Low-cost and efficient eye-controlled writing

(six month progress report)

Piotr Zieliński

October 11, 2006

This document discusses the topics I have been working on for the last six months, the current state of the research, and most probable directions of future work.

1 Introduction to Dasher

The goal of the Dasher Project [1] is to develop an efficient eye-controlled system, which can be used to enter text on any computer equipped with a standard £50 webcam. Our approach is not only radically cheaper than commercial solutions, which cost at least £2000, but also more efficient: Dasher can be eye-driven at 25 words per minute, whereas on-screen keyboards (its main competitor) achieve only 15 wpm [2].

Dasher achieves this speed using a simple method that makes use of the human brain's excellent abilities at image processing and navigation [3]. Users input text by moving their gaze focus through a sequence of labelled boxes (Figure 1). For example, to write "this", the user first steers towards the pink box labelled "T", then towards the green box "h" contained in the box "T", and finally towards the pink "i" and white "s". The image on the screen is being constantly magnified as the user navigates through the letters. Box sizes are proportional to letter probabilities determined by an advanced language prediction system [4], which facilitates writing common words and phrases.

2 Two-dimensional Dasher

The two-dimensional Dasher project attempts to fix a problem with the original version, in which most of the letters are tightly squeezed along the right edge of the input box (Figure 1). As a result, almost all navigation is vertical, and the horizontal component of user input is mostly ignored. In theory, one could double the input speed by utilizing both dimensions equally.

Two-dimensional Dasher (Figure 2) is my prototype implementation of the above idea, which distributes the letters uniformly over the entire input box. This approach has two major advantages over the standard Dasher. First, both

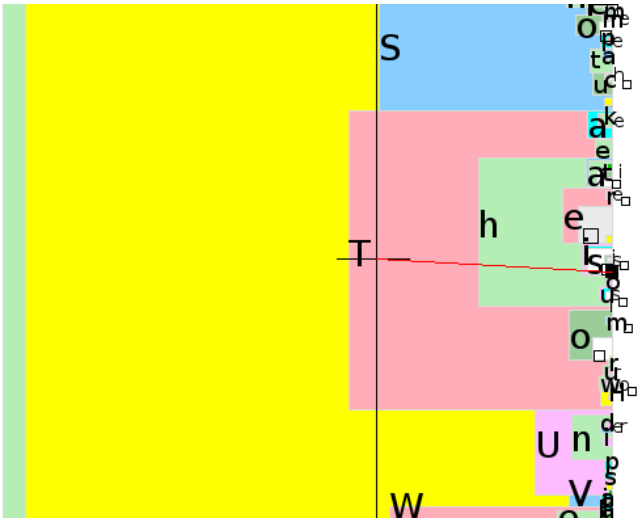


Figure 1: Writing the word “this” with Dasher. Other visible words: some, such, take, that, the, there, this, to, Tom, under, up, us.

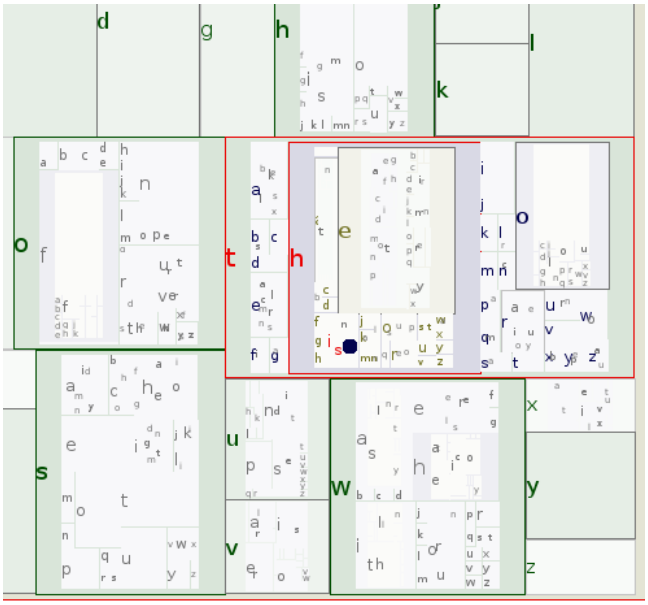


Figure 2: Writing the word “this” with 2D Dasher. Other visible words: his, of, on, or, out, othe., our, over, tea, the, there, they, them, that, thy, to, too, turn, try, two, say, said, she, sin, sit, up, us, use, very, war, was, were, why, Sam, whe.., wha., with, win, won.

