Universidade Federal
do Rio de Janeiro
Escola Politécnica

WEBCAMPAPERPEN: A LOW-COST GRAPHICS TABLET

Gustavo Thebit Pfeiffer

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

Orientador: Ricardo Guerra Marroquim

Rio de Janeiro
Março de 2014

WEBCAMPAPERPEN: A LOW-COST GRAPHICS TABLET

Gustavo Thebit Pfeiffer

PROJETO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO.

Examinadores:

Prof. Ricardo Guerra Marroquim, D.Sc.

Prof. Antonio Alberto Fernandes de Oliveira, D.Sc.

Prof. Fernando Gil Vianna Resende Junior, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2014

Pfeiffer, Gustavo Thebit

WebcamPaperPen: A Low-Cost Graphics Tablet/Gustavo Thebit Pfeiffer. – Rio de Janeiro: UFRJ/POLI – COPPE, 2014.

XI, 51 p.: il.; 29, 7cm.

Orientador: Ricardo Guerra Marroquim

Projeto (graduação) – UFRJ/ Escola Politécnica/ Curso de Engenharia de Computação e Informação, 2014.

Bibliography: p. 50 – 51.

1. vision-based user interface. 2. paper-pen technology. 3. tracking. I. Marroquim, Ricardo Guerra. II. Universidade Federal do Rio de Janeiro, Escola Politécnica/ Curso de Engenharia de Computação e Informação. III. Título.

Acknowledgements

I would like to thank my family, for their unconditional love and support for all these years.

All my teachers and professors, for opening a world of knowledge and research to me, and for their dedication in such honorable profession.

All my friends, for all the fun we have had together, all the company, for staying by my side when I needed, and for forgiving me after our undesirable misunderstandings.

I thank particularly my friend and colleague Marcello Salomão, for allowing me into his adventurous project and my supervisor Prof. Ricardo Marroquim, for his attention, his perceptive recommendations and his patience with me. Not forgetting Profs. Antonio Oliveira and Fernando Gil for their disposition to evaluate the project, and all the other ones who helped in any way the completion of this work, either with insights and opinions or testing the system. My special thanks to Leonardo Carvalho for the drawings.

Abstract of the Undergraduate Project presented to Poli/COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Computer and Information Engineer.

WEBCAMPAPERPEN: A LOW-COST GRAPHICS TABLET

Gustavo Thebit Pfeiffer

March/2014

Advisor: Ricardo Guerra Marroquim

Course: Computer and Information Engineering

We present an inexpensive, practical, easy to set up and modestly precise system to generate computer mouse input in a similar fashion to a graphics tablet, using a webcam, paper, pen and a desk lamp.

Keywords: vision-based user interface, paper-pen technology, tracking.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

WEBCAMPAPERPEN: UMA MESA DIGITALIZADORA DE BAIXO CUSTO

Gustavo Thebit Pfeiffer

Março/2014

Orientador: Ricardo Guerra Marroquim

Curso: Engenharia de Computação e Informação

Apresentamos um sistema de baixo custo, prático, de fácil configuração e modestamente preciso para controlar o mouse do computador de forma semelhante a uma mesa digitalizadora, utilizando webcam, papel, caneta e uma luminária.

Palavras-Chave: vision-based user interface, paper-pen technology, tracking.

Contents

List of Figures

Chapter 1

Introducing WebcamPaperPen

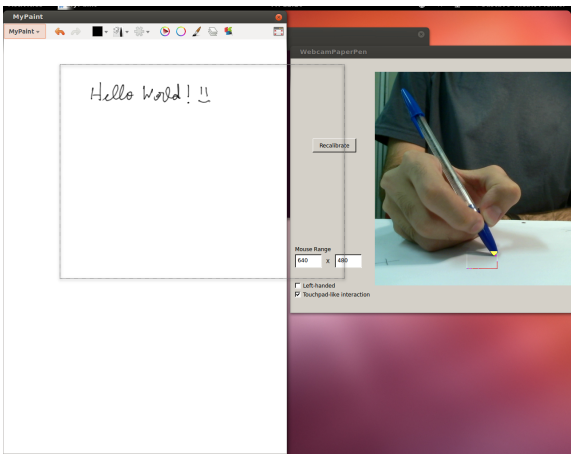
1.1 Introduction

In many applications such as handwriting and drawing, using the computer mouse as interface may be inappropriate or even prohibitive, and a graphics tablet would be desired. However, graphics tablets may be an expensive item for some users and requiring them to own one is not reasonable. There are also cases where the user would like to try the application before buying a proper graphics tablet.

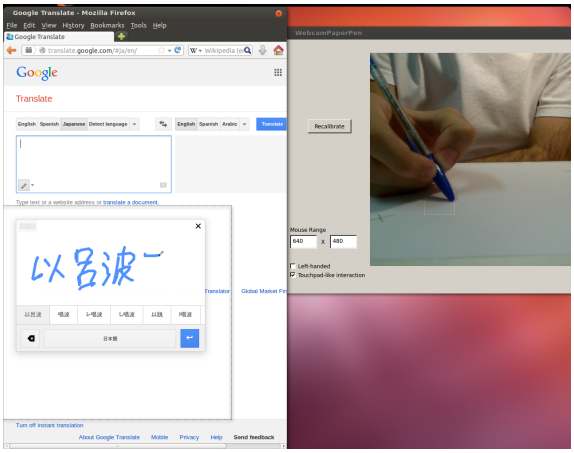
Aiming at this scenario we have devised a low-cost system to control the computer mouse similarly to a graphics tablet, to facilitate drawing and handwriting applications. Our method only requires a webcam, a sheet of paper, a blue-capped pen and a desk lamp: practical and easy to set up, not requiring building a special pen or a support for the camera or the lamp. It is precise enough for drawing applications and fast in the sense that it can reach a high FPS rate in a single-threaded implementation in a modern desktop computer, thus it will not require extra hardware and is not expected to impact the interactivity of the application.

1.2 System Description

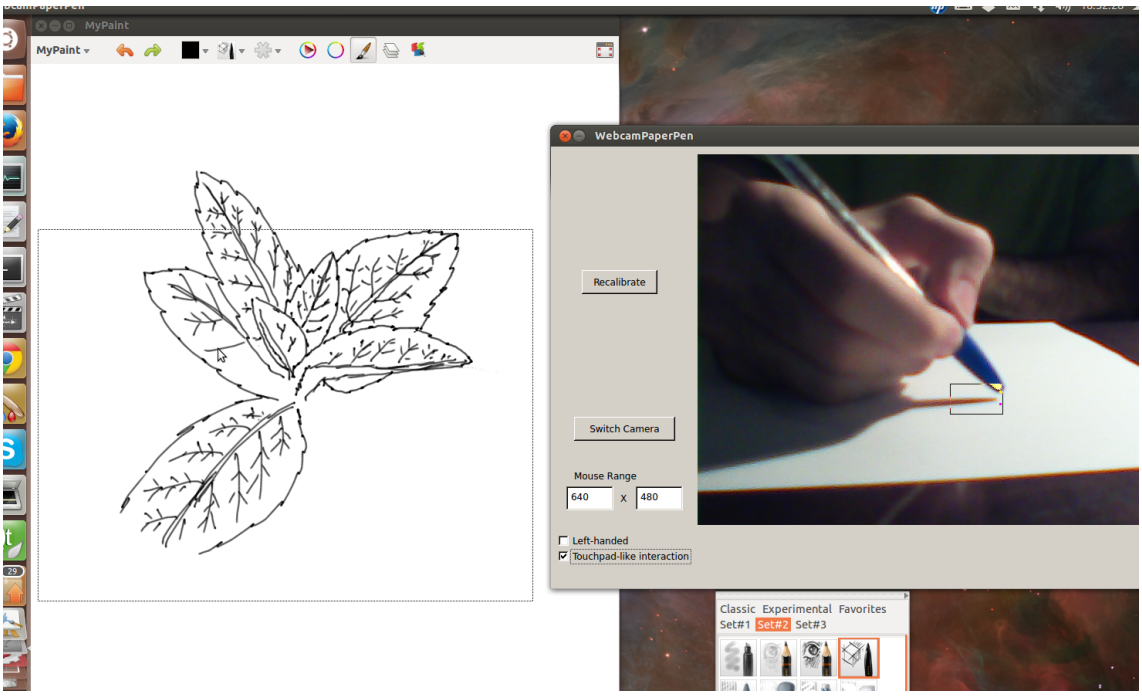
Our system is configured as shown in Figure 1.2: The user places a sheet of white paper over the desk and positions the webcam on the desk between the paper and the monitor only slightly above the paper, facing the user. The system is calibrated by drawing four crosses on the paper, indicating rectification corners. Mouse clicks are detected by analyzing the shadow of the pen, so it is often necessary to place a lamp on the left (supposing the user is right-handed). The pen must have a blue cap (ideally a common BIC blue pen), as the user will use the pen with the cap shut, never releasing ink on the paper. We have restricted the pen color to blue in order to minimize the interference with the user's hands and with the shadow, also



(a) Handwriting in MyPaint



(b) Inputting Chinese characters in Google Translate



(c) Drawing in MyPaint

Figure 1.1: Our system being used within several applications.

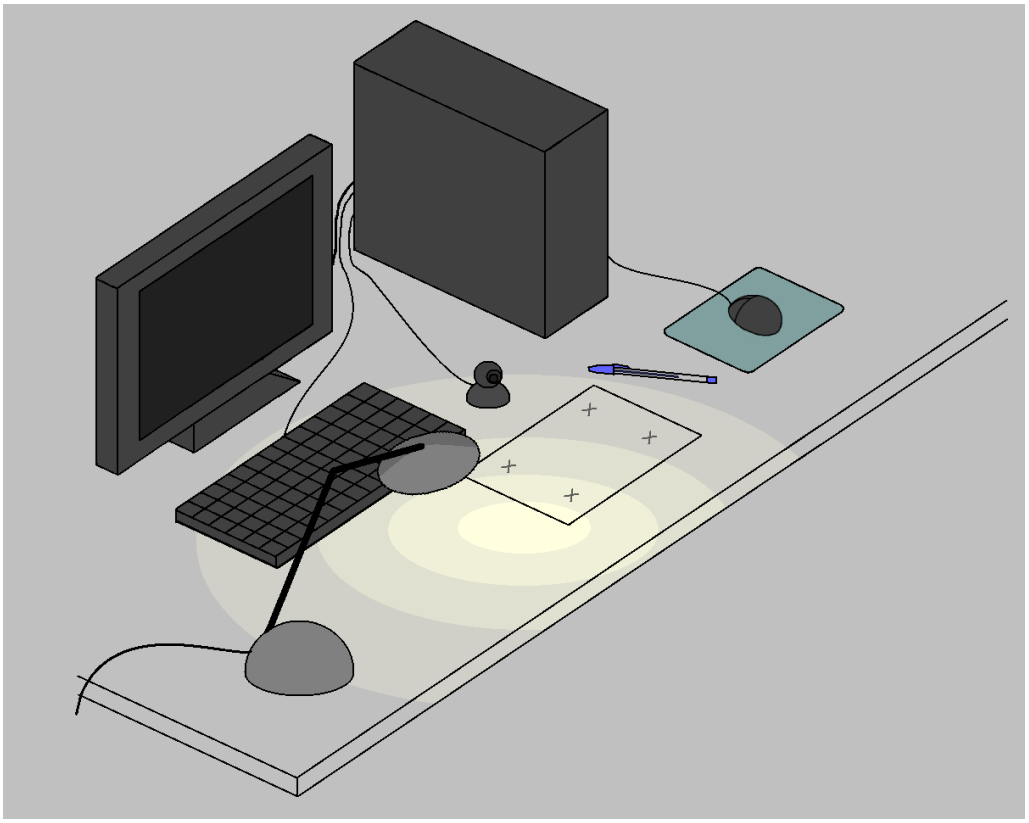


Figure 1.2: System setup illustration.

taking advantage of the ubiquitous character of this sort of pen.

There are a number of reasons for not letting the user write or draw (with ink) on the paper. First of all, graphics tablet users usually do not look at the tablet while drawing: The range of applications in which the user would be required to actually look at the paper (and not at the monitor) while drawing or writing is much more limited, as the user would not generally be able to interact with elements on the screen, unless they are, for instance, projected onto the paper. Not to mention that common drawing operations such as as erasing, moving, scaling, changing the brush or the color would not synchronize with what is drawn on the paper. Also, in most of those applications where the user does not need to look at the monitor, but only at the paper, the software response does not need to be in real time, i.e., one can draw and afterwards take a picture using the webcam and rectify, or film themselves drawing and then process the video. In any case, this would require completely different methods and would be application-specific. Secondly, detecting the blue cap is much easier and less time-consuming than detecting the pen tip, and ink is one more obstacle in making pen cap tip and shadow tracking algorithms correct and precise. A third reason is that, for a system that controls the mouse, permitting ink would consume a lot of paper, which is not something we would like to encourage.

The user interaction in our system is divided in two steps: The calibration step, when our method computes the rectification (homography) matrix (Section 2.2.2); and the drawing step, in which the pen cap and its shadow are tracked (Sections 2.2.3, 2.2.4, 2.2.5 and 2.2.6). As with a graphics tablet, the user can move the mouse cursor without clicking by moving the pen near the paper, without touching it. A limitation of our system compared to graphics tablets is the lack of touching pressure measurement.

Also differently from the graphics tablet, here, as all the processing is done on 640×480 images from the webcam capture, we do not have enough precision to control the mouse in screen resolution, so we limit mouse control to within a $W_F \times H_F$ window of space, to the which we will refer as "mouse range window". However, we have designed our tracking algorithms to have subpixel precision in image coordinates, so W_F and H_F can be set to resolutions larger than that of the webcam. Nevertheless, considering that the user will usually draw on an A4-sized sheet of paper, it is better in terms of user interface that this window is not much larger than 800×600. In this work we use by default 640×480.

