Opengazer: headtracker installation guide

Emli-Mari Nel
Cavendish Laboratory
University of Cambridge
Cambridge
CB30HE
en256@cam.ac.uk

25 May 2010

1 Introduction

This is the installation guide for the head-localisation component of Opengazer (see Figure 1). Research has been done on each of the components depicted in Figure 1. We hope to make more frequent software releases in the future. Note that at present, I am the sole researcher and software developer on this project - feedback on emails may be somewhat slow depending on demand.

2 Algorithm overview

The goal of this head tracker is to locate the largest face in the video stream (captured from a file/camera) as fast as possible, on a frame-by-frame basis. The xy-coordinates from tracking can already be used to type using Dasher. This can be done in 1D mode (e.g., from tracking just the y-coordinates), or in 2D mode. Although much better results can be expected after the release of our head-pose software, this software is already useful for fast face localisation.

The head-localisation algorithm described in this document is an elementary algorithm, based on the Viola-Jones face detector [1]. We make extensive use of the OpenCV library [2]. Our algorithm applies a simple autoregressive lowpass filter on the xy-coordinates and scale of the detection results from the Viola-Jones face detector, and also restricts the region of interest from frame to frame. The detection parameters have been determined according to our specific application. The algorithm detects and tracks faces at 30 frames/s on 320x240 images, and can handle rotations of up to 30 degrees (pitch, roll and yaw).

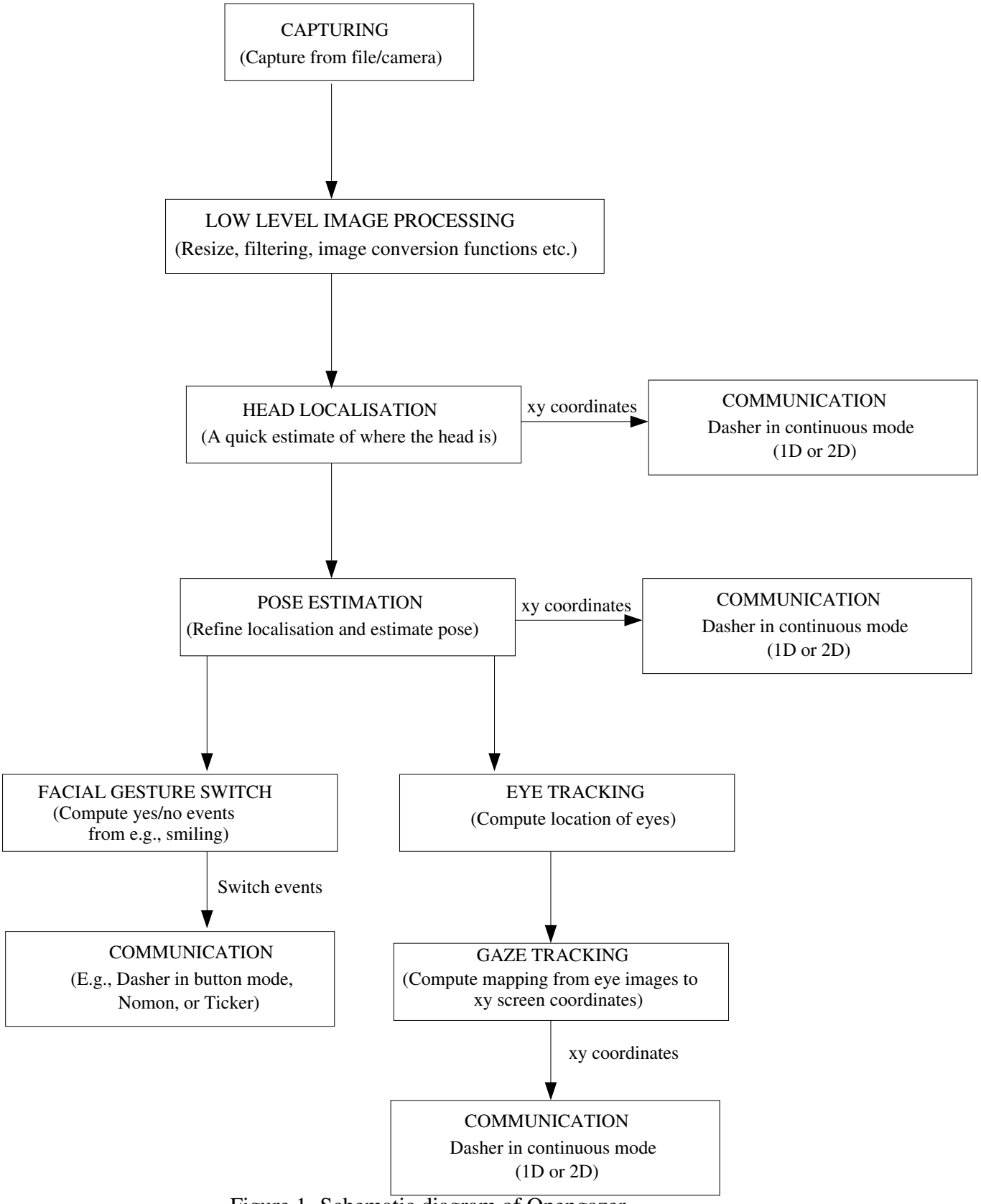

Figure 1. Schematic diagram of Opengazer.