vertical and horizontal navigation input is used, which makes it ideal for gaze-based control. Secondly, the two-dimensional layout uses the available display space much better, so the same information can be displayed on a smaller screen, which is critical for deployment in mobile devices. The main challenge in two-dimensional Dasher is designing a layout and a navigation scheme that are both space-efficient and easy for the human brain to process at high speeds.

Layout. I have investigated three possible philosophies of arranging letters inside a box.

1. **Static.** In this method, letters in each box are arranged in exactly the same way (Figure 4). For example, `abcd` in the first line, `efghi` is the second, etc. The advantage of this scheme is that the user knows where the letters are and will not waste time looking for them. On the other hand, the lack of adaptivity of this scheme results in wasting some space by not using the entire box. This is because this scheme must use square-like boxes; otherwise we might end up with unusably thin or wide boxes in a few iterations (Figure 7).

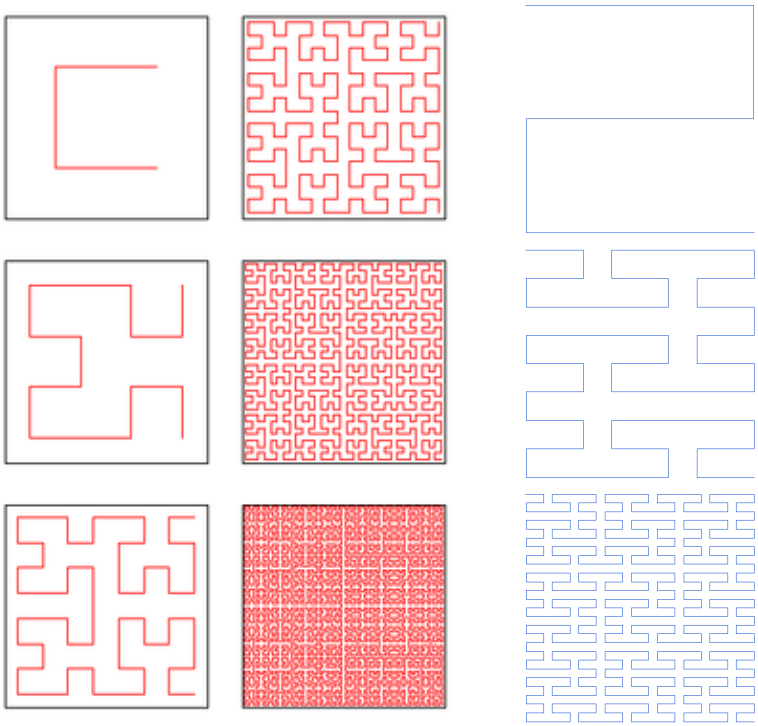
2. **Adaptive.** The problem of wasting space can be overcome by using space-filling curves (Figure 3) to come up with close-to-square boxes that fill the entire box (Figure 6). The disadvantage here is that all these layouts can change from box to box in an unpredictable (for a human being) way. As a result, the user wastes a lot of time locating letters.

3. **Independent zoom.** Another method of dealing with thin or wide boxes (Figure 7) is decoupling the vertical and horizontal zoom levels, and rescale the picture so that the current box looks like a square. However, this method result in a unnatural experience, and tends to confuse the user as well.

I have implemented all above layouts, and concluded that the simplest one (static) is probably the best, as fast as the overall speed and comfort are concerned. To deal with wasted space, I slightly increase the size of small boxes, if there is room (Figure 6). The optimal design might also incorporate some independent zoom, small enough not to cause too much confusion.