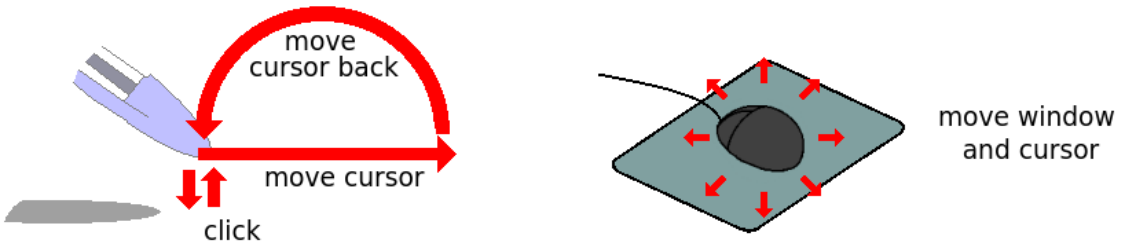
In order to move the mouse range window to reach the whole screen, we have devised two interaction modes in our software (see Figure 1.3). In the "normal" mode, the user moves the computer mouse in order to move the window, and the pen moves the cursor inside the window. In the "touchpad-like" mode, the user may raise the pen above a certain threshold (about 1cm) and return in a different position: This act will not move the cursor, and the window will be moved accordingly, thus enabling the user to reach the whole screen without using the computer mouse, by using the pen and the paper similarly to a touchpad from a note- or netbook. Some graphics tablet interfaces also provide this interaction mode.

1.3 Motivation

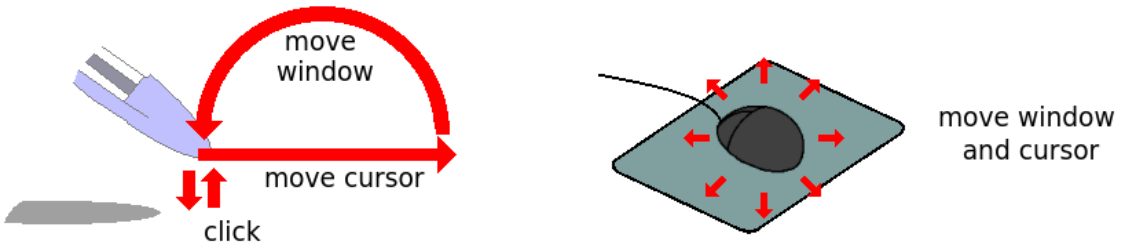
It is important to mention that the motivation to this work started with Marcello Salomão's educational project Libera Akademio.

Libera Akademio is a set of software tools and services to provide video lectures to the masses. The video lectures are similar to Khan Academy in style, i.e. they show a black board with handwritten text and illustrations, though LA uses its own, low bit-rate video format, and was designed in a way that anyone could create and publish their own video lectures.

However, in order to handwrite in this virtual black board, the system would require a graphics tablet, and we can expect that most of the potential video creators for LA would be reluctant to buy a graphics tablet only to create the video lectures. Therefore the system would have difficulties to be widely adopted. For this reason



(a) "Normal" mode



(b) "Touchpad-like" mode

Figure 1.3: Interaction modes from our implementation. Illustration shows how the mouse cursor and the mouse range window are moved by the mouse and the pen.

we started considering the idea of creating a software tool to complement LA by replacing the graphics tablet with a webcam and paper and pen.

Initially our idea was to let the user write (with ink) on the paper, and possibly post-process the video, but we eventually switched to the current approach as it can be used for a wider range of applications.

1.4 Related Work

There are a lot of similar works attempting to control the mouse with a webcam, but none with the same setup (color pen, paper, webcam and desk lamp), and none, to the best of our knowledge, achieving our balance of ease of use, low cost and precision.

There are many works [1] [2] [3] that create a human-computer interface using laser pointers and similar devices, but none using an ordinary blue-capped pen as ours. Works such as the one from Piazza and Fjeld [1] require building a complex device to set the webcam in an appropriate position (i.e., not as easy to set up as ours), while Lee's [2] requires the Nintendo Wiimote (i.e., has a comparatively higher cost) and Derhgawen's [3], which tracks the laser light on a surface, is inappropriate, in terms of user interaction, for drawing and handwriting applications.

There are also works [4] [5] with pen tip tracking (without the pen cap), though

designed specifically for handwriting applications such as signature recognition, and not for controlling the mouse cursor. These works allow ink on the paper and make no use of the shadow of the pen: Munich and Perona [4] detect touching using the ink path, which cannot be used for (instantaneous) mouse clicks (i.e., only dragging and dropping), must use some sort of post-processing (inappropriate for a real-time application as mouse control) and requires more complicated algorithms as an ink path is much weaker than a shadow from a strategically positioned light source; while Yasuda et al. [5] use two cameras and do not detect touching at all, they do signature recognition considering the movement of the pen in the air as part of the signature.

Finally, there is a number of works devising systems for people with disabilities and/or repetitive strain injury by tracking body parts such as the eye or the hand in the air [6] [7] [8] or a color pen in the air [9]. Although typically more practical and easier to set up (no calibration, fewer lighting constraints), they are not suitable, in terms of user interaction, for applications such as drawing and writing.

Chapter 2

Method and Development

2.1 Technologies Used

All the project was coded using C++, without parallelism except for the user interface, which was implemented in Qt [10]. To get the webcam image and to control the mouse we used, in Windows, respectively ESCAPI [11] and the Windows API [12], while in Linux we used OpenCV-HighGUI [13] and X11 [14] plus writing directly on the operating system files. Linear algebra operations were solved using Eigen [15].

2.2 Method Description

2.2.1 Notation and Preliminary Observations

An image is represented as three signals $R(x, y)$, $G(x, y)$ and $B(x, y)$ (i.e. the red, green and blue color channels), quantized in integer values between 0 and 255, with the origin $(0, 0)$ located at the top-left corner and the y axis oriented downwards, and $(x, y) \in ([0, 640) \times [0, 480)) \cap (\mathbb{Z} \times \mathbb{Z})$. We will refer to the sum $p(x, y) = R(x, y) + G(x, y) + B(x, y)$ as "intensity". We will conveniently use the abuse of notation $p(r) = p(r_1, r_2)$ for $r \in \mathbb{R}^2$ (by default the x coordinate is denoted as r_1 and the y coordinate r_2) or $p(u) = p(u_1/u_3, u_2/u_3)$ for $u \in \mathbb{R}^3$ (homogeneous coordinates [1]).

All frames are normalized to satisfy a mean pixel intensity of 252 in the area satisfying $40\% \leq y/H < 90\%$ (where H is the frame height, 480), which is the area (roughly) where the paper is normally placed. However, to save computation time, instead of modifying directly the image, we change all the thresholds of our method accordingly (i.e., all constants described in the following sections are defined

[1] See Appendix A.1

(a) (b) (c) (d) (e) (f) (g)

Figure 2.1: Illustration of the overall calibration algorithm. **(a)** Search the paper in the image using a hierarchical algorithm, yielding an estimate to the paper intensity. **(b)** Iteratively predict the intensity of the paper in each pixel as a quadratic function of the position. **(c)** Comparing the expected paper intensity for each pixel and the actual intensity of the pixel, classify the pixel as paper or non-paper. **(d)** Classify segments of non-paper areas as cross or non-cross following a few criteria. **(e)** Attempt to find the cross center by minimizing intensity after blur. **(f)** Update center iteratively using a quadratic fit. **(g)** Classify crosses.

supposing, without loss of generality, that the mean intensity is equal to 252), and only one of every 10 lines of this area of the image is included in the computation of the mean.

All the constants defined in this section were chosen applied to the common BIC blue pen, it is possible that other pens require different values. Also without loss of generality, the method is described supposing the user is right-handed.

2.2.2 Calibration

The objective of this step is to find four crosses drawn on the paper and estimate their centers. Ideally calibration and drawing should not be separate steps of our system, and our software should track crosses simultaneously to pen and shadow, recalibrating every time the camera or the paper was moved, but currently our method solves these problems in two separate steps. Having the paper accidentally moved should not be a problem as the user is not expected to look at the paper while drawing or writing. Also, recalibrating automatically could be complicated as the crosses are often occluded when the user draws. Our calibration step could also be replaced by manually selecting the crosses on the webcam image, however, this would hinder the ease of use of the system.

We chose the sequence of algorithms below and not a general-purpose feature detector because the cross on the paper is actually a very subtle feature that appears highly distorted and blurred due to perspective. The idea of our method for calibration is to discover first where the paper is and how it looks like, and then detect any marks that appear on it, no matter if they are cross-shaped or not.

The first step of cross search is a hierarchical algorithm to search the paper in the image (Figure 2.1(a)). We divide the image in 4 quadrants, and compute the mean

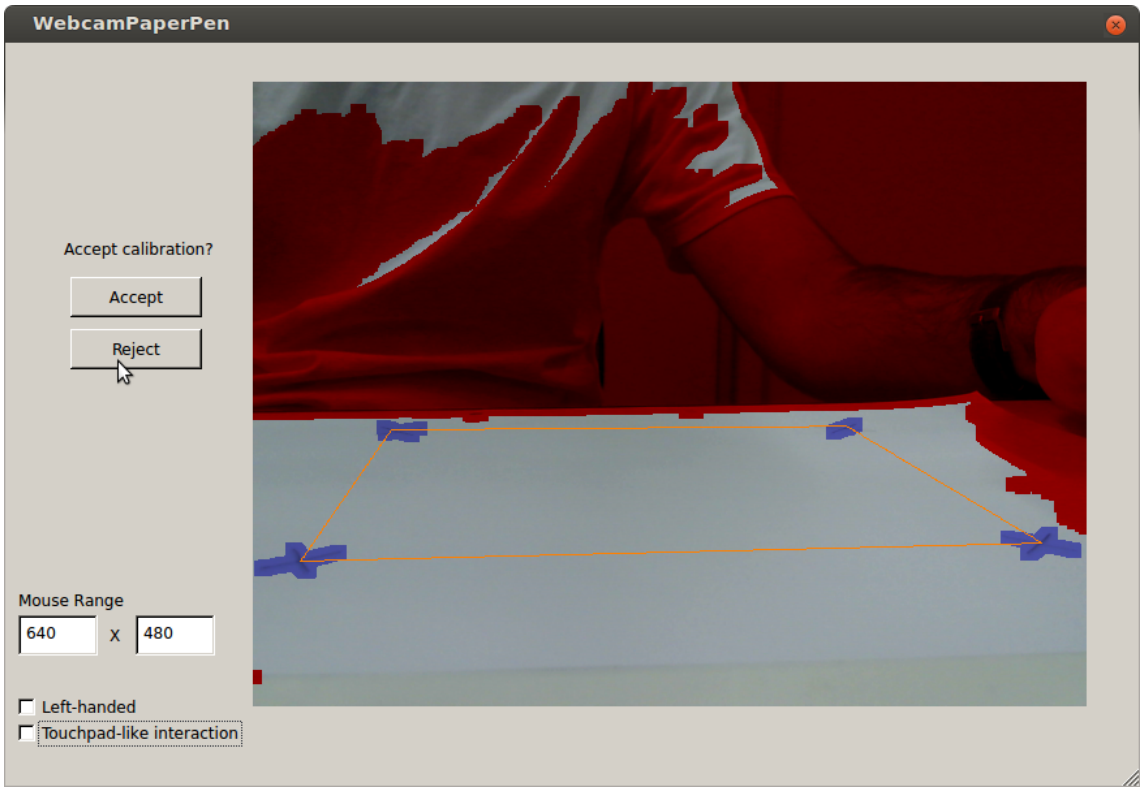
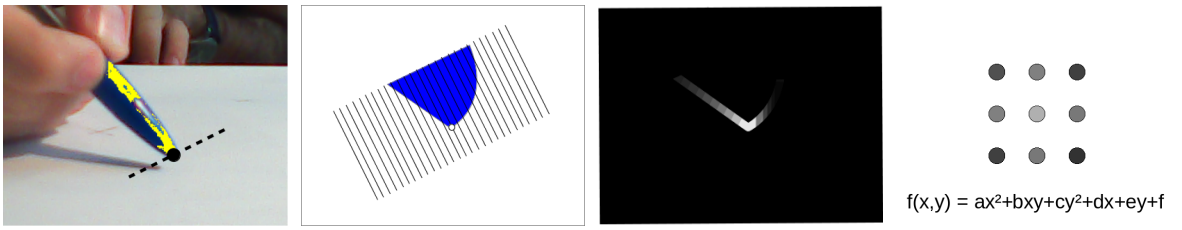


Figure 2.2: User interface for calibration. The user is shown the result of the cross classification step, with crosses highlighted in blue and unidentified regions in red (the rest is what was considered paper).

intensity of the pixels of each quadrant, yielding values μ_1, ..., μ_4. The quadrant of greatest mean is selected as the most probable location of the paper, and the algorithm is repeated for this quadrant. The algorithm stops when the variance of these means, $\epsilon^2 = \frac{1}{3}\sum_{i=1}^{4}(\mu_i - \frac{1}{4}\sum_{j=1}^{4}\mu_j)^2$, satisfies $\epsilon^2 < \frac{\sigma^2}{N/4}$, where N is the area of the four quadrants and $\sigma^2 = 0.17 \cdot 10^5$ is a multiple of the variance we expect pixel intensity to have in a region that contains only paper. At this point, $w_0 = \sum_{i=1}^{4}\mu_i/4$ is the initial estimate for paper intensity.

Then we compute the expected paper intensity for each pixel (Figure 2.1(b)), considering that the paper might not be homogeneously illuminated. We approximate it as a quadratic function of the position, in the form $w(x,y) = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^T R \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, for some right triangular matrix R. Initially we have $R_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & w_0 \end{bmatrix}$. We compute by linear regression $R_{i+1} = \arg\min_R \sum_{u\in\Omega_i}(u^T R u - p(u))^2$, where Ω_i is the set of pixels in the form $(x,y,1)^T$ satisfying $|u^T R_i u - p(u)| < 20$, which is our criterion to consider a pixel an inlier of this quadratic function. This procedure is repeated for 4 iterations.

