WebcamPaperPen: A Low-Cost Graphics Tablet

Gustavo Pfeiffer, Ricardo Marroquim, Antonio A. F. Oliveira
Laboratório de Computação Gráfica (LCG) - COPPE - UFRJ

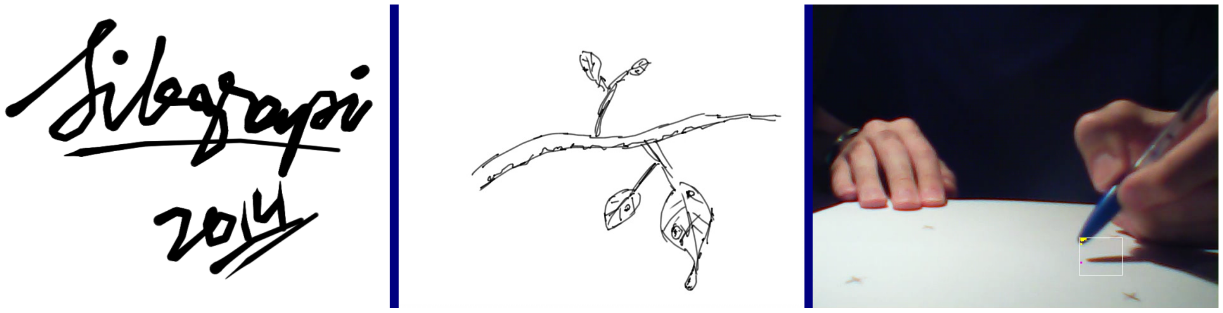


Fig. 1. Drawing and handwriting examples using our system in MyPaint software.

***Abstract*—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 webcam, paper, pen and a desk lamp. None of the existing methods, to our knowledge, solves the task with all these qualities. Our method detects clicks using the pen shadow and computes the mouse cursor position by predicting where the pen tip and its shadow will hit each other. Our method employs a series of image processing algorithms and heuristics to extract interest features with subpixel precision, projective geometry operations to reconstruct the real-world position, and hysteresis filters to improve stability. We also provide an algorithm for easy calibration. Finally, the quality of our method is evaluated with user tests and a quantitative experiment.**

***Keywords*-vision-based interface; paper-pen technology; tracking**

I. 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 (Fig. 1). Our method only requires a webcam, a sheet of paper, a blue-capped pen and a desk lamp: it is 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.

II. 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 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, differently from ours, 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

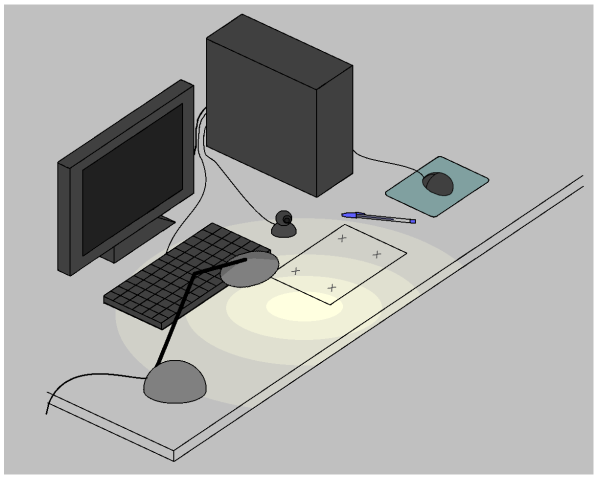


Fig. 2. System setup illustration.