Apart from speed and comfort, static designs have one more advantage over the dynamic ones. We observed that in static designs users quickly learn the sequence of letters in common words *geometrically*, which gives an additional speedup. In other words, they learn the position of consecutive letters and know where to look for them, even before those letters appear on the screen.

Psychophysical factors. I found out that the main source of user confusion are rapid and non-continuous changes in the user interface, and eliminating them increases the comfort. This is true not only in layout, but also in the colour scheme used. All changes to the user interface, such as new letters or boxes appearing, are made gradually by a graceful transition from the parent

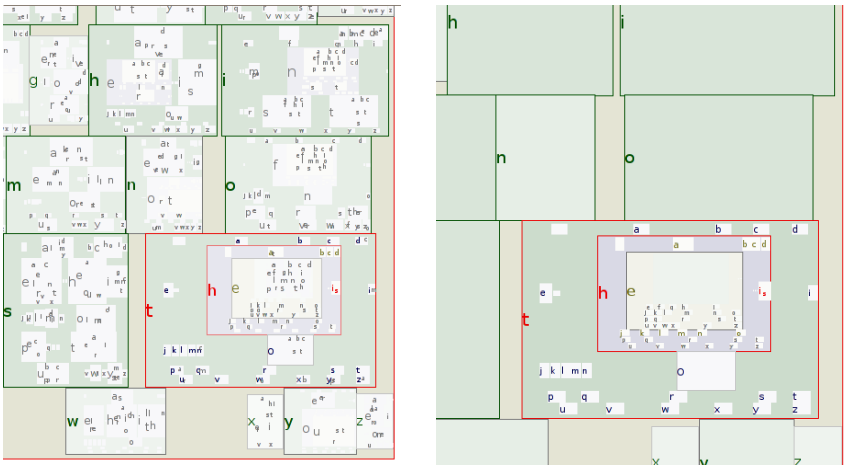


(a) Hilbert curve (b) Peano curve

Figure 3: Several iterations of two space-filling curves.

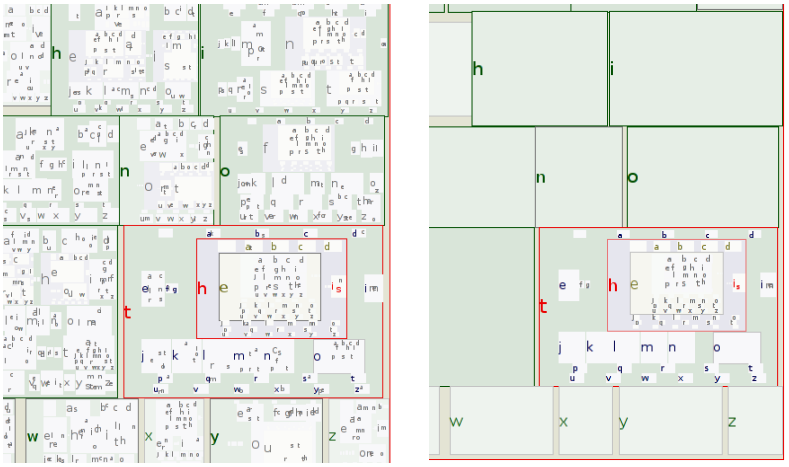
colour to the target colour. Entire boxes are coloured, as opposed to only borders, which lets the user quickly assess the situation without looking away from the centre of attention. Finally, the current sequence of letters and boxes are highlighted in bright red to prevent the user losing track of the letters.

Future work. Some work remains to be done in tuning the layout, the dynamic colour scheme, and maybe coming up with new anti-confusion ideas. However, the most important goal now is to integrate the program with other input devices (gaze tracker), and to rewrite the current Python prototype into a production quality software. I would also like to replace the current simple dictionary-based model used by 2D Dasher with the advanced language prediction system from standard Dasher [4].