Cross detection is done by a threshold-based segmentation algorithm (Figure 2.1(c)). A pixel is classified as paper if it is brighter than $0.92w(x,y)$, or



(a) Apply pen blue filter and maximize 2y+x

(b) Minimize column sum (horizontal tilted coordinate Δ_1) and maximize Sobel (vertical tilted coordinate Δ_2)

(c) Maximize objective function f

(d) Maximize fitted quadratic function from 9 pixels

Figure 2.3: Illustration of the overall pen cap tip tracking algorithm.

non-paper otherwise. This classification is dilated [2] (expanding the non-paper areas) using an 11×11 square filter, as crosses are drawn typically thin and are therefore vulnerable to disruption due to noise; then connected components ("segments") of non-paper areas are identified. Non-paper areas correspond mainly to: Crosses, noise on the paper and the background (i.e.: the table, the user, walls, etc.). Segments of non-paper areas can be distinguished as crosses (Figure 2.1(d)) if they respect the following criteria: 1) More than 50 pixels (so that they are not noise on the paper), 2) Less than 2000 pixels (so that they are not the background) and 3) the summation of $(\partial_x p)^2 + (\partial_y p)^2$ inside the segment is greater than $0.25 \cdot 10^6$, where ∂_x and ∂_y denote Sobel [3] filters.

The first guess for the position of the center of the cross (Figure 2.1(e)) is the pixel of minimum intensity inside the segment after a Gaussian-blur [4] with parameter $\sigma = 5$. This position is then updated (Figure 2.1(f)) by fitting a quadratic function (again in the form $u^T R u$) estimating this Gaussian-blurred intensity in a 7×7 window and selecting the minimum of the fitted function, process which is repeated for 5 iterations. This quadratic fit is weighted using a Gaussian function of the distance to the center of the window, with parameter $\sigma^2 = 5$.

If the overall algorithm finds 4 crosses in 3 consecutive frames, it stops and asks the user to approve or reject the calibration (Figure 2.2). The four crosses are classified as top-left, top-right, bottom-left and bottom-right (Figure 2.1(g)) by sorting their x coordinate to separate left-crosses from right-crosses, then the top- and bottom-crosses of each group are discriminated by comparing their y coordinate.

2.2.3 Pen Cap Tip Tracking

To track the pen cap tip, again, instead of using a general tracking method, we employ a custom one in order to achieve high precision, described as follows.

The first estimate for the pen cap tip position (Figure 2.3(a)) is computed by finding the pixel (x, y) that maximizes $2y + x$ and satisfies the following constraints: $B(x+i, y+i) > 40$, $B(x+i, y+i) > 1.6R(x+i, y+i)$ and $B(x+i, y+i) > 1.6G(x+i, y+i)$ for all $i \in \{-4, -3, -2, -1, 0\}$, and there must exist a pixel in the line segment between (x, y) and $(x, y+30)$ that is found in the convex hull of the four crosses. The reason for taking $i \in \{-4, -3, -2, -1, 0\}$ and not simply $i = 0$ is an attempt to avoid that random points on the paper pass this blue color filter, while the reason for considering any point in the line segment linking (x, y) and $(x, y+30)$ and not only the point (x, y) is that the shadow is expected to be at most 30 pixels below the pen cap tip (i.e. we only analyze the regions that make it possible that our algorithm finds a shadow residing inside the convex hull of the four crosses). Also, to save computation time, the pen cap tip is only searched for in one of every 3 lines. We call the coordinate pair of the estimated position $\tilde{z} = (x, y)^T$.

This estimate is refined (Figure 2.3(b)) to $\tilde{z}' = \tilde{z} + T\Delta$ for $T = \left[\begin{smallmatrix} 1 & 1/2 \\ -1/2 & 1 \end{smallmatrix}\right]$ and $\Delta \in \mathbb{R}^2$ computed as follows: First Δ_1 is chosen as the tilted column (i.e. after transformation T) that locally minimizes the sum of the intensities in this column, while Δ_2 is the row that maximizes a derivative filter within the column. Precisely speaking, we compute $\Delta_1 = \arg\min_{i \in \{-6,\dots,6\}} c(i-1) + 2c(i) + c(i+1)$ where $c(i) = \sum_{j=-2}^{10} p(\tilde{z} + T\left[\begin{smallmatrix} i \\ j \end{smallmatrix}\right])$, and $p(x, y)$ for non-integer x or y is computed using bilinear [5] interpolation; then $\Delta_2 = \arg\max_{j \in \{-2,-1,\dots,10\}} \partial_j p(\tilde{z} + T\left[\begin{smallmatrix} \Delta_1 \\ j \end{smallmatrix}\right])$, where ∂_j denotes a tilted Sobel filter:

$$\begin{aligned} \partial_j p(r) =& p(r + T\left[\begin{smallmatrix} -1 \\ 1 \end{smallmatrix}\right]) + 2p(r + T\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]) + p(r + T\left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]) \\ & - p(r + T\left[\begin{smallmatrix} -1 \\ -1 \end{smallmatrix}\right]) - 2p(r + T\left[\begin{smallmatrix} 0 \\ -1 \end{smallmatrix}\right]) - p(r + T\left[\begin{smallmatrix} 1 \\ -1 \end{smallmatrix}\right]) \end{aligned}$$

also using bilinear interpolation when necessary. $\tilde{z}'$ is rounded down in the end.

$\tilde{z}'$ is further refined (Figure 2.3(c)) to $\tilde{z}''$ by maximizing an objective function of the pixel position. We start from $\tilde{z}'$ and compute this objective function at the point and at its 8 surrounding pixels. The one with maximum value is selected and the process is repeated until convergence (maximum argument at the center). The objective function we chose is $f(r) = e^{(y+2x)/25} \cdot (L \star (\partial_y(3(R+G) - B)))$, where L is a 23×23 blur filter in the form $(sinc(x/12)sinc(y/12))^2$ [6] and ∂_y is the $\left[\begin{smallmatrix} -1 \\ 0 \\ 1 \end{smallmatrix}\right] \star$

[2] See Appendix B.6
[3] See Appendix B.4
[4] See Appendix B.5
[5] See Appendix B.9
[6] See Appendix B.7

cross-correlation [7] filter.

The choice for this objective function is quite empirical: this one worked the best compared to other attempts and it minimizes the "serif" effect (see Section 3.1). The idea behind this function is as follows. Typically, the y derivative of the intensity will be highest at the pen cap tip. However, when the pen touches its shadow, the point of maximum y derivative is highly dubious, due to the horizontal shape of the shadow. So, roughly speaking, out of these candidates, we would like to choose the one with highest x coordinate value. The use of an exponential function for maximizing the latter is justified by the behavior of $\log f$, when f is positive: notice that $\nabla \log f(r) = \begin{bmatrix} 2/25 \\ 1/25 \end{bmatrix} + \frac{\nabla \widetilde{p}}{\widetilde{p}}$, with $\widetilde{p} = L \star (\partial_y (3(R+G) - B))$, implying that critical points occur when $\widetilde{p}$ falls more than a certain percentage. The color ponderation was chosen to prioritize pen-paper rather than shadow-paper transitions and the squared sinc filter was chosen for its quality as low-pass filter.

The final estimate z for the pen cap tip position (Figure 2.3(d)) is computed by fitting the objective function described above to a quadratic function (again in the form $u^T R u$) using its measured value at $\widetilde{z}''$ and the 8 surrounding pixels, and then maximizing this fitted function. Although technically speaking this does not interpolate the objective function, the result is very similar to an interpolation since the function is largely smooth and can be properly approximated by a quadratic one.

2.2.4 Shadow Tracking

As the user cannot look at the paper while drawing, we must provide a hint of to where the pen is pointing, so that the user can know where the click is going to be performed before they click. In order to achieve this we must be able to predict where the pen will hit the paper.

Theoretically, if one can track the coordinates of the pen tip z and shadow tip s, and one knows the vanishing point d correspondent to the direction toward the which the user moves their hand (assuming a linear movement) and the position l of the light source projected on the table following this direction, the hitting position will be at $h = (s \times l) \times (z \times d)$, in homogeneous image coordinates. See Figure 2.4 for an illustration.

One possible technique to calibrate l and d would be to ask the user to double click on some points on the paper, then observe the trajectories of z and s and find the vanishing points where these trajectories cross, yielding d for z and l for s.

However, in order to avoid an extra calibration procedure and to simplify computation, we simply assume that $d = (0, 1, 0)^T$ and $l = (1, 0, 0)^T$, yielding

[7] See Appendix B.2

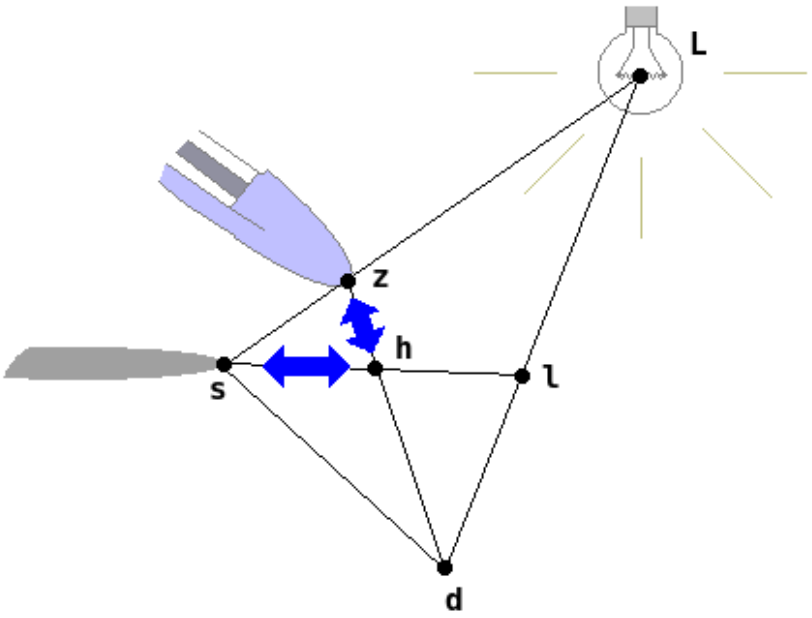


Figure 2.4: Let L be the position of the light in homogeneous image coordinates. Then L, z and s are collinear, and their projection onto the desk following direction d, respectively l, h and s must also be collinear. Therefore, when the user moves the pen down, as the hitting point h remains constant, z must move on the $\overline{zhd}$ line and s on the $\overline{shl}$ line, resulting that $h = (s \times l) \times (z \times d)$.

$h = (z_1, s_2, 1)^T$. This assumption does make some restrictions in webcam and lamp position: the lamp should not be too near to the paper and the webcam may not be too much downwards inclined (should have pitch and roll approximately zero). However, this assumption also means that we only need to calculate the y coordinate of the shadow, as described in the paragraphs below.

All shadow tracking is performed in a rectangular window containing the pixels $(x, y) \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] = [\lfloor z_1 \rfloor - 65, \lfloor z_1 \rfloor + 3] \times [\lfloor z_2 \rfloor - 10, \lfloor z_2 \rfloor + 29]$, as the shadow usually appears under the pen on the left side, i.e. there is no need to track the shadow at the right of the pen or above it.

The first step is to compute the paper intensity in the region. We compute the maximum intensity M of the line $y = \lfloor z_2 \rfloor + 15$ in this window and then the mean intensity μ of all pixels greater than $0.75M$ in this same line (i.e. the pixels considered paper in this line). Our threshold between paper and non-paper is then set to $w = 0.75\mu$. We chose this threshold for the next steps (and not one depending only on M) because M is too unstable and would make our method less precise.

For each line in the window, let $g(y)$ be the number of pixels in line y with intensity greater than w. Starting from the bottom of the window upwards we search the first value of y where $g(y)$ falls to less then a limit $\bar{g}$ set to 70% of the length of the window (69 pixels), and call this value $\tilde{s}_2$. However, as we need subpixel

precision, we would like to know where exactly between $\widetilde{s}_2$ and $\widetilde{s}_2+1$ this transition occurs.

To achieve this we interpolate the function $g : \mathbb{Z} \to \mathbb{Z}$ to $g : \mathbb{R} \to \mathbb{Z}$ as follows. $p(x, y)$ for non-integer y and integer x is obtained interpolating linearly [8] after gamma correction [9], i.e., $p(x, y)^\gamma = (1-(y-\lfloor y \rfloor))p(x, \lfloor y \rfloor)^\gamma + (y-\lfloor y \rfloor)p(x, \lfloor y+1 \rfloor)^\gamma$, for $\gamma = 2.2$. Then $g(y)$ for non-integer y is computed with respect to this interpolated line. To find then $y \in [\widetilde{s}_2, \widetilde{s}_2 + 1]$ where $g(y) = \bar{g}$, we compute first for every x such that $p(x, \widetilde{s}_2) > w$ but $p(x, \widetilde{s}_2 + 1) \leq w$ or vice-versa the value y_x where $p(x, y_x) = w$. These values are sorted in a list, and then we can easily find this transition point y and set s_2 to it, following the pseudocode below:

$\Omega = \{\}$
for $x = x_{\min}, ..., x_{\max}$ **do**
  $y_x \leftarrow \widetilde{s}_2 + \frac{w^\gamma - p(x,\widetilde{s}_2)^\gamma}{p(x,\widetilde{s}_2+1)^\gamma - p(x,\widetilde{s}_2)^\gamma}$
  if $p(x, \widetilde{s}_2) > w$ **and** $p(x, \widetilde{s}_2 + 1) \leq w$ **then**
    Add $(y_x, -1)$ to Ω
  else if $p(x, \widetilde{s}_2) \leq w$ **and** $p(x, \widetilde{s}_2 + 1) > w$ **then**
    Add $(y_x, +1)$ to Ω
  end if
end for
$g \leftarrow g(\widetilde{s}_2)$
for all $(y, r) \in \Omega$, sorted in increasing order of y **do**
  $g \leftarrow g + r$
  if $g \geq \bar{g}$ **then**
    $s_2 \leftarrow y$
    return s_2
  end if
end for

This meticulous interpolation procedure with gamma correction is very important to make shadow tracking accurate and minimize the "undulated diagonal" problem. This sort of problem occurs with more naive approaches to this interpolation step because the portion of the webcam image we use has a much lower resolution (particularly in the vertical axis) than the mouse range window: as the y coordinate (s_2) is biased, if one draws a diagonal line, they may see undesired undulations in its shape (Figure 2.5).