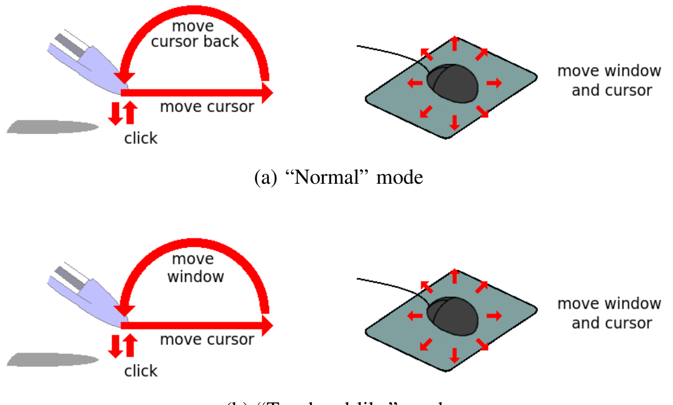


Fig. 3. Interaction modes from our implementation. Illustration shows how the mouse cursor and the mouse range window are moved by the mouse and the pen.

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.

III. System Description

Our system is configured as shown in Fig. 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, 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 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 IV-B); and the drawing step, in which the pen cap and its shadow are tracked (Sections IV-C, IV-D, IV-E and IV-F). 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 captured by the webcam, 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 implementation (see Fig. 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.

IV. Method Description

A. 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).

To make our method work with varied webcam models and illumination configurations, we perform the following normalization: For every image line where $y \equiv 0 \pmod{10}$, we compute the mean intensity within the line. These values are sorted in a list, and we select the 10 lines of greatest intensity (these lines are very likely to contain part of the paper). The image is then normalized to satisfy a mean intensity of 420 in these 10 lines. 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 supposing, without loss of generality, that the mean intensity in these 10 lines equals 420).

All the constants defined in this section were chosen applied to the blue BIC Cristal 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.

B. 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, it 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. In fact, 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 find in the image a region containing reliably only paper and estimate its intensity (Fig. 4(a)). We divide the image in

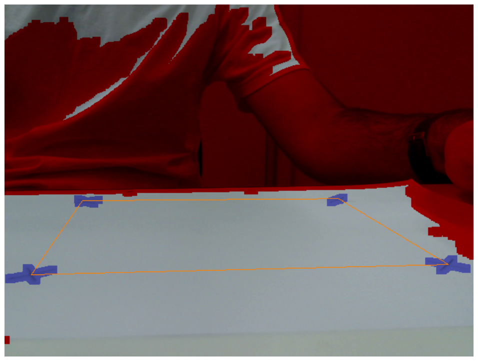


Fig. 5. The calibration procedure classifies areas of the image as paper (white), cross (blue) or unknown (red). The orange lines link the estimated centers of the four crosses. Once calibration converges, the user is asked to confirm calibration, which is done in our implementation by showing the image above to the user.

4 quadrants, and compute the mean 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 (Fig. 4(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. Additionally, to make the algorithm faster, the i-th iteration only considers one of every 2^{5-i} image lines.

Cross detection is done by a threshold-based segmentation algorithm (Fig. 4(c)). A pixel is classified as paper if it is brighter than $0.92w(x, y)$, or non-paper otherwise. This classification is dilated (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 (Fig. 4(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 are computed using

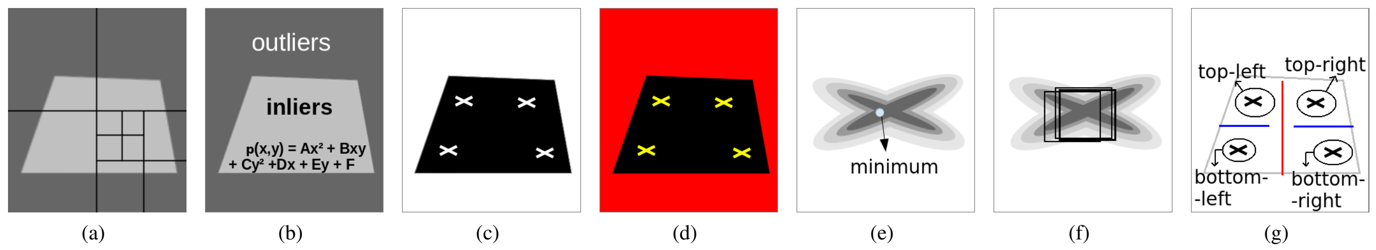


Fig. 4. Illustration of the overall calibration algorithm. **(a)** Search the paper in the image using a hierarchical algorithm, yielding an estimate to the paper intensity. **(b)** Iteratively predict the intensity of the paper in each pixel as a quadratic function of the position. **(c)** Comparing 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. **(e)** Estimate the cross center first by minimizing intensity after blur. **(f)** Update center iteratively using a quadratic fit. **(g)** Classify crosses.