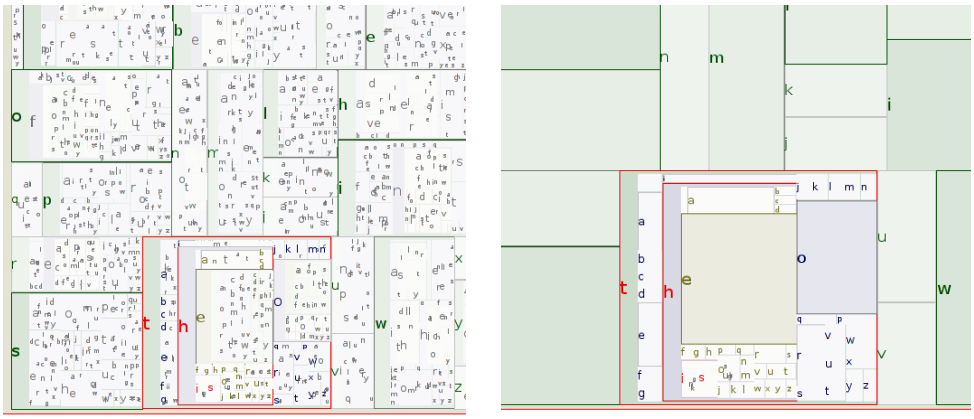
(a) all boxes visible, confusing (b) only near boxes visible

Figure 4: Two-dimensional Dasher with the standard static box arrangement. The letter arrangement in all boxes in the same. As a result, keeping the box share not too far from a square results in a lot of wasted space.



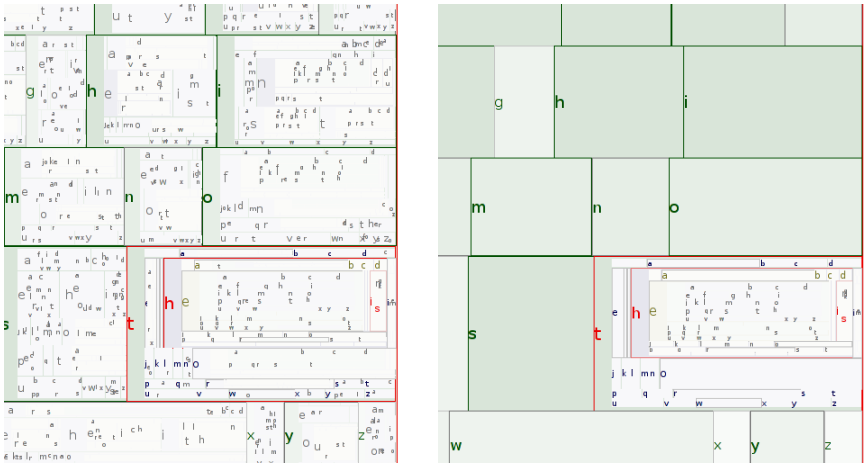
(a) all boxes visible, confusing (b) only near boxes visible

Figure 5: Two-dimensional Dasher with the extended static box arrangement. As in the standard version, the letter arrangement in all boxes in the same, and the boxes themselves are square-like. However, some small boxes are bigger than probabilistically justified, to reduce the space waste.



(a) all boxes visible, confusing (b) only near boxes visible

Figure 6: Two-dimensional Dasher with the dynamic box arrangement. Square-like boxes are distributed along the Peano curve from Figure 3. No space is wasted, however, the letter arrangement changes from box to box, which is very confusing.



(a) all boxes visible, confusing (b) only near boxes visible

Figure 7: Two-dimensional Dasher with the zoom box arrangement. The letter arrangement in all boxes is the same and no space is wasted. This leads to some boxes being very thin or tall, which makes navigation impossible. The problem can be rectified by decoupling horizontal and vertical zoom levels, to maintain a sensible aspect ratio of the current box.