Other approaches such as plain linear interpolation of $g(y)$ are inappropriate because in fact $g(y)$ is usually not quite linear between $\widetilde{s}_2$ and $\widetilde{s}_2+1$ (See Figure 2.6). Usually the shadow appears almost tangent to the horizontal axis in the transition

[8] See Appendix B.8
[9] See Appendix B.1

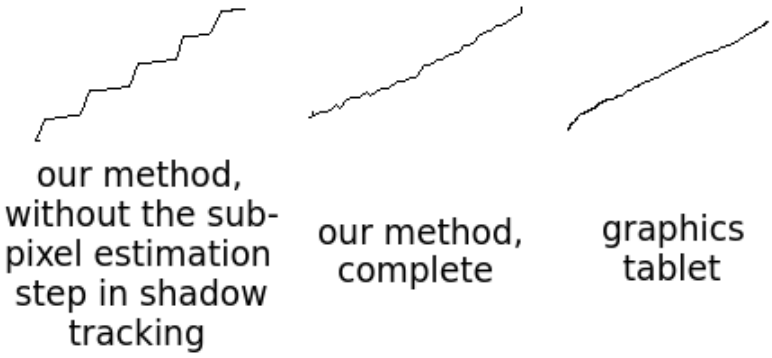


Figure 2.5: Diagonal lines as they appear using different interfaces. Drawings were done with $W_F \times H_F = 1280 \times 1024$, which is the same resolution used by the graphics tablet. (Drawn in Kolourpaint)

point, so that a large number of pixels crosses the threshold approximately at the same value of y, and the transition point ends up depending largely on the predominant intensity of the pixels of both lines.

Finally, if we happened to get $s_2 < z_2 - 3$, or if no transition point $\widetilde{s}_2$ was found, then we assume that no shadow was present in the window.

2.2.5 Mouse Motion

The position of the mouse cursor within the mouse range window is computed by applying a rectification (homography [10]) technique using the four crosses from the calibration process and mapping the point $h = (z_1, s_2, 1)^T$ to the window, yielding a point $m \in \mathbb{R}^2$. Mouse coordinates $\widetilde{m} \in \mathbb{Z}^2$ are updated from m using a hysteresis technique in order to increase stability:

$$\widetilde{m}_k^{t+1} = \begin{cases} \lfloor m_k^{t+1} + 0.5 \rfloor & , \text{ if } |m_k^{t+1} - \widetilde{m}_k^t| \geq 1 \\ \widetilde{m}_k^t & , \text{ otherwise} \end{cases}$$

where $k \in \{1, 2\}$ denotes the coordinate (x or y) and t and $t+1$ denotes the frame.

2.2.6 Mouse Click

We employ two conditions for mouse click. The most obvious condition is that the shadow and the pen must be sufficiently near to each other. For this we use $s_2 < z_2 + 7$. However, as this criterion may fail (i.e., the pen cap may be mistaken by a shadow in the shadow tracking step), we resort to an extra condition.

We apply a more lenient version of the color filter used in pen cap tip tracking — in this case $B > 20$, $B > 1.6R$ and $B > 1.6G$ — in a 30×6 window centered

[10] See Appendix A.2

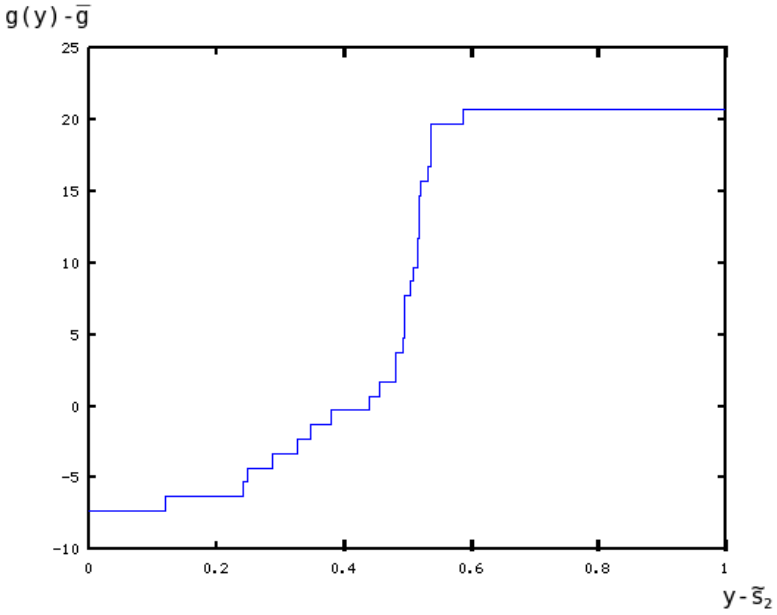


Figure 2.6: Subpixel estimation step in shadow tracking. This figure shows an example of how the $g(y)$ interpolated curve looks like between $\widetilde{s}_2$ and $\widetilde{s}_2 + 1$.

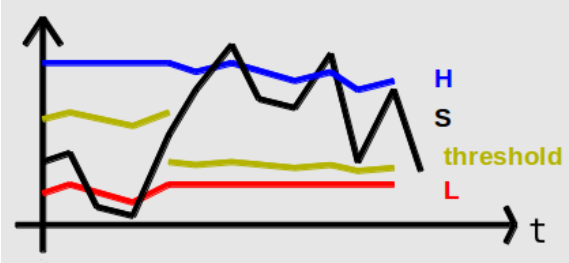


Figure 2.7: Adaptive threshold used in mouse click (illustration). L is updated when the pen is not touching, while H when it is, and the threshold that discriminates touch is a value between L and H following a hysteresis technique to avoid undesired mouse clicks or mouse button releases.

at $(\lfloor z_1 \rfloor - 0.5, \lfloor z_2 \rfloor - 0.5)$ to isolate pen cap pixels from paper and shadow pixels. We compute the mean μ and standard deviation σ of the non-pen pixels inside this window and use a quantity $S = \sigma/\mu$ to discriminate if the pen and shadow are touching one another. S is expected to be high (around some value H) when touching and low (around L) when not touching. However, appropriate values for H and L depend on conditions such as illumination quality, the position of the light, the way the user holds the pen, and even the portion of the paper currently being used. For this reason we use an adaptive threshold (See Figure 2.7), by learning expected values of H and L.

We start with $H_0 = 0.3$ and $L_0 = 0.2$. At frame t, the criterion for touching the paper (apart from the shadow position one) uses a hysteresis technique imposing that $S_t > 0.4L_{t-1} + 0.6H_{t-1}$ if the pen was not touching the paper at $t - 1$, and

$S_t > 0.9L_{t-1} + 0.1H_{t-1}$ otherwise. This helps avoid unwanted mouse clicks or mouse button releases.

L_t and H_t are updated as follows. If the pen is touching at frame t, we update $H_t = 0.8H_{t-1} + 0.2S_t$ and $L_t = L_{t-1}$; if not, we do the opposite. If the pen cap is currently unavailable (for instance if it is out of the trackable region), we do instead $H_t = 0.95H_{t-1} + 0.05H_0$ and $L_t = 0.95L_{t-1} + 0.05L_0$.

Chapter 3

Results

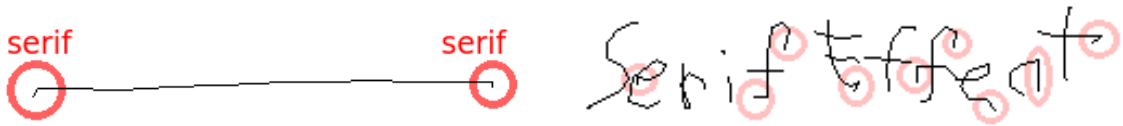
We first introduce in Section 3.1 the main problems we find by using a common pen to mimic a graphics tablet under our approach, and then we show in Section 3.2 how these problems are perceived in user tests. Finally, in Section 3.3 we present a quantitative measurement of the precision of our tracking algorithms.

3.1 Limitations

First of all, our system may misbehave (ex.: perform undesired clicks, release the mouse button during drag-and-drop, not accepting clicks on a determined region of the paper, suddenly warping the mouse cursor to another position during one frame, among other problems) if the restrictions on illumination, webcam, paper, etc. are not met (as described in Appendix C). Also, we admit that we did not employ good techniques to judge "if the detected pen was actually a pen", or "if the detected shadow was actually a shadow", which ends up making these restrictions even stricter. For instance, sunlight may cause interference, non-incandescent desk lamps do not work well with our system and shadows from strange objects may also be problematic. If the pen is not correctly illuminated precision may be lost, and if the paper is not homogeneously illuminated the adaptive threshold for clicking may also misbehave. Users may have difficulties in placing the lamp in an appropriate position, and the way the user holds the pen also influences the quality of the mouse control.

Apart from this sort of limitation, there is one type of artifact that is inherent from this method of simulating mouse input, which we call the "serif" effect.

The "serif" effect (Figure 3.1) is characterized by a rapid change of the mouse cursor position when the pen touches the paper. It is particularly undesirable because when it happens the user will not have clicked on the position he wanted to. One cause of this effect is that pen and shadow merge their colors when they touch each other, affecting both (pen and shadow) tracking algorithms. We chose



(a) Mild serif effect on a single stroke. (Drawn in Kolourpaint)

(b) Severe serif effect caused by undulated paper due to sweating. Red circles are drawn calling attention to the serifs. (Drawn in Kolourpaint)

Figure 3.1: "Serif" effect examples.

the algorithms attempting to minimize this cause, however, there is a second cause, over the which we have no control, which is undulated paper. These undulations may be very subtle, but they will make the pen and the shadow move downwards when the user clicks, thus creating this "serif"-like artifact. This effect becomes more evident if the user sweats by their hands, as the paper becomes more undulated quickly. It could be reduced if the webcam was placed on a higher position, but as our method was not designed for this scenario, other parts of the method will show deficiencies. Another alternative is to remove the paper after calibration and use directly the table (if it is white). A third cause to the serif effect may reside in the adaptive threshold algorithm we employ for mouse click conditions, however, this is an improbable cause because its hysteresis scheme would make the effect more noticeable in the end of the stroke rather than in the beginning, which is not true (as one can see in Figure 3.1).

3.2 User Tests

We made a survey (Appendix D) with voluntary testers asking them to set the system up in their homes and try to use it, evaluating the ease to set up and the quality of the mouse control.

Unexpectedly, many users could not participate as they did not own a movable webcam (but only the one that comes on the top of a notebook monitor, for instance). Some tried connecting a smartphone to the computer and used the phone camera as webcam, and some also tried connecting a second monitor to their notebooks and inclining the first monitor down looking at the paper, but our method was not supposed to be used this way. Also a great number of users could not prepare proper lighting conditions because they did not own an incandescent desk lamp; and some even had no blue-capped pen, which means that our system is not as "buildable everywhere" as we expected. Due to these problems, we resorted to asking some users to try to set up and test our system in a laboratory ambient satisfying all constraints.

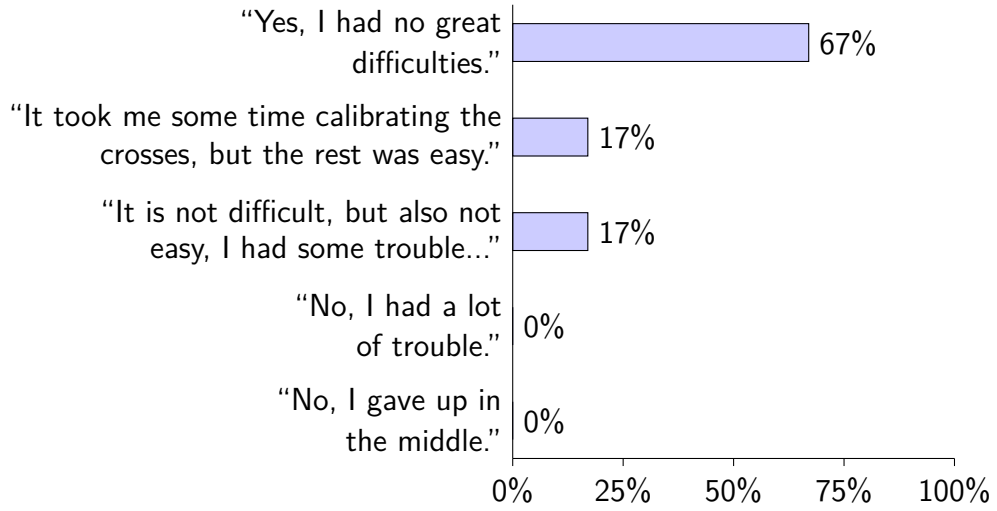


Figure 3.2: Ease of setup ("Did you find the system easy to set up?")

Out of the users who managed to satisfy the basic conditions for testing (2 persons), and the ones who tried the software in the laboratory (28 persons), most found the system easy to set up (Figure 3.2), and estimated having required, on average, 4min15s for the task (the shortest time reported was 15s, while the longest was 15min) [1]. Precision was evaluated as modest (Figure 3.4) but users in general would accept using our system replacing the graphics tablet (Figure 3.5 [2]). Although most users were inexperienced with graphics tablets (Figure 3.3 [3]), there does not seem to be a correlation between experience with the graphics tablet and the acceptance to our system.