The algorithm has been tested on Ubuntu Hardy, Jaunty and Lucid using a Logitech QuickCam 4000 Pro USB webcam. The main dependencies include OpenCV (with python wrappers), Qt (with Python wrappers), and Python. These are all crossplatform libraries. Hence, the software should work on Windows, Mac, and Linux. This guide only includes installation instructions on

Linux. Any comments on successful installation on other platforms will be more than welcome and added to our *howto* guide.

3 Capturing unit

Dependencies which need to be installed (tested on Ubuntu 10.04 (Lucid Lynx)):

- Python version 2.6.5-0ubuntu1
- Numerical Python version 1:1.2.1-1ubuntu1: *apt-get install python-numpy*.
- Python binding for OpenCV version 2.0.0-3ubuntu2: *apt-get install libcv4, python-opencv*. (We had to install earlier versions of OpenCV from source, and fixed a few bugs before all the necessary functionalities worked - this was specifically on Ubuntu Jaunty. This is not necessary for version 2.0.0, as tested on Ubuntu Lucid Lynx).
- Python binding for qt4 version 4.7.2-0ubuntu1: *apt-get install python-qt4*.

Make sure that the *Video4Linux* driver is compatible with your camera (OpenCV relies on these drivers), and make sure that your camera can capture at least at 30 frames/s. One way to verify this is to

- use *ekiga* and selecting the V4L drivers (after *apt-get install ekiga*). Adjust the camera settings, to make sure the image quality is as good as possible. Close ekiga and then,
- use the command *setpwc -f 30* to capture at 30 frames/s (after *apt-get install setpwc*). The images on my camera are then automatically resized to 320×240 to be able to capture at this framerate.

4 The algorithm

The head-tracker interface is depicted in Figure 2.

- To start the program, type *python tracker.py* in a terminal.
- The program will automatically start to track the head. (Experiment to obtain the largest range of motion).
- To activate cursor control, check the *Activate cursor control* check box (see Figure 2). Note that the program takes control of the mouse - press Escape to quit this mode of operation. Vary the x and y gain scrollbars to control how much influence the x and y translations will have. For example, to control only the y mouse coordinate, set the x gain to zero. Doing so, one can easily type in Dasher in 1D mode. Note that even though the cursor coordinate results are noisy - it is already good enough to type in Dasher (where more sophisticated smoothing occurs).
- Data can be recorded by pressing on the *Record* button (see Figure 2).
- The algorithm can be tested on pre-recorded data by clicking on the *Display .avi* button and selecting an appropriate .AVI file. The algorithm will then operate on the selected file.