Sobel filters.

The first guess for the position of the center of the cross (Fig. 4(e)) is the pixel of minimum intensity inside the segment after a Gaussian-blur with parameter $\sigma = 5$. This position is then updated (Fig. 4(f)) by fitting a quadratic function (again in the form $u^T Ru$) 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 (Fig. 5). The four crosses are classified as top-left, top-right, bottom-left and bottom-right (Fig. 4(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.

C. 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 in this specific scenario, described as follows.

The first estimate for the pen cap tip position is computed using a blue color filter (Fig. 6(a)): we find the pixel (x, y) that maximizes $2y+x$ and satisfies the following constraints: $B(x+i, y+i) > 60$, $B(x+i, y+i) > 1.6R(x+i, y+i)$ and $B(x+i, y+i) > 1.6G(x+i, y+i)$ for all $i \in \{-8, -7, -6, ..., -1, 0\}$, and there must exist a pixel in the line segment between (x, y) and $(x, y + 50)$ that is found in the convex hull of the four crosses. The reason for taking $i \in \{-8, -7, -6, ..., -1, 0\}$ 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 $(x, y + 50)$ and not only the point (x, y) is that the shadow is expected to be at most 50 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 $\tilde{z} = (x, y)^T$.

As the previous estimate is imprecise and often located in a more inner portion of the cap, our next step is to attempt to pull

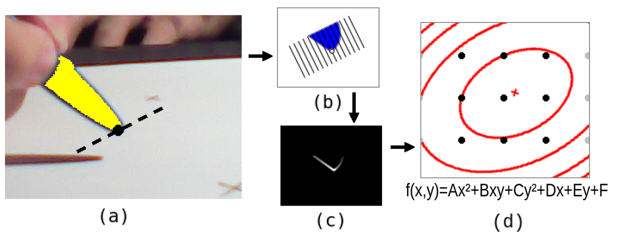


Fig. 6. Illustration of the overall pen cap tip tracking algorithm: **(a)** Apply pen-blue filter and maximize 2y+x; **(b)** Minimize intensity sum of tilted column and maximize Sobel within the column); **(c)** Maximize objective function f; **(d)** Maximize fitted quadratic function from 9 pixels

this estimate closer to the actual tip (Fig. 6(b)). $\tilde{z}$ is refined to $\tilde{z}' = \tilde{z} + T\Delta$ for $T = \begin{bmatrix} 1 & 1/2 \\ -1/2 & 1 \end{bmatrix}$ 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 $\Delta_1 = \arg\min_{i \in \{-6,...,6\}} c(i-1) + 2c(i) + c(i+1)$ where $c(i) = \sum_{j=-2}^{10} p(\tilde{z} + T\begin{bmatrix} i \\ j \end{bmatrix})$, and $p(x, y)$ for non-integer x or y is computed using bilinear interpolation; then $\Delta_2 = \arg\max_{j \in \{-2,-1,...,10\}} \partial_j p(\tilde{z} + T\begin{bmatrix} \Delta_1 \\ j \end{bmatrix})$, where ∂_j is computed using a tilted Sobel filter:

$$\begin{aligned}\partial_j p(r) = & p(r + T\begin{bmatrix} -1 \\ 1 \end{bmatrix}) + 2p(r + T\begin{bmatrix} 0 \\ 1 \end{bmatrix}) + p(r + T\begin{bmatrix} 1 \\ 1 \end{bmatrix}) \\ & - p(r + T\begin{bmatrix} -1 \\ -1 \end{bmatrix}) - 2p(r + T\begin{bmatrix} 0 \\ -1 \end{bmatrix}) - p(r + T\begin{bmatrix} 1 \\ -1 \end{bmatrix})\end{aligned}$$

also using bilinear interpolation when necessary. $\tilde{z}'$ is rounded down in the end. Note that the search window is not symmetric in j as we know $\tilde{z}$ is unlikely to be located below $\tilde{z}'$.