The most often reported defects were undesired clicks and the "serif" effect, reported by, respectively, 47% and 40% of the users. Those who tried drawings were particularly disturbed by the "serif" effect, and some were also uncomfortable with the restrictions on the way of holding the pen. Another frequent complaint is the maximum height (30 pixels, or about 1cm) that the pen may take in order to move the mouse cursor by hovering.

We also asked some users to make comparisons between the graphics tablet and our system, and the result can be seen in Figures 3.6 and 3.7.

Some users that were not used to graphics tablets reported having found using pen and paper more comfortable (ergonomically speaking) than the graphics tablet, because the paper has a millimetric thickness, being at the same level of the surface,

[1] Some users interpreted "time to set up" to be the time spent in the calibration step, others as the time spent until one can make the system work correctly.

[2] Some users communicated having interpreted the answer "No, it does not fit my uses" as "No, I have no use for it.", having chosen this one even though they were satisfied with the system.

[3] Some users misunderstood "graphics tablet" with "tablet computer" in this question. Others communicated having considered an "experience with the graphics tablet" the act of using them or similar devices to sign in government establishments.

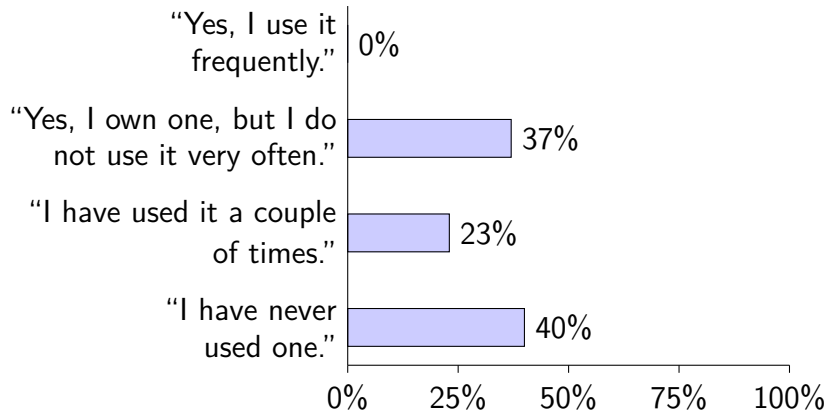


Figure 3.3: Experience of users with graphics tablets ("Are you used to the graphics tablet?")

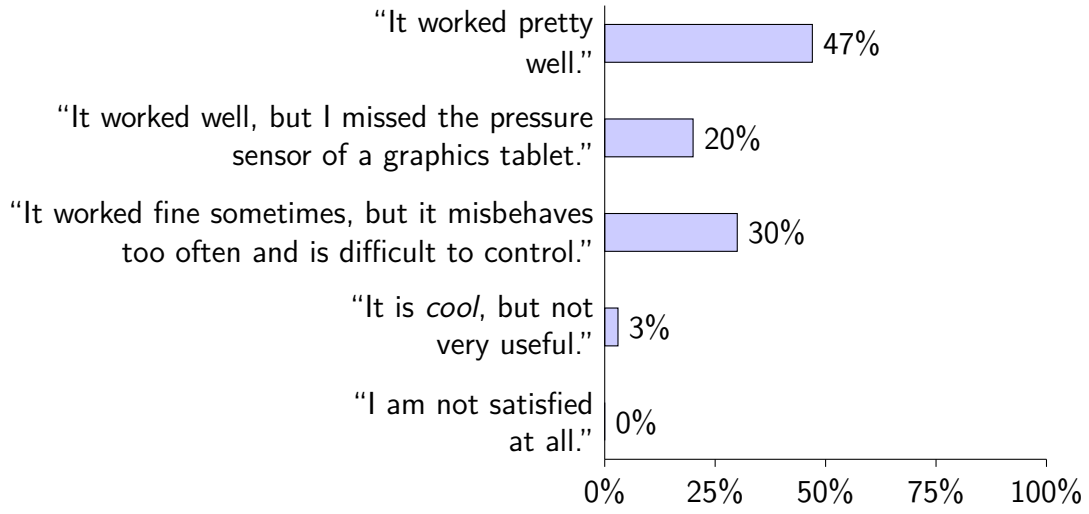


Figure 3.4: Quality evaluation ("What did you think about the quality of the mouse control?")

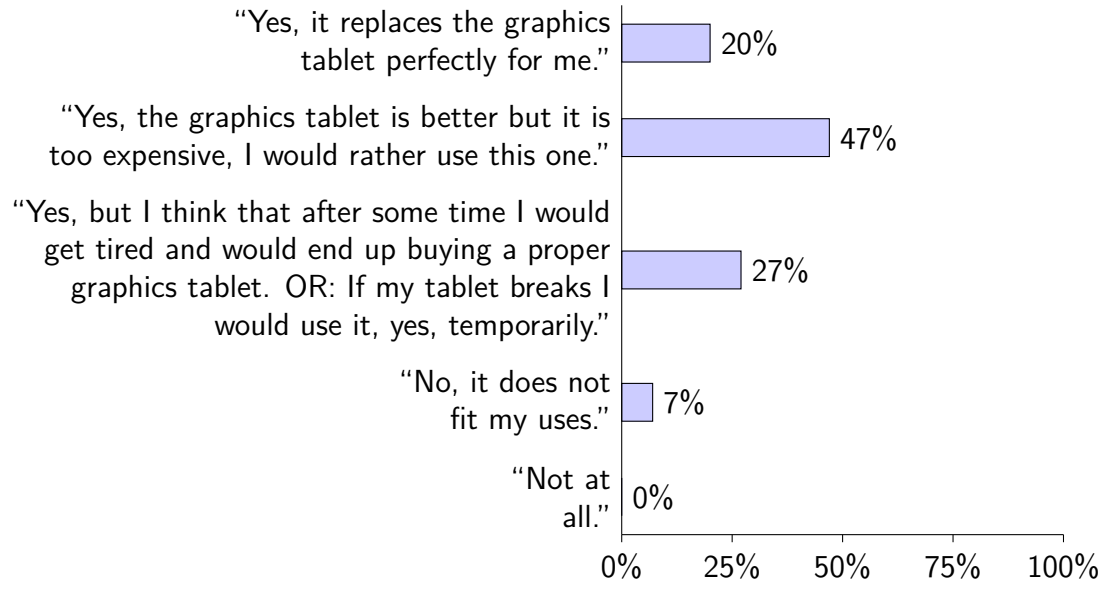


Figure 3.5: Overall evaluation ("Would you use our system?")

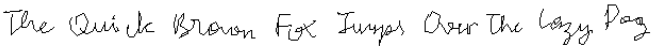
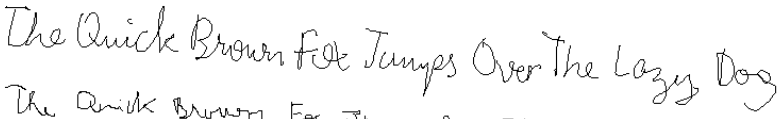
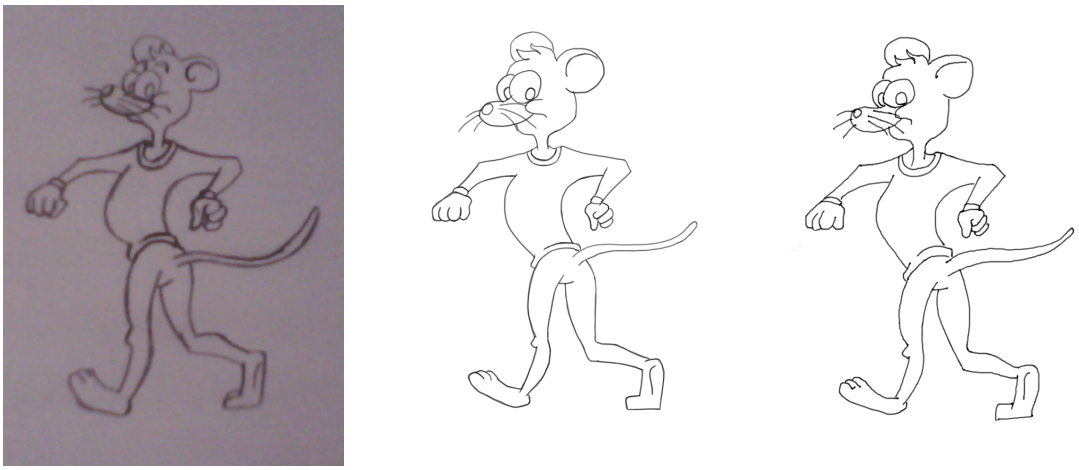
Interface	Drawing Time	Output
Webcam-PaperPen	23.82s	
Graphics Tablet	22.72s	
Mouse	62.21s	

Figure 3.6: Comparison of our system with graphics tablet and optical mouse. We used a range window of 640×480 in our software and crosses forming a 15cm×12cm rectangle (approximately), while the tablet used a resolution of 1280×1024 and has an input area sized 15cm×9.2cm. The difference in time between the graphics tablet and our system is mainly because this sentence does not fit in the range window of our system (640×480), requiring it to be moved to the side at least once while writing the sentence. All the three cases were drawn in Kolourpaint, by the same user.



(a) Drawn by hand (photograph)

(b) Drawn using a graphics tablet (in MyPaint)

(c) Drawn using our system (in MyPaint)

Figure 3.7: Comparison for drawing applications. All the three drawings were drawn by the same user.

although this may be due to the graphics tablet model we used in the tests. Also, some of the problems found with our method (such as imprecise mouse cursor control when hovering or changes in the cursor position right before clicking) are also found with graphics tablets, albeit not as noticeable or not as disturbing when using the latter.

3.3 Quantitative Precision Measurement

We have employed a quantitative experiment to measure the precision of the system.

One user was asked to hold the pen still for some seconds, hovering or touching the paper, on several positions and holding poses. During this time, we measured z_1^t, z_2^t and s_2^t, where z and s are respectively the pen cap tip and shadow tip positions in webcam image coordinates, and t indicates the time frame. After measurement, we analyzed the values of $f^t - f^{t-1}$, for a variable f among z_1, z_2 and s_2.

These values are not a direct measurement of the precision of our algorithm because they are affected by intentional movement of the pen, which happened in this experiment when the user changed the position of the pen or the holding pose.

Attempting to eliminate the measurements where the user is intentionally moving the pen, we opted to discard all the values where $|f^t - f^{t-1}| \geq 0.5$. Out of the 2146 frames measured, the discarded values correspond to:

- 12.0224% of the measurements of z_1;
- 9.8322% of the measurements of z_2;

- 2.0969% of the measurements of s_2.

Using the remaining values, we estimated an error in the form $\sigma_f = \sqrt{\frac{1}{2}\mathrm{E}[(f^t - f^{t-1})^2]}$ [4] yielding:

- $\sigma_{z_1} = 0.116029$ pixels;
- $\sigma_{z_2} = 0.102873$ pixels;
- $\sigma_{s_2} = 0.094950$ pixels.

These values may vary, however, depending on the webcam, the lighting conditions, the person who is holding the pen, the distance to the webcam, and other factors. They prove, however, that our algorithms reach subpixel precision on image coordinates if the ambient is properly configured. Nonetheless, after these coordinates are mapped to mouse range window coordinates, this error measure becomes much larger, specially for the y coordinate (i.e., the error measure in m_2, the mouse pointer y coordinate before truncation, is much larger than the one in s_2, because the webcam is usually placed on the table looking at the paper from a very inclined angle in relation to the normal of the table).

[4] The reason for the $\frac{1}{2}$ factor is that $\mathrm{E}[(X_1 - X_2)^2] = 2\mathrm{Var}(X)$ for two identical and independently distributed random variables X_1 and X_2.

Chapter 4

Conclusions and Future Work

We have presented a low-cost, practical and easy-to-set-up system to generate mouse input using paper, pen and webcam, aimed at handwriting and drawing applications, for situations where the computer mouse would not be precise, fast or comfortable enough and a graphics tablet would be unaffordable. If the system is properly configured, it is precise enough for handwriting and simple drawings, successfully complementing the mouse. However, user tests proved our system to be still unusable for more artistic applications, particularly due to the "serif" effect.

Apart from correcting the limitations mentioned in Section 3.1, we would also like to keep precision high in different configurations of illumination, paper, pen, webcam position, etc.. Particularly, flexibility in the range of pens and light sources that can be used is desired. We would also like to achieve pixel precision (after rectification) in a higher resolution, and to discover a way to eliminate the "serif" effect completely.

An obvious extension of this work on the which we are interested is obtaining the 3D position of the pen tip, which can be easily done by using the shadow as reference and making a few assumptions; and use it in applications such as 3D modeling.

Appendices

Appendix A

Computer Vision

A.1 Homogeneous Image Coordinates

Homogeneous image coordinates is a mathematical tool to deal with projective geometry, i.e. 3D objects after perspective projection on a 2D space. A point $r = (x, y)$ can be represented using homogeneous coordinates as a column-vector $u = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, or any vector $u' = \alpha u$, for some $\alpha \in \mathbb{R} \setminus \{0\}$. Therefore r can be recovered from u' using the perspective division: $r = \pi(u') = ({u'}_1/{u'}_3, {u'}_2/{u'}_3)$. Note also that the equation $\exists \alpha \neq 0 : u' = \alpha u$ is equivalent to $u' \times u = 0$ if $u, u' \neq 0$.

Analogously, a line in homogeneous image coordinates is represented by a row-vector $l = \begin{bmatrix} a & b & c \end{bmatrix}$ so that points u lying on this line satisfy $lu = 0$. Naturally, multiples αl of l are equivalent. Two different points u_1 and u_2 can be linked with the line $(u_1 \times u_2)^T$ and two different lines l_1 and l_2 cross each other on the point $l_1^T \times l_2^T$.

