HAL
open science

Single User MultiTouch on the DiamondTouch: From 2x1D to 2D

François Bérard, Yann Laurillau

To cite this version:

François Bérard, Yann Laurillau. Single User MultiTouch on the DiamondTouch: From 2x1D to 2D. Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, 2009, New York, NY, United States. pp.1–8, 10.1145/1731903.1731905 . hal-00953295

HAL Id: hal-00953295

https://inria.hal.science/hal-00953295

Submitted on 28 Feb 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Single User MultiTouch on the DiamondTouch: From 2 x 1D to 2D

François Bérard, Yann Laurillau
LIG, University of Grenoble
BP 53, 38041 Grenoble Cedex 9, FRANCE
(Francois.Berard,Yann.Laurillau)@imag.fr
(+33)476 514 365

ABSTRACT
The DiamondTouch is a widely used multi-touch surface that offers high quality touch detection and user identification. But its underlying detection mechanism relies on two 1D projections (x and y) of the 2D surface. This creates ambiguous responses when a single user exercises multiple contacts on the surface and limits the ability of the DiamondTouch to provide full support of common multi-touch interactions such as the unconstrained translation, rotation and scaling of objects with two fingers. This paper presents our solution to reduce this limitation. Our approach is based on a precise modeling, using mixtures of Gaussians, of the touch responses on each array of antennas. This greatly reduces the shadowing of the touch locations when two or more fingers align with each other. We use these accurate touch detections to implement two 1D touch trackers and a global 2D tracker. The evaluation of our system shows that, in many situations, it can provide the complete 2D locations of at least two contacts points from the same user.

Author Keywords
DiamondTouch, input device, multitouch, tracking, expectation maximization

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—*Input devices and strategies, prototyping*

INTRODUCTION
Tabletops and interactive surfaces require some sensing mechanism in order to report touch locations. Many technical solutions have been offered in the literature [3, 4, 6, 11, 14, 15]. However, none of them is clearly superior to the others: they all carry strengths and weaknesses, as will be detailed in section "Related Work" below.

The DiamondTouch [3] is a tabletop device that offers ease of setup, robustness, fine spatial and temporal resolution, limited pressure required for touch detection, excellent touch threshold (i.e. the contact is generally detected as expected by users), and the unique ability to associate touch locations to identified users. This makes this device a frequently used research tool [2, 5, 7, 9, 10, 12, 16]. However, we believe that its use is bounded by a strong limitation: the DiamondTouch is designed for a single contact point per user.

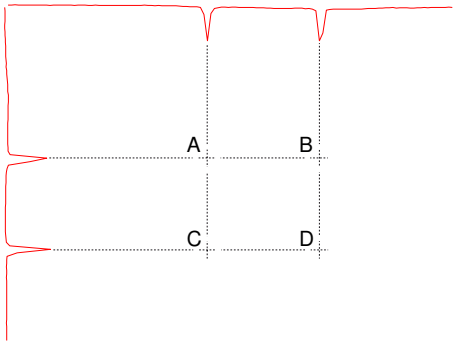

Figure 1. Signals from the horizontal and vertical arrays of sensors of the DiamondTouch (plain lines on top and left) when two fingers of the same user are in contact. Touching the surface in A and D or B and C generates the same output.

The surface of the DiamondTouch embeds two arrays of conductive antennas, one vertical and one horizontal, for touch sensing. While the DiamondTouch can handle the simultaneous contact from 4 different users, it cannot directly locate multiple contacts from the same user. When two fingers from a single user touch the surface, the DiamondTouch output is ambiguous, as illustrated on Figure 1. There is no information to match the peaks on the horizontal array to the peaks on the vertical array. Hence the DiamondTouch does not provide full support of common multi-touch interactions such as the ubiquitous translation, rotation and scaling of images with two fingers. While some efforts have been produced to extract more than a single point location per user on the DiamondTouch [16], none have provided a solution to extract the 2D location of multiple touches from a single user.

In this paper, we present our approach to mitigate the one finger per user constraint of the DiamondTouch. After reviewing the related work, we present the principle of our approach. It is based on the temporal tracking of response

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ITS '09, November 23-25, 2009, Banff, Alberta, Canada
Copyright 2009 ACM 978-1-60558-733-2/09/11... $10.00.

peaks. We show that this approach lacks robustness when implemented with a simple localization of peaks. We thus propose to model the DiamondTouch arrays' outputs as mixtures of Gaussians. This modeling provides an accurate tracking of 1D peaks, which is used in our tracking algorithm described next. Finally, we report about the evaluation of our approach and conclude.