$\tilde{z}'$ is further refined (Fig. 6(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 (argument of maximum 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 $(sinc(x/12)sinc(y/12))^2$ [1] and ∂_y is the $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \star$ cross-correlation filter.

[1]We use the following definition for the sinc function: $sinc(x) = \frac{\sin(\pi x)}{\pi x}$

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 V-A). 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 (Fig. 6(d)) is computed by fitting the objective function described above to a quadratic function (again in the form $u^T Ru$) 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.

D. 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 knows where the click is going to be performed beforehand. 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 clicking direction d (assuming a linear movement of the user's hand) and the position l of the light source projected onto the table following this direction, the hitting position will be at $h = (s \times l) \times (z \times d)$, in homogeneous image coordinates. See Fig. 7 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 $h = (z_1, s_2, 1)^T$. This assumption does make some restrictions to the positions of the webcam and the lamp: the lamp should not be too near to the paper and the webcam may not be too inclined downwards (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 $(x, y) \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] = [\lfloor z_1 \rfloor - 65, \lfloor z_1 \rfloor + 3] \times [\lfloor z_2 \rfloor - 10, \lfloor z_2 \rfloor + 49]$, 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.

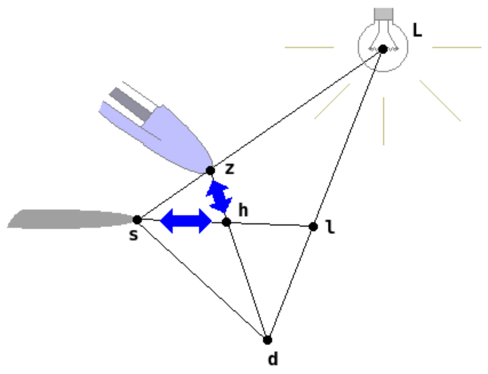


Fig. 7. Let L be the position of the light in homogeneous image coordinates. Then L, z and s are collinear, and their projection onto the desk following direction d, respectively l, h and s must also be collinear. Therefore, when the user moves the pen down, as the hitting point h remains constant, z must move on the $\overline{zhd}$ line and s on the $\overline{shl}$ line, resulting that $h = (s \times l) \times (z \times d)$.

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 $\widetilde{s}_2$. 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 after gamma correction, 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 $y \in [\widetilde{s}_2, \widetilde{s}_2 + 1]$ where $g(y) = \bar{g}$, we first compute for every x where $p(x, \widetilde{s}_2) > w$ and $p(x, \widetilde{s}_2 + 1) \leq w$ or vice-versa the value y_x satisfying $p(x, y_x) = w$. These values are sorted in a list, and we set s_2 as the first value y_x such that $g(y_x) \geq \bar{g}$.

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 (Fig. 8).

Other approaches such as plain linear interpolation of $g(y)$ are inappropriate because in fact $g(y)$ is usually not

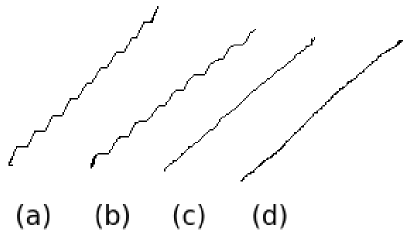


Fig. 8. Diagonal lines as they appear using **(a)** no interpolation, **(b)** linear interpolation of $g(y)$, **(c)** our method and **(d)** a graphics tablet. All interfaces used a resolution of $W_F \times H_F = 1920 \times 1080$ and an input area sized 15cm×9.2cm. (Drawn in Kolourpaint software)

quite linear between $\widetilde{s}_2$ and $\widetilde{s}_2 + 1$. Usually the shadow appears almost tangent to the horizontal axis in the transition 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.