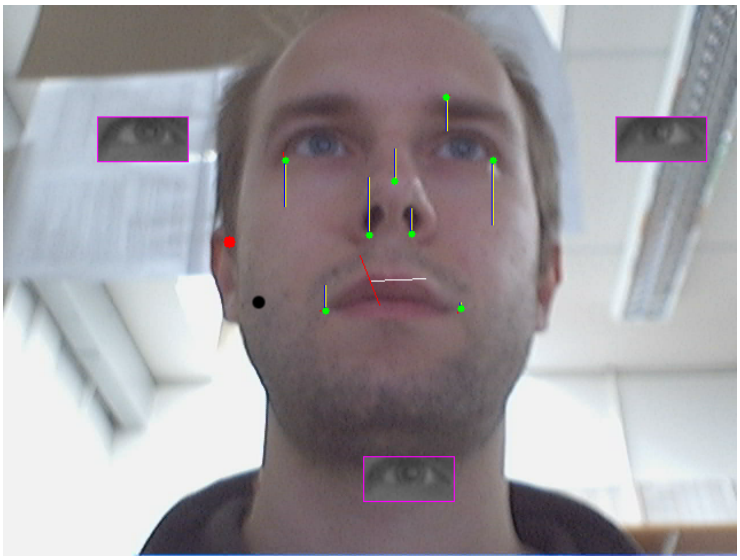


Figure 8: Gaze-tracking with a webcam. Green points are tracked facial features, and the vertical lines their inferred depth with respect to the camera. Red and white lines represent the estimated 3D head pose. The black dot is the real gaze focus, the red dot is the gaze focus inferred from the current eye image and 3 images acquired during the calibration. With 25 images, the average accuracy is 1.2 degrees vertically and 1.6 horizontally.

3 Head tracking

The first step towards gaze tracking is head tracking. At the moment, I am using simple algorithms to track outer eye corners, and other facial features that can be chosen at random. I developed an algorithm that, from the tracker outputs, computes the relative 3D rotation of the face. At the same time, the position of eye corners are used to extract the eyes images and normalize them by appropriate scaling and rotation (Figure 8).

The 3D rotation estimation is object-agnostic: it works without any knowledge of the object being tracked (a face). For this reason it requires calibration.

Future work. At the moment, points to track are selected manually. In the future, I plan to employ off-the-shelf face finder and eye locators to initialize those points automatically. I do not plan to focus on tracking accuracy, because it is already better than the gaze tracking part can ever achieve, so improving it will not make a difference (this is because a head is 10x bigger than an eye therefore much easier to track).

4 Gaze tracking

Gaze tracking is performed using a machine learning approach, similar to that described by Williams et al. [5]. Initially, I gained some insight into the tracker implementation by porting it into Linux. I soon realized however, that some changes needed to be made to accommodate a change of a goal. The original software was designed for a webcam equipped with a telelens or, equivalently, placed around 5cm from the eye. In our setting, we have a normal webcam located under the monitor, 30–40 cm from the user. As a result, the quality of the image is degraded correspondingly. The fact that most common cameras lossily compress 640×480 images does not help either (this problem did not exist in [5] as he used a better, but not common, camera).

To address these problems, I have written a new prototype gaze tracker from scratch. There are two main changes to the algorithm. First, I use template eye corner trackers, instead of the machine learning approach. Second, since the input image is 5x smaller than in [5], the filtering algorithm used is also different. While [5] used directional filters, I found out that using the opposite (moderate blurring) significantly improves the accuracy. At the moment, with a still head, the system achieves horizontal and vertical accuracy of 1.2 and 1.6 degrees, respectively.

I have recently performed a comparison between different filtering algorithms (median filter and other custom filters) and found out that simple blurring generally performed at least as good as them. In all the comparisons I was using a set of 25 calibration eye images with known gaze location. The accuracy can be additionally improved by using eye images in which gaze location is *not known*. These images, the large number of which can easily obtained, can be used to teach the machine learning engine how a low-resolution eye image should look like. I tested this technique with 72 unlabelled eye images, improving the accuracy to 0.9 and 1.2 degrees, respectively. This technique (PCA), however, can be computationally expensive.

Future work. One possibility is an automatic search to optimize the filtering algorithm, however, I do not think there is space for a significant improvement here. Instead, I would prefer to investigate whether increasing the number of unlabelled eye images leads to a noticeable improvement in accuracy. So far, all methods used are linear; non-linear methods (KPCA [6], LLE [7]) might be worth a try.