RELATED WORK

Tracking multiple fingers of the same user has been achieved through different means, each having its strengths and weaknesses.

Frontal visual tracking [6, 14] offers the potential to extract a rich set of informations (hand shapes, document contents), but it is sensitive to occlusion, and robustness is hard to achieve as the visual scene being processed is uncontrolled. Rear-surface visual tracking using direct infrared light [15] offers a better control on the scene, but lights in the background must be controlled as they may appear as contact points. Frustrated total internal reflection [4] also suffers from background lights, and often requires a significant pressure on the surface for touches to be detected. The robustness of both rear-surface visual tracking may be reduced by surface contamination (marks left by the fingers). Also, the fine calibration of parameters may be required, for example for the light intensity thresholds used for the image segmentation and for defining the touch threshold.

The SmartSkin [11] follows a different approach than visual tracking: it is similar to the DiamondTouch in the sense that capacitive sensors are embedded in the surface for touch detection. Hence, it has the good properties of capacitive sensing: ease of setup, robustness, limited pressure required. The SmartSkin is superior to the DiamondTouch in the way that touch locations are directly reported as 2D positions, hence the SmartSkin does not suffer from the DiamondTouch ambiguity problem (Figure 1). However, 2D detection on the SmartSkin requires more complex electronics than the DiamondTouch, and makes it less scalable in term of spatial and temporal resolution. To our knowledge, the DiamondTouch is the only large scale (31 or 43 inches in diagonal) capacitive sensing surface available.

The DViT™technology [13] embeds custom designed cameras in the corners of a frame surrounding the interactive surface, the frame itself being equipped with LEDs. The cameras scan a thin volume above the surface. Hover and touch are detected when an object occludes the LEDs' light from the cameras. The high frequency of the cameras (100 Hz) offer excellent temporal resolution, and fingers can be localized even while hovering. However, this approach suffers from the same kind of ambiguity as with the DiamondTouch: when two objects get in contact with the surface, each camera detects two occlusions (in the general case) but there is no way to associate each occlusion of a camera to the occlusions of the other cameras. Indeed, it appears that the DViT approach could benefit from techniques similar to the ones presented in this paper in order to allow multiple touch locations.

Finally, while some clever processing can be done to infer which user is interacting with the surface [8], the DiamondTouch is currently the only solution that directly associates touch locations to identified users. Identifying users is a strong feature for collaborative applications, which are some of the main applications of interactive surfaces. Hence, this is a strong motivation to improve the DiamondTouch.

With regard to overcoming the single touch design of the DiamondTouch, Wu et al. [16] manage to create a rich set of multi-touch gestures for menu activation and object manipulation on a DiamondTouch. The careful design of their gestures sidesteps the ambiguity in the DiamondTouch data, but they do not overcome its limitations. We are not aware of any published effort related to the recovery of the 2D locations of multiple touch of a single user on the DiamondTouch.

TEMPORAL TRACKING AND IT LIMITATIONS

A simple idea to alleviate the ambiguity of the DiamondTouch data is to analyze touch positions over time instead of analyzing every data frame independently of the others. In other word, this is tracking touch locations over time. For example, assuming that touches have been localized in the neighborhoods of point A and D in the previous data frame (see Figure 1), then in the current frame A and D are more probable candidates than B and C. However, the tracking approach has two main limitations: bootstrapping and shadowing.

Bootstrapping

Tracking requires that the ambiguity has been removed in previous frames in order to *solve* the current frame. But users may initiate several contact at the same time, such as touching in A and D in the same data frame. The DiamondTouch samples all its antennas at a frequency of 40 Hz per user. We conducted an informal experiment where participants were asked to perform a scaling gesture on the surface: touching the surface with two fingers and moving the fingers apart. We observed that most of the time the response peaks of the two fingers appeared in the same data frame. Even worse, for one single contact, the response peak on the vertical and horizontal arrays may appear in successive data frames, making the association of peaks even more difficult in case of several contacts. In this situation, the tracking approach has no history to look for previous contact locations, and it is not possible to remove the ambiguity: there is no previous nor external information to solve the problem of Figure 1. In the conclusion of this paper, we discuss two envisioned ways to deal with bootstrapping. In the rest of the paper, we assume that users are cooperating by always making contacts successively. We report, in the "Informal evaluation" section below, that this does not appear as a notable encumbrance.

