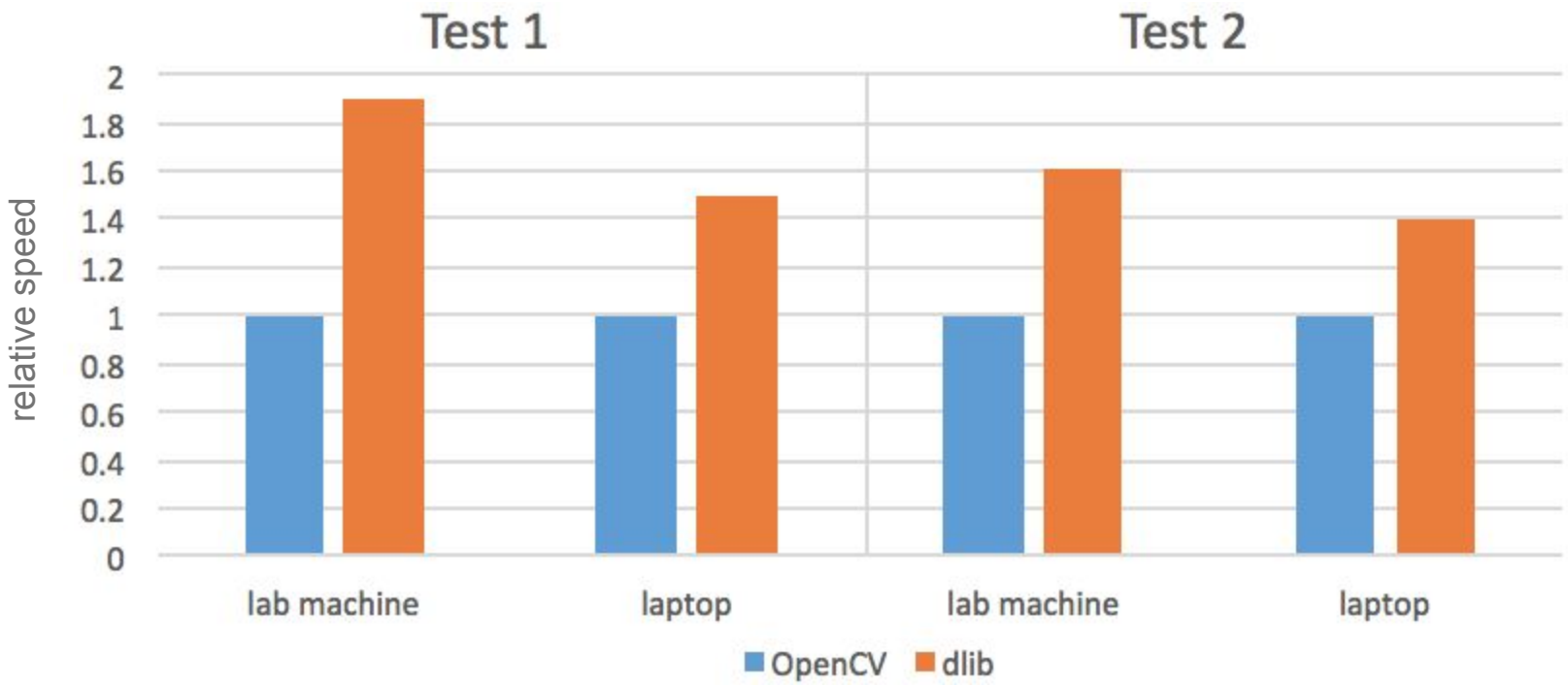
Comparing performance of `FeatureTracker` implementations



Vision - Feature Detection

Relative time to process test videos

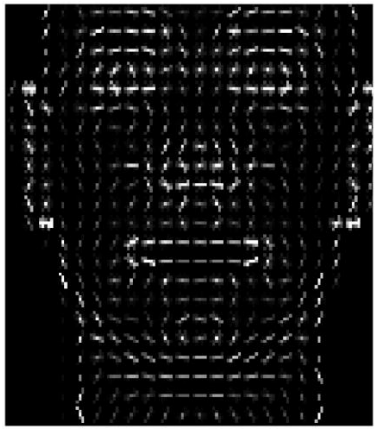
Each column is an average of 3 runs in similar conditions