Another advantage of using homogeneous image coordinates is the ease to represent points in infinity by setting the third coordinate to 0. For instance, two parallel lines $l_1 = \begin{bmatrix} a & b & 1 \end{bmatrix}$ and $l_2 = \begin{bmatrix} a & b & 2 \end{bmatrix}$ cross each other at $u = \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix}$, while the line that links this crossing point u to any point $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ is $\begin{bmatrix} a & b & c \end{bmatrix}$ for some $c \in \mathbb{R}$.

A.2 Rectification

Consider we have a rectangle in 3D space with corners $q_1 = (0, 0, 0)$, $q_2 = (X, 0, 0)$, $q_3 = (0, Y, 0)$ and $q_4 = (X, Y, 0)$. If we take a picture from this rectangle after some rotation Q and translation t (i.e. the translation and rotation of the camera in relation to this rectangle), points $p = (x, y, 0)$ inside this rectangle will be projected to the image in homogeneous coordinates $u = K(Qp + t)$, where K is the intrinsic matrix of the camera, disregarding lens distortions. We can rewrite this equation as $u = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, where $H = \begin{bmatrix} KQ \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} & Kt \end{bmatrix}$, since we know that $p_3 = 0$. Therefore

we can easily map points u in the image to points p in the 3D world (inside the rectangle) once we know H. The transformation this matrix does is often called "homography" [16]. Note that, in homogeneous image coordinates, any multiple αH of H has the same effect, i.e., they are equivalent.

Suppose now that we have the points $q'_1, ..., q'_4$ where $q_1, ..., q_4$ are projected. H can be recovered by solving the system $(H(q_i + e_3)) \times q'_i = 0$ for $i = 1, ..., 4$, where $e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. This is a system of 9 variables (the entries of H) and 12 equations; however, as cross products have rank 2, the complete system has rank 8. In practice, we disregard the third component of the cross product, and end up with only 8 equations. The system is solved by applying a (full) singular value decomposition and extracting the right singular vector assigned to the singular value zero.

Appendix B

Image Processing

B.1 Gamma Correction

Computer images are usually quantized using gamma correction to take advantage of the sensitivity of the human eye. In a simplified model, the webcam converts the energy B_0, R_0 and/or G_0 measured in a pixel to $R = R_0^{1/\gamma}$, $G = G_0^{1/\gamma}$, $B = B_0^{1/\gamma}$, while computer monitors and printers revert this to $R' = R^\gamma$ and so forth. Most systems use $\gamma \approx 2.2$ [16].

B.2 Cross-Correlation

Cross-correlation is the operation defined as:

$$(f \star g)(x) = \int_{-\infty}^{\infty} f^*(t) g(x+t) dt$$

or in $\mathbb{R}^2$:

$$(f \star g)(x,y) = \int_{\mathbb{R}^2} f^*(t,\tau) g(x+t, y+\tau) dt d\tau$$

or in $\mathbb{Z}^2$ (images):

$$(f \star g)(x,y) = \sum_{(t,\tau)\in\mathbb{Z}^2} f^*(t,\tau) g(x+t, y+\tau)$$

where f^* denotes the complex conjugate of f.

We conveniently adopt the operator order $f \star g \star h = f \star (g \star h) = (f * g) \star h$, as filters are placed on the left side and images on the right side. "$*$" denotes convolution, as in $(f * g)(x) = (g * f)(x) = \int_{\mathbb{R}} f(t) g(x-t) dt$.

In image processing, cross-correlation may be graphically represented using a matrix as in:

$$(F\star) = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \star$$

This is equivalent to letting F be a function $F : \mathbb{Z}^2 \to \mathbb{R}$ such that

$$F^{\text{function}}(x, y) = \begin{cases} F^{\text{matrix}}_{y+2,x+2} & , \quad \text{if } (x, y) \in \{-1, 0, 1\}^2 \\ 0 & , \quad \text{otherwise} \end{cases}$$

i.e., the center of the matrix is by default at the origin of the equivalent function.

B.3 Binomial Filter

A binomial filter of length $(2k + 1)$ in one dimension is the cross-correlation filter $B_{2k+1}\star$ where:

$$B_{2k+1}(i) = \begin{cases} \binom{i+k}{2k} & , \quad i \in \{-k, ..., k\} \\ 0 & , \quad \text{otherwise} \end{cases}$$

B.4 Sobel Filter

A Sobel filter is a 3×3 cross-correlation kernel consisting of a derivative operator smoothed in the perpendicular direction with a binomial filter. The Sobel filter for the y derivative (denoted here ∂_y) is graphically represented as:

$$\partial_y = \left(\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \right) \star = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \star$$

B.5 Gaussian Blur

Gaussian-blur on images is performed using a $(2k+1)\times(2k+1)$ cross-correlation filter $G_{2k+1,\sigma}\star$ whose entries are the values of the Gaussian probability density function, normalized to sum 1, i.e.:

$$G_{2k+1,\sigma}(x, y) = \begin{cases} \frac{\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)}{\sum_{(i,j)\in\{-k,...,k\}^2} \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)} & , \quad (x, y) \in \{-k, ..., k\}^2 \\ 0 & , \quad \text{otherwise} \end{cases}$$

B.6 Dilation

Dilation on binary images $p : \mathbb{Z}^2 \to \{0, 1\}$ is a cross-correlation filter where multiplication is replaced by the AND logical operator and addition is replaced by the OR logical operator. The $(2k + 1) \times (2k + 1)$ square dilation filter $D_{2k+1}\star$ results in the image:

$$(D_{2k+1} \star p)(x, y) = \begin{cases} 1 & , \quad \text{if } \exists (i, j) \in \{-k, ..., k\}^2 : \\ & \quad p(x + i, y + j) = 1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

B.7 Sinc Function

We use the following definition for the sinc function:

$$sinc(x) = \frac{\sin(\pi x)}{\pi x}$$

B.8 Linear Interpolation

A function $f : \mathbb{Z} \to \mathbb{R}$ can be interpolated to $\widetilde{f} : \mathbb{R} \to \mathbb{R}$ using:

$$\widetilde{f}(x) = (1 - (x - \lfloor x \rfloor))f(\lfloor x \rfloor) + (x - \lfloor x \rfloor)f(\lfloor x + 1 \rfloor)$$

B.9 Bilinear Interpolation

Bilinear interpolation in images equivales to applying linear interpolation to each axis, i.e.:

$$\begin{aligned} \widetilde{f}(x, y) = &(1 - (x - \lfloor x \rfloor))(1 - (y - \lfloor y \rfloor))f(\lfloor x \rfloor, \lfloor y \rfloor) + \\ &(x - \lfloor x \rfloor)(1 - (y - \lfloor y \rfloor))f(\lfloor x + 1 \rfloor, \lfloor y \rfloor) + \\ &(1 - (x - \lfloor x \rfloor))(y - \lfloor y \rfloor)f(\lfloor x \rfloor, \lfloor y + 1 \rfloor) + \\ &(x - \lfloor x \rfloor)(y - \lfloor y \rfloor)f(\lfloor x + 1 \rfloor, \lfloor y + 1 \rfloor) \end{aligned}$$

Appendix C

Setup Instructions Sent to Testers (Translated to English)

You will need:

- 1 BIC blue-capped pen (the cap is indispensable!)
- 1 movable webcam (that one that comes inside the monitor is inappropriate)
- 1 A4-sized sheet of white paper
- 1 desk lamp of incandescent light

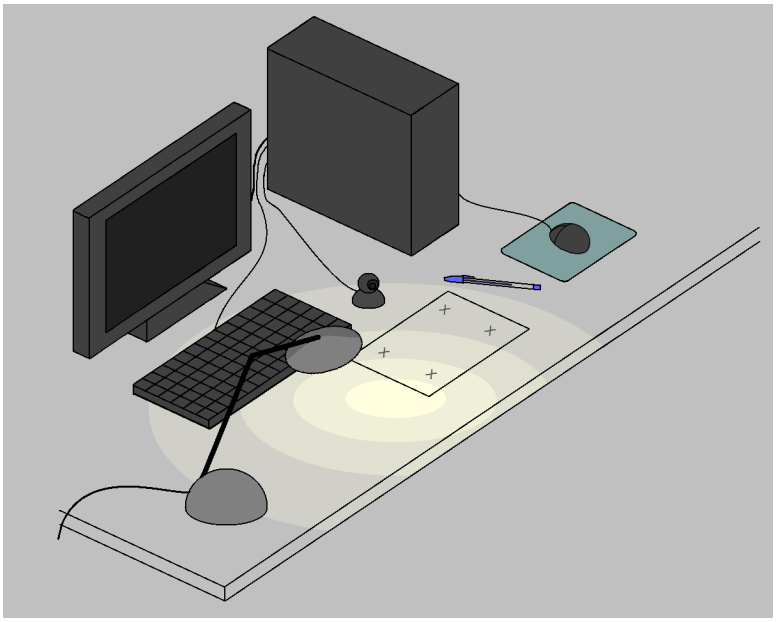


Figure C.1: General illustration of the system setup.

C.1 General Instructions

C.1.1 First of all:

Configure the ambient as in Figure C.1.

- Place the paper directly over the table, and draw four crosses on it, as in the figure (preferentially using pencil or pen of non-blue ink). Do not draw the crosses too close to the border. The paper must be as even as possible.
- Place the desk lamp at your left (if you are right-handed. Otherwise put it at your right and turn the left-handed mode of the software on). Make sure that the desk lamp is capable of generating a strong shadow. Hint: The stronger the light of the desk lamp, the higher is the frame rate of the webcam.
- Place the webcam between the paper and the monitor, on the table. DO NOT put it over a support, unless absolutely necessary. The webcam must be looking approximately to the horizon, do not use it too much inclined. (See Figure C.2) Make sure that the webcam image is good enough (is properly focused, has distinct colors, etc.).

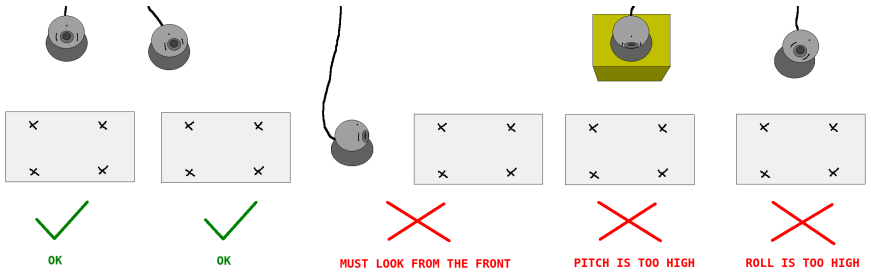


Figure C.2: Restrictions to camera rotation. It can have yaw as long as it continues seeing the paper from the front, it can have only a little bit of pitch and it must have no roll.

- Make sure that the room is properly illuminated (with ceiling light). Sunlight may disturb our software, it is recommended to shut the curtains.

C.1.2 Calibration Step

- In this step we recommend turning the desk lamp off (use only the ceiling light).
- The software will try to find the crosses. You will see the crosses highlighted in blue and non-identified portions in red. You can move the paper, the webcam

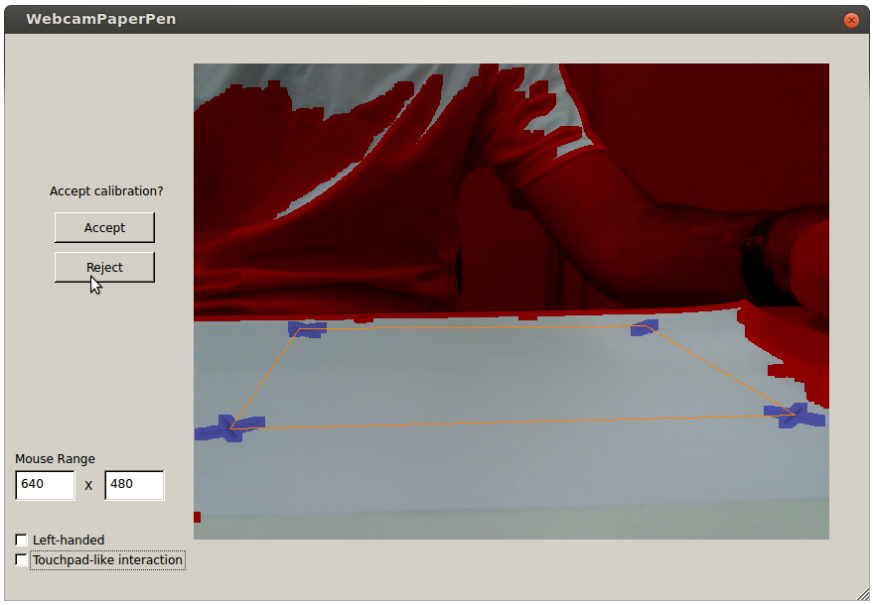


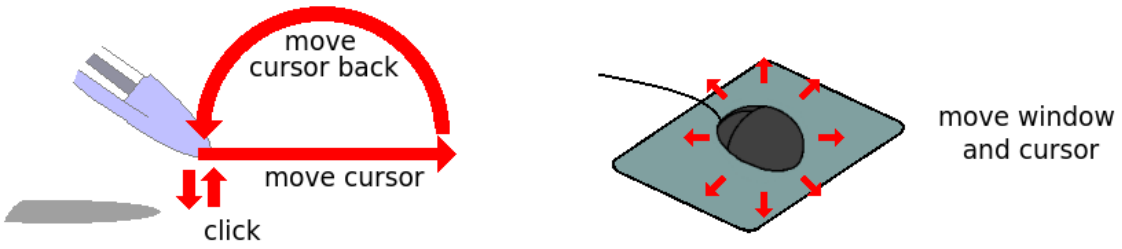
Figure C.3: Calibration confirmation.

and the desk lamp during this step to make sure that everything is correctly placed and that the software can detect the crosses.