My general feeling is, however, that we are approaching the limit to the gaze information that can be extracted from such low-resolution images. I believe that a bigger improvement can be achieved by developing error-compensation algorithms especially designed for 2D Dasher. Averaging the input from both eyes can also decrease the average error.

5 Other research

This section briefly describes other research topics I was investigating recently.

Distributed computing. Consider the following example of online banking. Assume Alice and Bob simultaneously and independently decide to withdraw everything from their shared £1000 account. Of course, they cannot both get £1000, so only the first of them will succeed. However, who should get the money if, due to communication delays, Alice's request arrives first in the London branch, but not in the Cambridge one?

The requirement of handling such complicated, but very rare, scenarios correctly makes most protocols slower also in common simple cases. Recently, I have shown how to improve the common-case performance while maintaining correctness in all cases, by exploiting the fact that most of the time the clocks at different branches are very well synchronized [8]. I also developed a generic method of constructing such protocols easily, even if some of the participants are corrupt [9]. This method can then be used to design new such protocols in a fully automatic way [10]. In the future, I plan to develop similar automatic methods for other problems in distributed computing.

Computer Security. Imagine the following scenario. In a small room, a number of the Security Council delegates gathered around a table to decide whether to invade an enemy country. One delegate wishes to veto the measure, but worries that such a move might jeopardize the relations with some other member states. Can he successfully veto the proposal without revealing his identity?

The delegate's job is not easy. All communication is public, so everybody can hear everything he says. There are no pre-established secret keys he could use to encrypt his messages. Besides, any such messages would reveal his identity to the receiver anyway.

Feng Hao and I [11] have recently investigated this anonymous veto problem in the committee decision-making settings. In particular, we demonstrated how a member can veto that decision without revealing his/her identity, even if almost all others conspire against him/her . Our protocol makes heavy use of large groups of prime order. In the future, we plan to extend it to also anonymously count the number of vetoes.

6 Future work

There are several open research questions in the project, some more important than others. In my opinion, the most important task now is the integration of the gaze tracker output with 2D Dasher to see what improvements can be done at this level. I would also like to incorporate the information from the 3D head position tracker into the output of the gaze tracker.

References

[1] Dasher Project, 2005. URL `http://www.inference.phy.cam.ac.uk/dasher/`.

[2] D. J. Ward and D. J. C. MacKay. Fast hands-free writing by gaze direction. *Nature*, 418(6900):838, 2002.

[3] David J Ward, Alan F Blackwell, and David J C MacKay. Dasher: a data entry interface using continuous gestures and language models. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2000.

[4] W. Teahan. Probability estimation for PPM. In *Proceedings of the New Zealand Computer Science Research Students' Conference*, Univ. of Waikato, Hamilton, New Zealand, 1995.

[5] O. Williams, A.Blake, and R.Cipolla. Sparse bayesian regression for efficient visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(8):1292–1304, August 2005.

[6] B. Scholkopf, A. Smola, and K.R. Muller. Kernel Principal Component Analysis. *Advances in Kernel Methods-Support Vector Learning*, pages 327–352, 1999.

[7] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by Locally Linear Embedding. *Science*, 290(5500):2323, 2000.

[8] Piotr Zieliński. Low-latency Atomic Broadcast in the presence of contention. In *Proceedings of the 20th International Symposium on Distributed Computing*, Stockholm, Sweden, September 2006.

[9] Piotr Zieliński. Optimistically Terminating Consensus. In *Proceedings of the 5th International Symposium on Parallel and Distributed Computing*, Timisoara, Romania, July 2006.

[10] Piotr Zieliński. *Minimizing latency of agreement protocols*. PhD thesis, Computer Laboratory, University of Cambridge, 2006. Chapter 3, paper in preparation.

[11] Feng Hao and Piotr Zieliński. A 2-round anonymous veto protocol. In *Proceedings of the 14th International Workshop on Security Protocols*, Cambridge, UK, March 2006.