Vision - Feature Detection

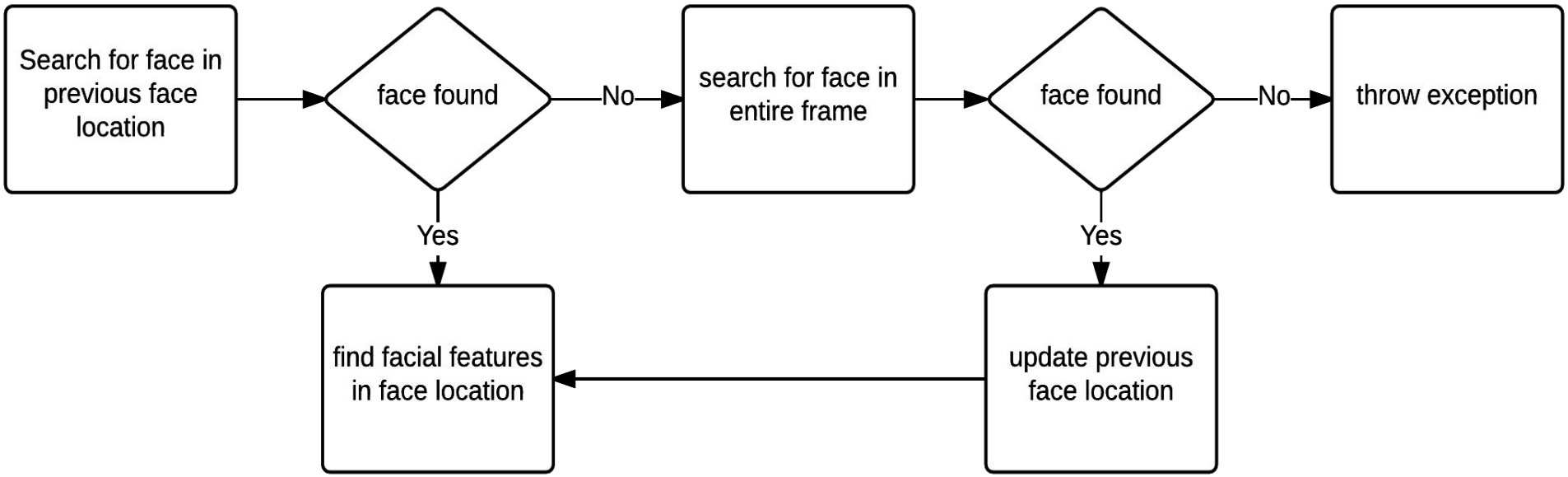
Final method:

- dlib - `frontal_face_detector` class
  - gets the rect of the whole face
- Histogram of Oriented Gradients technique
  - `shape_predictor` (facial landmarks)