E. Mouse Motion

The position of the mouse cursor within the mouse range window is computed by applying a rectification (homography) 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:

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

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

F. 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 $B > 30$, $B > 1.6R$ and $B > 1.6G$ — in a 30×6 window centered 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 the coefficient of variation $S = \sigma/\mu$ 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, 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 $L_t = 0.95L_{t-1} + 0.05L_0$.

V. Results

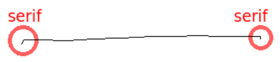
We first introduce in Section V-A the main problems we find by using a common pen to mimic a graphics tablet under our approach, and then we show in Section V-B how these problems are perceived in user tests. Finally, in Section V-C we present a quantitative measurement of the precision of our tracking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

A. Limitations

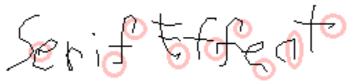
First of all, our system may misbehave (e.g.: perform undesired clicks, release the mouse button during drag-and-drop, not accept clicks on a determined region of the paper, suddenly warp the mouse cursor to another position during one frame, among other problems) if the setup was not appropriate (i.e. the configuration on illumination, webcam, paper, etc.). As we did not employ more generic and robust techniques to judge "if the detected pen was actually a pen", or "if the detected shadow was actually a shadow", the setup restrictions may not be so straightforward. 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 setup limitation, there is one type of artifact that is inherent to this method of simulating mouse input, which we call the "serif" effect.

The "serif" effect (Fig. 9) 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 captured click will not be at the position the user intended. 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 the algorithms attempting to minimize this cause, however, there is a second cause, over 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



(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)

Fig. 9. "Serif" effect examples.

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 the case (as one can see in Fig. 9).

B. User Tests

We made a survey with 30 voluntary testers asking them to set up the system and try to use it, provided a setup manual.

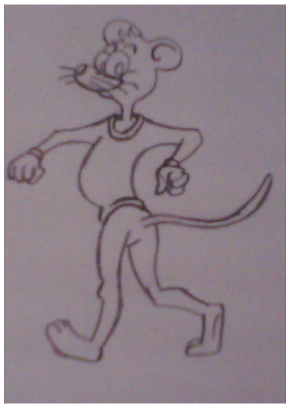
Most users (67%) reported having found the system easy to set up (rating it 1 from 1 to 5), and estimated having required, on average, 4min15s for the task (the shortest time reported was 15s, while the longest was 15min) [2]. Precision was evaluated as modest: 47% said the mouse control worked satisfactorily (rating 1 from 1 to 5), while 30% said that it misbehaves too often and is difficult to control (rating 3). Nevertheless, in general, users would accept using our system replacing the graphics tablet: 20% answered that the graphics tablet could be replaced perfectly by our system for them, while 47% said that the graphics tablet is better but too expensive, preferring thus our system.

Although 63% of the users had very little to no experience with the graphics tablet, there does not seem to be a correlation between experience with the graphics tablet and the acceptance to our system.

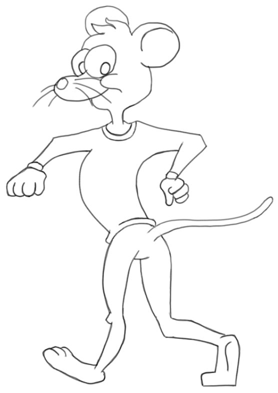
The most often reported defects were undesired clicks and the "serif" effect, reported by, respectively, 47% and 40% of the users. Those who tried drawings were particularly disturbed by the "serif" effect, and some were also uncomfortable with the restrictions on the way of holding the pen.