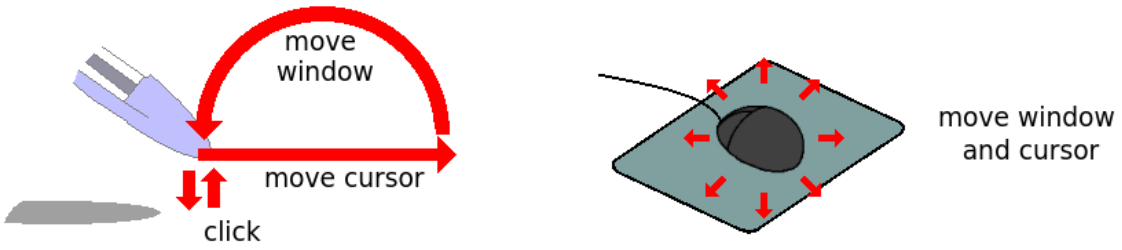
- All the crosses must be entirely visible to the webcam; if a cross is not being identified, it is possible that it is exceedingly far from or near to the webcam.
- If your webcam has autofocus, try moving something over the paper until the webcam focuses and then remove it.
- In the end you will be asked to confirm if the calibration is correct, as in Figure C.3.

C.1.3 Drawing Step

- Turn the desk lamp on.
- Use the pen with the cap shut, never releasing ink on the paper. The mouse click will be simulated when the pen cap touches its shadow.
- Avoid writing looking at the paper. Look always at the screen and use the mouse cursor as spatial reference. Do not write out of the area delimited by the 4 crosses.
- Mouse control is limited by a mouse range window. You can use one of our two interaction modes in order to reach the whole screen (Figure C.4) or edit the “Mouse Range” field for a larger resolution.



(a) Normal mode



(b) "Touchpad-like" mode.

Figure C.4: Interaction modes of our software. The illustration shows how the mouse cursor and the mouse range window are moved by the mouse and by the pen.

C.2 Frequent Problems

C.2.1 Calibration Step

- **"There is a big red blot on the middle of the paper"**
  - Cause: It can be the shadow of your own arm or of part of the desk lamp. Try to remove this shadow.
- **"The paper is not found (the segmentation between red and white does not seem to take the paper into account)"**
  - Cause: Some part of the image is whiter than the paper: it can be the wall, a table, white clothing, etc.. Try isolating these objects.
- **"The back crosses are not appearing"**
  - Try turning the lamp off.
  - Try approximating the paper a little bit to the camera.
- **"The back crosses are not being recognized"**
  - Try approximating the paper a little to the camera.
  - You should not draw the crosses too near to the border of the paper.

- **"The front crosses are not being recognized"**
  - Try placing the paper a little bit farther from the camera.
  - Remember that the crosses must lie entirely inside the webcam image.
  - Make sure that the webcam is correctly focused. If it has autofocus, try moving something over the paper in order to make it focus.

C.2.2 Drawing Step

- **"The pen is not detected"**
  - Possible cause: The lamp is turned off.
  - Possible cause: The lamp is not of incandescent light.
- **"The pen does not click"**
  - Possible cause: The shadow is not distinct enough. Place the lamp closer to the paper, close the curtains.
  - Possible cause: The paper is not homogeneously illuminated. Try scribbling (still with the pen cap shut) somewhere else on the paper and try again.
  - Possible cause: You are not holding the pen properly as in Figure C.5.
  - Possible cause: You are using the left-handed mode but you are right-handed, or vice-versa.
  - Possible cause: The pen is not a common blue BIC.

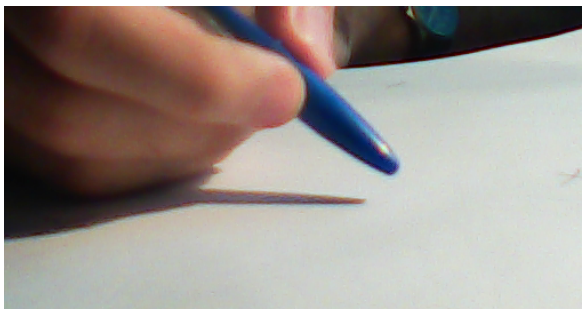


Figure C.5: Correct way of holding the pen (seen from the camera). Hold the pen with the fingers not too close to the tip, inclined in relation to the table (never completely in a straight vertical position), pointing approximately to the front and to the left (if right-handed).

- **"The click is released while I am drawing"**

- Possible cause: You are drawing too close to a cross.
- Possible cause: The paper has been folded.
- Check also the items "The pen trembles too much" and "The pen does not click".

- **"The pen trembles too much"**
  - Possible cause: It is possible that the illumination is not adequate. The pen must neither appear too bright nor too dark to the webcam. Try repositioning the lamp, turning the ceiling light on, closing the curtains. The desk lamp must be of incandescent light.
  - Hint: The portion of the paper that is closest to the webcam is the one that has the best tracking quality, you may prefer drawing on this portion.

- **"In the instant when I make the pen touch the paper, the mouse cursor moves"**
  - Possible cause: The paper is not perfectly even (it may occur if you sweat by your hands). You may remove that sheet of paper and put another one (recalibrating is not necessary) or write on some other white, even and hard surface. Another alternative is turning the "touchpad-like" mode on and keeping the hand still on the same portion of the paper, moving only the fingers.
  - Possible cause: The contact surface is not hard
  - Possible cause: The webcam is too much downwards inclined or rolled (See Figure C.2).
  - Possible cause: The lamp is not placed at the left side of the paper.

- **"In the 'touchpad-like' mode, as I raise the pen high from the paper, the mouse cursor jumps to somewhere else."**
  - There is probably some interference between the shadow of your hand and the one of the pen. Try placing the desk lamp farther (more to the side).
  - Another possible cause are unexpected shadows from other light sources. In this case you can try a stronger lamp (or place it closer to the paper).
  - Possible cause: Your pen is not a BIC pen (its cap is too thick).

- **"The mouse cursor is not moving the way I want"**
  - Possible cause: You are drawing too close to the border of the table

  - Possible cause: There are unexpected blue objects close to the paper. Note: Blue clothes (dressed by the user) cause usually no interference, unless the paper had been positioned too close to the border of the table.
  - Possible cause: There are unexpected shadows on the paper.
  - Possible cause: Your hand does a large shadow on the paper (say, it shadows most of the paper)

- **"Sometimes the mouse cursor does not move."**
  - Aren't you drawing outside of the area delimited by the four crosses?

C.2.3 Other Problems

- **"The program stopped working!"**
  - Possible cause: You disconnected the webcam or the computer mouse while the program was running.

Appendix D

Survey Sent to Testers (Translated to English)

Did you find the system easy to set up?

- Yes, I had no great difficulties.
- It took me some time calibrating the crosses, but the rest was easy.
- It is not difficult, but also not easy, I had some trouble...
- No, I had a lot of trouble.
- No, I gave up in the middle.

Did you read the instructions?

- Yes, I read before setting the system up.
- I just skimmed before setting up.
- I did not read before setting up, just consulted when some problem occurred.
- No, I did not.

How much time did it take you, approximately, to set the system up? (If you read the instructions before, then disregard the time you spent reading)

Are you used to the graphics tablet?

- Yes, I use it frequently.
- Yes, I own one, but I do not use it very often.

- ○ I have used it a couple of times.
- ○ I have never used one.

What did you think about the quality of the mouse control?

- ○ It worked pretty well.
- ○ It worked well, but I missed the pressure sensor of a graphics tablet.
- ○ It worked fine sometimes, but it misbehaves too often and is difficult to control.
- ○ It is *cool*, but not very useful.
- ○ I am not satisfied at all.

Check if you noticed and was bothered by any of the problems below.

- ☐ "Serif" effect (a mark when you touch or release the pen from the paper)
- ☐ Change in the mouse cursor position immediately before clicking
- ☐ Undesired click
- ☐ Mouse click does not work
- ☐ When I drag and drop the click is unwillingly released
- ☐ Undulated diagonal stroke
- ☐ Mouse cursor does not move anymore

Would you use our system?

- ○ Yes, it replaces the graphics tablet perfectly for me.
- ○ Yes, the graphics tablet is better but it is too expensive, I would rather use this one.
- ○ Yes, but I think that after some time I would get tired and would end up buying a proper graphics tablet. OR: If my tablet breaks I would use it, yes, temporarily.
- ○ No, it does not fit my uses.
- ○ Not at all.

Other Comments:

Appendix E

Setup Instructions Sent to Testers (Original, in Portuguese)

Você precisará de:

- 1 caneta BIC azul, com tampa (a tampa é indispensável!)
- 1 webcam móvel (não serve a embutida no monitor)
- 1 folha de papel branco tamanho A4
- 1 luminária de luz incandescente

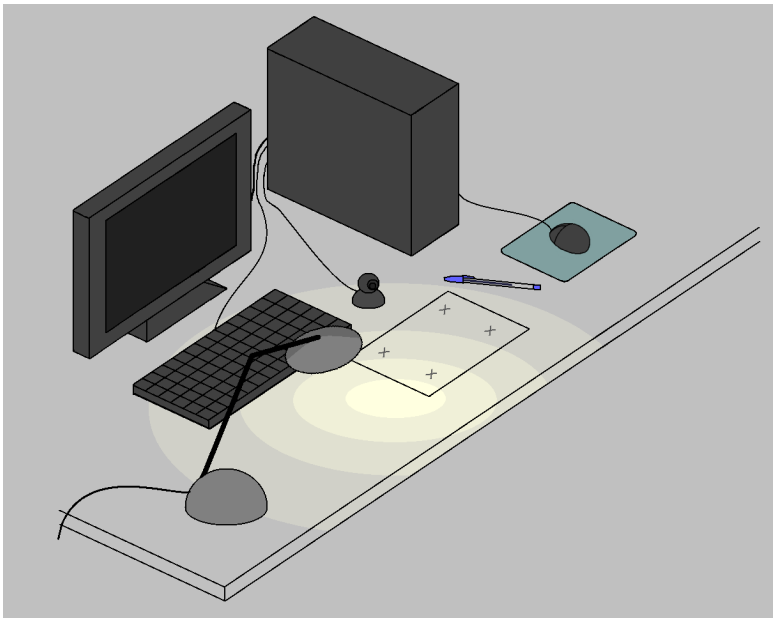


Figura E.1: Ilustração geral da configuração do sistema.

E.1 Instruções Gerais

E.1.1 Primeiro de tudo:

Configure o ambiente como na Figura E.1.

- Coloque o papel diretamente sobre a mesa, e desenhe quatro cruzes sobre ele, como na figura (preferencialmente a lápis ou com tinta de qualquer cor que não seja azul). Não desenhe as cruzes muito próximas à borda. O papel deve estar bem liso.

- Coloque a luminária à esquerda (caso destro. Se for canhoto coloque à direita e ligue o modo canhoto do software). Certifique-se de que a luminária é capaz de gerar uma sombra bem nítida. Dica: Quanto mais intensa a luz da luminária, melhor é o frame rate da webcam.

- Coloque a webcam entre o papel e o monitor, sobre a mesa. NÃO coloque um suporte levantando a webcam, a menos que absolutamente necessário. A webcam deve estar olhando mais ou menos para o horizonte, não a utilize muito inclinada. (Veja Figura E.2) Certifique-se de que a imagem da webcam é razoável (está focada, é colorida, etc.).

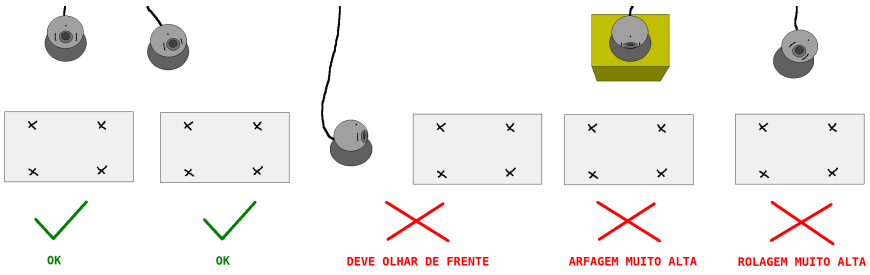


Figura E.2: Restrições quanto à rotação da câmera. Pode estar guinada desde que continue vendo o papel de frente, pode estar apenas um pouco arfada e não pode estar rolada.

- Certifique-se de que a sala está bem iluminada (com lâmpada de teto). Luz do sol pode ser prejudicial, recomenda-se fechar as cortinas.

E.1.2 Etapa de Calibração

- Nesta etapa recomenda-se apagar a luminária (mantendo apenas a luz de teto).

- O software tentará identificar as cruzes. Você verá as cruzes azuladas e pontos não identificados avermelhados. Você pode mexer no papel, na webcam e na

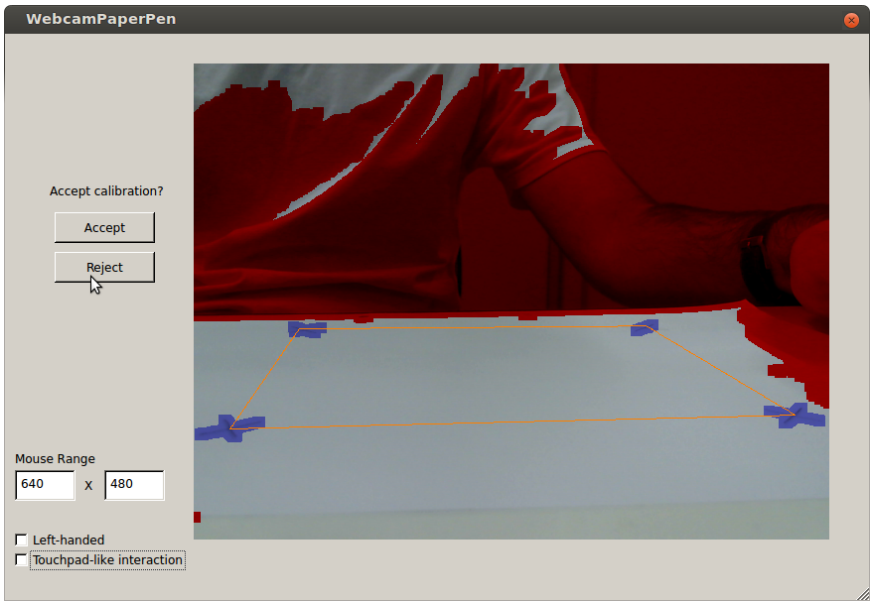


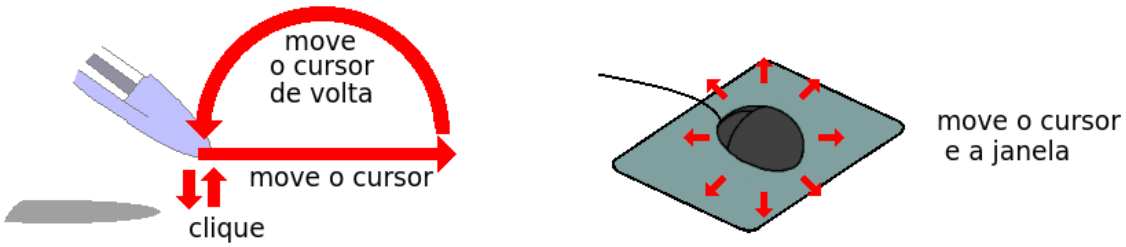
Figura E.3: Confirmação de calibração.

luminária durante esta etapa para certificar-se de que está tudo posicionado corretamente e que o software consiga detectar as cruzes.