Vision - Feature Detection

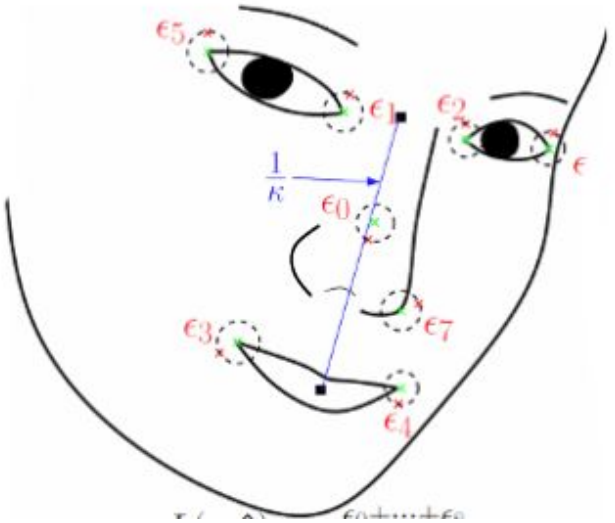
Feature detection algorithm



Vision - Head Direction

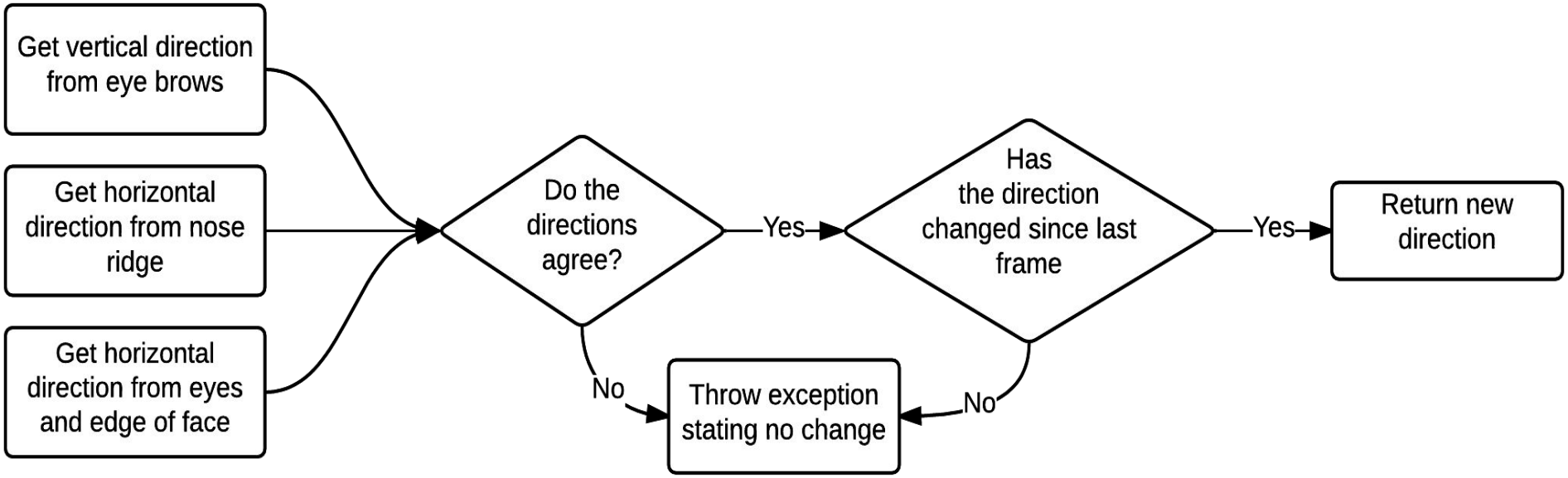
Determining Head Direction

- HeadTracker class
  - Looking for obvious variation in landmarks
  - Two methods for horizontal direction:
    - eyes and edge of face
    - bridge of the nose
  - Extension to vertical direction:
    - Eyes and Eyebrows
  - Voting