We also asked some users to make comparisons between the graphics tablet and our system, and the result can be seen in Figures 10 and 11.

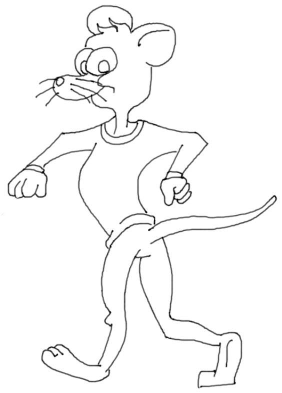
[2] Some users interpreted "time to set up" to be the time spent in the calibration step, others as the time spent until one can make the system work correctly.



(a) Drawn by hand (photograph)



(b) Drawn using a graphics tablet in MyPaint software



(c) Drawn using our system in MyPaint software

Fig. 10. Comparison for drawing applications. All the three drawings were drawn by the same user.

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 submillimetric thickness, being at the same level of the surface, 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.

C. 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.

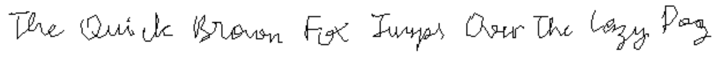
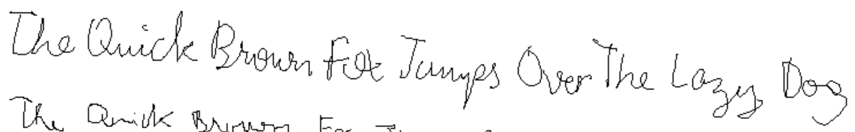
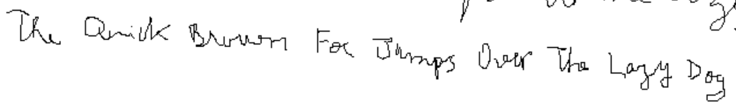
Interface	Drawing Time	Output
Webcam-PaperPen	23.82s	
Graphics Tablet	22.72s	
Mouse	62.21s	

Fig. 11. Comparison of our system with graphics tablet and optical mouse. We used a range window of 640 × 480 in our system 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.

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 $\sigma_f = \sqrt{\frac{1}{2}\text{E}[(f^t - f^{t-1})^2]}$ [3] 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, due to the positioning of the webcam.

VI. 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 V-A, 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 investigate ways to eliminate the "serif" effect completely.

An extension of this work would be to obtain 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.

VII. Acknowledgments

We would like to thank the voluntary testers, and laboratory colleagues Marcello Salomão, Claudio Esperança and Leonardo Carvalho for their collaboration to this work.

References

[1] T. Piazza and M. Fjeld, "Ortholumen: Using light for direct tabletop input," in *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on*, Oct. 2007, pp. 193–196.

[2] http://www.wiimoteproject.com/, accessed in March 2014.

[3] http://laserinteraction.codeplex.com/, accessed in March 2014.

[4] M. E. Munich and P. Perona, "Visual input for pen-based computers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 313–328, Mar. 2002.

[5] K. Yasuda, D. Muramatsu, S. Shirato, and T. Matsumoto, "Visual-based online signature verification using features extracted from video," *J. Netw. Comput. Appl.*, vol. 33, no. 3, pp. 333–341, May 2010.

[6] Z. Hao and Q. Lei, "Vision-based interface: Using face and eye blinking tracking with camera," in *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*, vol. 1, Dec. 2008, pp. 306–310.

[7] Y.-P. Lin, Y.-P. Chao, C.-C. Lin, and J.-H. Chen, "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*, Jan. 2005, pp. 3738–3741.

[8] K. Manchanda and B. Bing, "Advanced mouse pointer control using trajectory-based gesture recognition," in *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, Mar. 2010, pp. 412–415.

[9] I. Seeliger, U. Schwanecke, and P. Barth, "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*, ser. INTERACT '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 386–399.

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