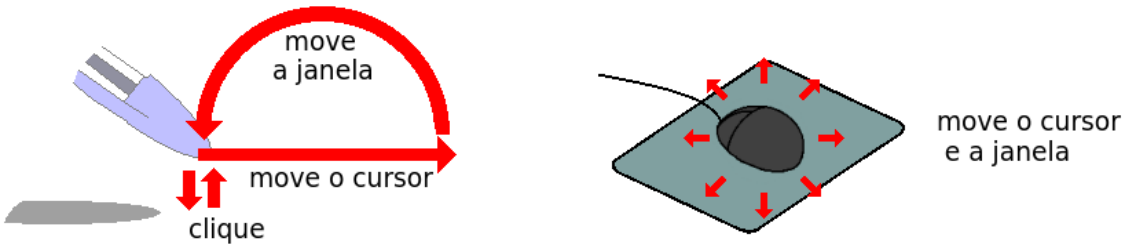
- Todas as cruzes devem estar integralmente visíveis pela webcam, se uma cruz não está sendo identificada, talvez ela esteja demasiado longe ou perto da webcam.
- Se a sua webcam tem autofoco, mexa alguma coisa sobre o papel até focar e depois remova.
- Ao final você será pedido de uma confirmação de se a calibração está correta, como na Figura E.3.

E.1.3 Etapa de Desenho

- Acenda a luminária.
- Utilize a caneta com a tampa fechada, nunca solte tinta no papel. O clique do mouse será simulado quando a tampa da caneta encostar na sua sombra.
- Evite escrever olhando para o papel. Olhe sempre para a tela e utilize o cursor do mouse como referência espacial. Não escreva fora da área delimitada pelas 4 cruzes.
- O controle do mouse é limitado por uma janela de alcance. Você pode seguir um dos nossos dois modos de interação para alcançar a tela toda (Figura E.4) ou alterar o campo “Mouse Range” para uma resolução maior.



(a) Modo normal



(b) Modo "touchpad-like".

Figura E.4: Modos de interação do software. A ilustração mostra como o cursor do mouse e a janela de alcance são movidos pelo mouse e pela caneta.

E.2 Problemas Frequentes

E.2.1 Etapa de Calibração

- **"Há um borrão vermelho no meio do papel"**
  - Causa: Pode ser a sombra do seu próprio braço ou de um pedaço da luminária. Tente remover esta sombra.
- **"O papel não é encontrado (a separação vermelho/branco não parece considerar o papel)"**
  - Causa: Alguma parte da imagem é mais clara que o papel: pode ser a parede, a mesa, roupa branca, etc.. Tente isolar estes objetos.
- **"As cruzes de trás não estão aparecendo"**
  - Tente apagar a luminária.
  - Tente aproximar um pouco o papel da câmera.
- **"As cruzes de trás não estão sendo reconhecidas"**
  - Tente aproximar um pouco o papel da câmera.
  - Você não deve desenhar as cruzes próximas demais à borda do papel.

- **"As cruzes da frente não estão sendo reconhecidas"**
  - Tente afastar um pouco o papel da câmera.
  - Lembre-se de que as cruzes devem estar contidas integralmente na imagem da webcam.
  - Certifique-se de que a webcam está focada corretamente. Se ela tem autofoco, tente movimentar algum objeto em cima do papel para fazê-la focar.

E.2.2 Etapa de Desenho

- **"A caneta não é detectada"**
  - Causa possível: A luminária está apagada.
  - Causa possível: A luminária não é incandescente.
- **"A caneta não está clicando"**
  - Causa possível: Sombra não está nítida o suficiente. Aproxime a luminária do papel, feche as cortinas.
  - Causa possível: O papel não está iluminado de forma homogênea. Tente rabiscar (ainda com a caneta tampada) em outra parte do papel e tente novamente.
  - Causa possível: Você não está segurando a caneta apropriadamente, conforme a Figura E.5.
  - Causa possível: Você está no modo canhoto e é destro ou vice-versa.
  - Causa possível: A caneta não é tipo BIC comum azul.

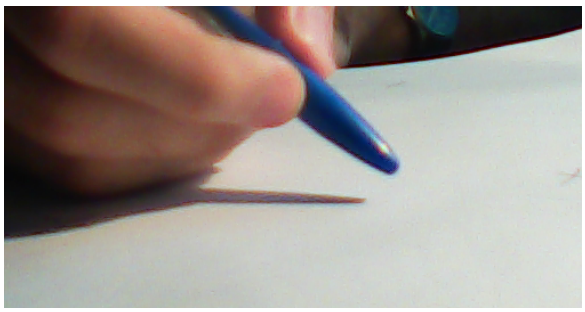


Figura E.5: Forma correta de segurar a caneta (visão da câmera). Segure a caneta com os dedos não muito próximos à ponta, de forma inclinada em relação à mesa (nunca completamente na vertical), apontando mais ou menos para a frente e para a esquerda (caso destro).

- **“O clique solta enquanto estou desenhando”**
  - Causa possível: Você está escrevendo próximo a uma cruz.
  - Causa possível: O papel está dobrado.
  - Confira também os itens “A caneta treme muito” e “A caneta não está clicando”.
- **“A caneta treme muito”**
  - Causa possível: Pode ser que a iluminação não esteja adequada. A caneta não pode aparecer nem muito clara nem muito escura para a webcam. Tente reposicionar a luminária, ligar a iluminação de teto, fechar as cortinas. A lâmpada da luminária deve ser incandescente.
  - Dica: A parte do papel mais próxima à webcam é a que tem melhor qualidade de tracking, você pode preferir desenhar nesta porção.
- **“No instante em que eu toco a caneta no papel, o cursor do mouse se mexe”**
  - Causa possível: O papel não está perfeitamente liso (pode ocorrer se você sua nas mãos). Você pode remover o papel e colocar outro (não é necessário recalibrar) ou escrever sobre alguma outra superfície branca lisa e rígida. Outra alternativa é deixar o modo “touchpad-like” ligado e manter a mão encostada sempre na mesma porção do papel, movendo apenas os dedos.
  - Causa possível: A superfície de contato não é rígida
  - Causa possível: A webcam está inclinada demais para baixo ou girada (rolada) (Veja Figura E.2).
  - Causa possível: A lâmpada não está posicionada à esquerda do papel.
- **“No modo ‘touchpad-like’, quando afasto a caneta do papel, o cursor do mouse dá um salto”**
  - Provavelmente há uma interferência entre a sombra da sua mão com a sombra da caneta. Experimente afastar um pouco a luminária (colocá-la mais para o lado).
  - Outra causa possível são sombras-fantasma. Neste caso você pode tentar uma luminária mais forte (ou aproximá-la do papel)
  - Causa possível: Sua caneta não é BIC (tem a tampa muito grossa).
- **“O cursor do mouse não está se mexendo da forma que eu quero”**

  - Causa possível: Você está desenhando muito próximo à borda da mesa
  - Causa possível: Há objetos azuis estranhos próximos do papel. Nota: Roupa azul (vestida pelo usuário) não costuma causar interferência, a menos que o papel tenha sido posicionado muito próximo à borda da mesa.
  - Causa possível: Há sombras estranhas no papel
  - Causa possível: A sua mão gera uma sombra muito grande no papel (digamos, ocupando a maior parte do papel)

- **"O cursor do mouse fica parado às vezes"**
  - Você não está desenhando fora da área delimitada pelas cruzes?

E.2.3 Outros Problemas

- **"O programa parou de funcionar!"**
  - Causa possível: Você desconectou a webcam ou o mouse do computador durante a execução.

Appendix F

Survey Sent to Testers (Original, in Portuguese)

Você considerou o sistema fácil de configurar?

- Sim, não tive grandes dificuldades.
- Demorei para calibrar as cruzes só, o resto foi fácil.
- Não é difícil, mas também não é fácil, deu um trabalhinho...
- Não, deu bastante trabalho.
- Não, desisti no meio.

Você leu as instruções?

- Sim, li antes de configurar o sistema.
- Mais ou menos, folheei primeiro antes de configurar.
- Não li antes, apenas consultei quando algum problema acontecia.
- Não li.

Quanto tempo você acha que levou mais ou menos para configurar? (se você leu as instruções antes, desconsidere o tempo de leitura)

Você tem experiência com mesa digitalizadora (tablet)?

- Sim, uso com frequência.
- Sim, tenho uma, mas não uso muito.
- Já usei uma vez ou duas.

- ○ Nunca usei.

O que achou da qualidade do controle do mouse?

- ○ Funcionou muito bem.
- ○ Funcionou bem, mas o sensor de pressão de um tablet faz falta.
- ○ Funcionou bem por alguns momentos, mas ele falha muito e é difícil de controlar.
- ○ É *legalzinho*, mas não serve para muita coisa.
- ○ Não fiquei satisfeito.

Marque se você notou e se incomodou com os seguintes problemas

- ☐ Efeito "serifa" (marca ao encostar / retirar a caneta do papel)
- ☐ Mudança da posição do cursor imediatamente antes de clicar
- ☐ Clique indesejado
- ☐ Clique não funciona
- ☐ Quando clico e arrasto o mouse solta sozinho
- ☐ Traço ondulado na diagonal
- ☐ Cursor não se mexe mais

Você usaria o nosso sistema?

- ○ Sim, para mim ele substituiria perfeitamente o tablet.
- ○ Sim, o tablet é melhor mas é muito caro, eu preferiria usar esse.
- ○ Sim, mas acho que depois de um tempo eu ficaria cansado e ia acabar comprando um tablet. OU: Se o meu tablet quebrar eu usaria, sim, temporariamente.
- ○ Não, para mim não serve.
- ○ Não, absolutamente.

Outros comentários:

Bibliography

[1] PIAZZA, T., FJELD, M. "Ortholumen: Using Light for Direct Tabletop Input". In: *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on*, pp. 193–196, out. 2007. doi: 10.1109/TABLETOP.2007.23.

[2] `http://www.wiimoteproject.com/`. Acessado em março de 2014.

[3] `http://laserinteraction.codeplex.com/`. Acessado em março de 2014.

[4] MUNICH, M. E., PERONA, P. "Visual Input for Pen-Based Computers", *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 24, n. 3, pp. 313–328, mar. 2002. ISSN: 0162-8828. doi: 10.1109/34.990134. Disponível em: ¡`http://dx.doi.org/10.1109/34.990134`¿.

[5] YASUDA, K., MURAMATSU, D., SHIRATO, S., et al. "Visual-based Online Signature Verification Using Features Extracted from Video", *J. Netw. Comput. Appl.*, v. 33, n. 3, pp. 333–341, maio 2010. ISSN: 1084-8045. doi: 10.1016/j.jnca.2009.12.010. Disponível em: ¡`http://dx.doi.org/10.1016/j.jnca.2009.12.010`¿.

[6] HAO, Z., LEI, Q. "Vision-Based Interface: Using Face and Eye Blinking Tracking with Camera". In: *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*, v. 1, pp. 306–310, dez. 2008. doi: 10.1109/IITA.2008.177.

[7] LIN, Y.-P., CHAO, Y.-P., LIN, C.-C., et al. "Webcam Mouse Using Face and Eye Tracking in Various Illumination Environments". In: *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pp. 3738–3741, jan. 2005. doi: 10.1109/IEMBS.2005.1617296.

[8] MANCHANDA, K., BING, B. "Advanced mouse pointer control using trajectory-based gesture recognition". In: *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, pp. 412–415, mar. 2010. doi: 10.1109/SECON.2010.5453841.

[9] SEELIGER, I., SCHWANECKE, U., BARTH, P. "An Optical Pen Tracking System As Alternative Pointing Device". In: *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II*, INTERACT '09, pp. 386–399, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN: 978-3-642-03657-6. doi: 10.1007/978-3-642-03658-3_44. Disponível em: ¡http://dx.doi.org/10.1007/978-3-642-03658-3_44¿.

[10] https://qt-project.org/. Acessado em março de 2014.

[11] http://sol.gfxile.net/escapi/. Acessado em março de 2014.

[12] http://msdn.microsoft.com/en-us/library/cc433218%28VS.85%29.aspx. Acessado em março de 2014.

[13] http://eigen.tuxfamily.org/. Acessado em março de 2014.

[14] http://www.x.org/wiki/. Acessado em março de 2014.

[15] http://eigen.tuxfamily.org/. Acessado em março de 2014.

[16] SZELISKI, R. *Computer Vision: Algorithms and Applications*. 1st ed. New York, NY, USA, Springer-Verlag New York, Inc., 2010. ISBN: 1848829345, 9781848829343.