Vision - Head Direction

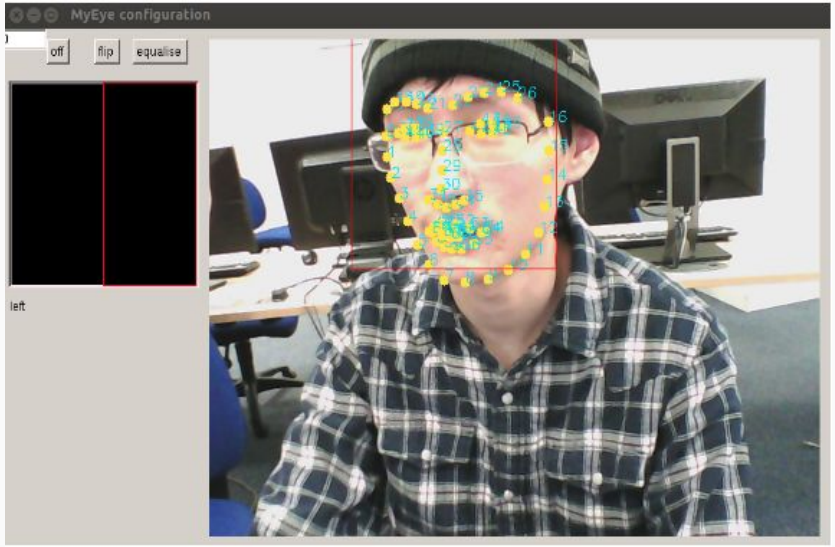
- HeadTracker class



Demo Video

Evaluation

Unit Testing & Integrated Testing




```
TEST(window_manager, can_repeatedly_set_focus_screen_left){

    wm w_m;

    for (auto i = 0; i < 5000; i++) {
        w_m.set_focus_screen(wm::Left);
    }

}

TEST(window_manager, can_repeatedly_set_focus_screen_right){

    wm w_m;

    for (auto i = 0; i < 5000; i++) {
        w_m.set_focus_screen(wm::Right);
    }
}
```

googletest

Google C++ Testing Framework

Project Evaluation

- Close to 100% precision in direction detection
- Compatible with existing window managers
- High time efficiency
- Robust towards gender, ethnicity, camera

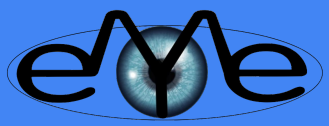
- Only works for two screens in current stage
- Window management depends on platform (currently requires X)

Conclusion and Future Extension

Future Work

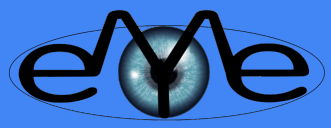
- Machine learning to detect direction(not just landmarks)
- Develop softwares/games based on the whole MyEye system(e.g. multi-screen based games)
- Increase accuracy of Eye capture function(track pupils)
- Develop softwares/games based on the Eye capture part(e.g. anti-fatigue driving system, smart home systems)

What We Learnt: Project Management



- Made our product releasable at every stage
- Used Trello(kanban) to keep track of the process at every stage of development
- Applied a divide and conquer method to manage people and project.
- Managed according to differing levels of expertise and interest. (ie only some of the group studied computer vision).

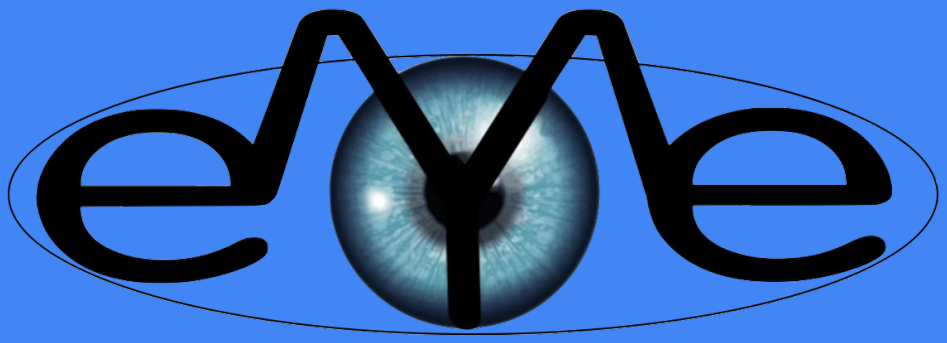
What We Learnt: Technically



- Used Xlib, openCV, dlib
- Practised programming in C and C++
- Utilised CMake tool to build a compilation environment
- Became familiar with the GoogleTest C++ unit testing framework
- Employed test-driven design patternless to make sure the program achieved expected features

Conclusion

- MyEye automatically switches focus according to head movement of users
- Compatible with existing window managers
- High run time efficiency of traditional computational intensive software
- Improved working efficiency of users



Thank you!