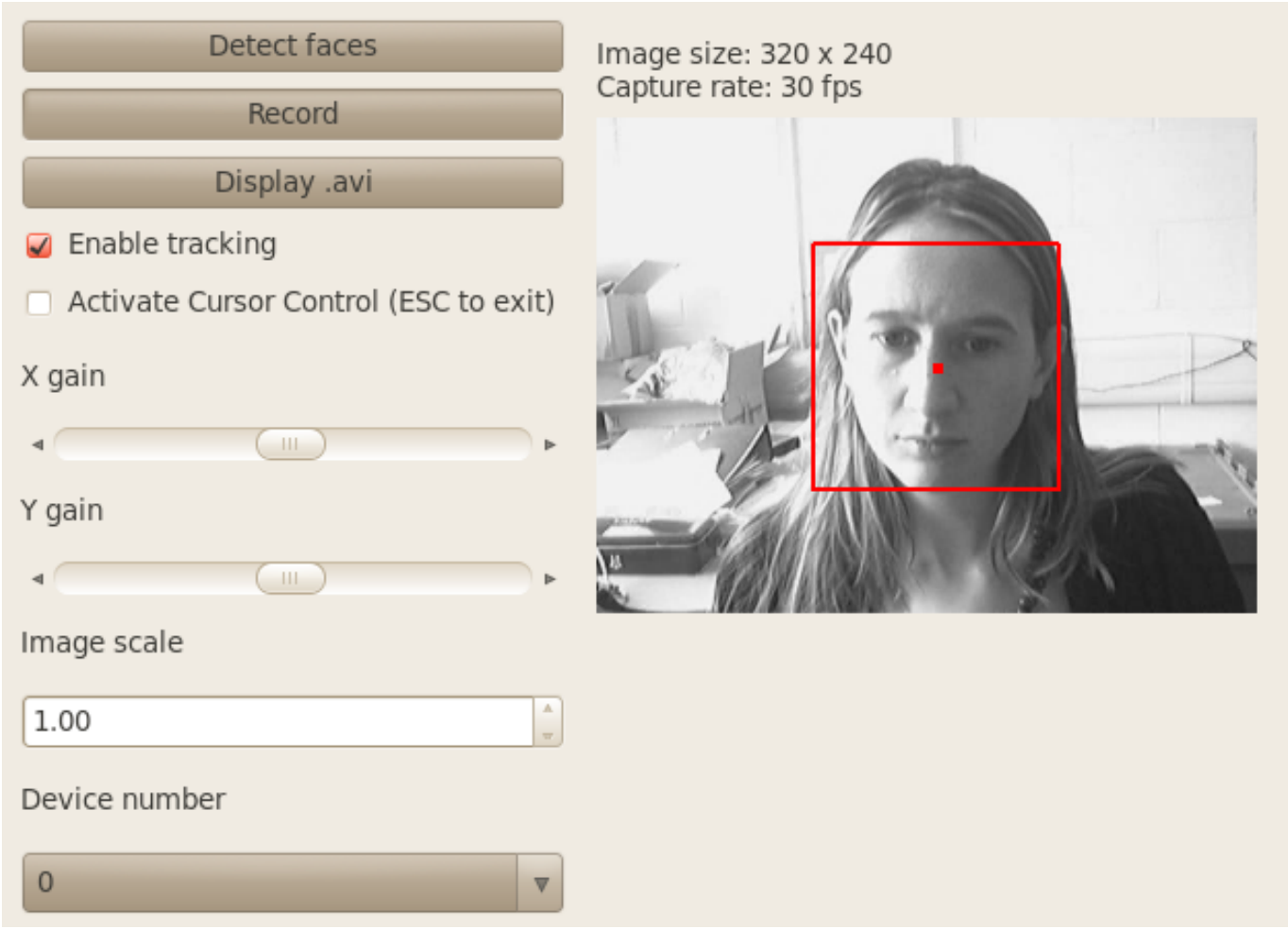

Figure 2. The head tracker interface.

- The rate at which the application is processing frames is displayed above the image (see Figure 2). If this rate is significantly lower than the frame rate you have set on the camera, then the program is not able to process images of the selected size rapidly enough on your processor. So, to increase the frame rate of the application, you need to reduce the *image scale*. Please note that the algorithm works best on 320×240 images, at a frame rate of 30 fps, and reasonable lighting conditions (e.g., as shown in Figure 2).

- If you would like to test another camera, e.g., your laptop's built-in camera produces low quality images, and you would like to use a better webcam like the Logitech QuickCam, select the appropriate *device number* . This device number can be found by connecting the external camera to the usb port and reading the result from *dmesg*. The output from *dmesg*, where the camera corresponds to device number 1, will look something like this:

```
[40869.993187] pwc: Logitech QuickCam 4000 Pro USB webcam detected.
[40869.993289] pwc: Registered as /dev/video1.
```

5 Current work, and upcoming releases

Project	Description	Status
Head tracking from single frame	Tracking small motions when the Viola-Jones tracker does not work. The user selects a region of interest to track (e.g., if the Viola-Jones face detector doesn't work the user can select a rectangle around the user's face/eyes to indicate the region of interest). This is useful, for example, if the user wears a head-band, or sits in a wheel chair that covers the neck. This algorithm will then track the region of interest from frame to frame. This algorithm is prone to drift as it is based on data from only one frame (the selected region of interest). However, it is good for tracking small motions.	Algorithm complete, preparing release code.
Facial gesture switch	An algorithm has been developed to detect facial gestures after a quick training phase.	Algorithm complete, preparing release code.
Ticker	An audio keyboard interface that can be used to communicate through the facial gesture switch. This program can accommodate noisy clicks (e.g., false positives/negatives from the gesture switch device), and is especially suited to visually-impaired single-switch users.	Algorithm complete, preparing release code.
Head tracking larger motions	Extending our current head trackers to detect faces for larger pose variations (between 15 and 45 rotations).	Current research
Accurate pose estimation	Refining the pose estimation after head localisation - eye tracking is part of this research.	Current research.
Gaze tracking	Tracking the direction of the gaze from images of the eyes (i.e., after eye localisation).	Initial prototype by Piotr Zielinski is available [3]. Research on this topic will resume at the end of June/July.

References

[1] P. Viola, M. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2) (2004) 137–154.

[2] G. Bradski, A. Kaehler, Learning OpenCV:Computer Vision with the OpenCV Library, O'Reilly Media, Inc, 2008.

[3] P. Zielinski, Opengazer: open-source gaze tracker for ordinary webcams, http://www.inference.phy.cam.ac.uk/opengazer/ (2007).