Shadowing

An other problem appears when two fingers align horizontally or vertically. For example when two fingers align vertically, the two peaks on the horizontal array blend into one: the fingers shadow each other (Figure 2, center). After the alignment the ambiguity reappears (Figure 2, bottom).

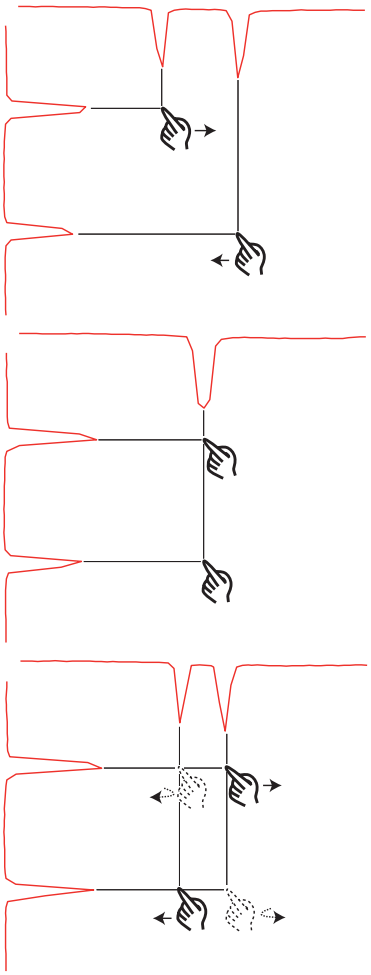

Figure 2. Shadowing when two fingers align vertically. Before (top), during (center) and after (bottom) the alignment. The ambiguity reappears after the vertical alignment: either the finger trajectories crossed (plain lines) or bounced back (dashed lines).

The tracking approach provides a solution assuming that hands have limited acceleration (i.e. they cannot change speed instantaneously), but have fast enough relative speed before the shadowing. In this case, it is valid to assume that the hands will not change direction during the shadowing, hence the "crossing" option can be chosen (Figure 2, bottom, plain lines). However, shadowing can occur with a low relative speed of the hands. In this case, the "crossing" and "bounce back" option (Figure 2, bottom, dashed lines) are both valid after the shadowing. In other words, the ambiguity reappears after shadowing of two fingers moving with low relative speed.

During informal experiments, we observe that shadowing occurs frequently when users perform multi-touch interactions. In these events, both the crossing and bounce back options are frequent. Contrary to the bootstrapping problem, where asking for cooperation appears to be acceptable for users, the shadowing problem should be handled by the system in a transparent manner: it would be very difficult for users either to consciously maintain a significant speed of motion of their fingers, or to avoid vertical and horizontal crossing of the fingers entirely.

In the next paragraph, we explain why a simple modeling of response peak does not allow for the implementation of a tracking algorithm that is robust enough to shadowing. This motivates a more precise model of the peaks that we will present in the next section.

Tracking with a simple peak model

The first step in tracking is to locate response peaks in both the horizontal and vertical 1D signals. The common way to do this, as provided in the DiamondTouch software, is to compute a threshold (based on the average noise level) and to group *adjacent antennas* which level is above the threshold. This is illustrated on Figure 3. A peak location is computed as the *centroid* of its antennas (i.e. the average of the antenna's positions weighted by their signal strength). The centroid results in peak positions having a better resolution than the number of antennas. However, during a shadowing event with fingers moving with low relative motion, the thresholding approach localize only one peak during several data frames. For example in the shadowing event represented in Figure 3, only one peak is visible during 7 data frames (frame #3 to #9).

A more robust model of peaks, with respect to shadowing, can be achieved by searching for local maxima. Touch locations are detected as maxima of the signal that satisfy some criteria, such as being above a threshold and passing some minimal amount of altitude change between its neighbors. However, in the event represented on Figure 3, this would only save frame #3 and would not prevent the two peaks to shadow each other during 6 frames (#4 to #9).

The duration of the shadowing has a strong effect on the robustness of the tracking. Tracking predicts the position of a peak in a new data frame based on its last known position and speed. In the absence of shadowing, this prediction is

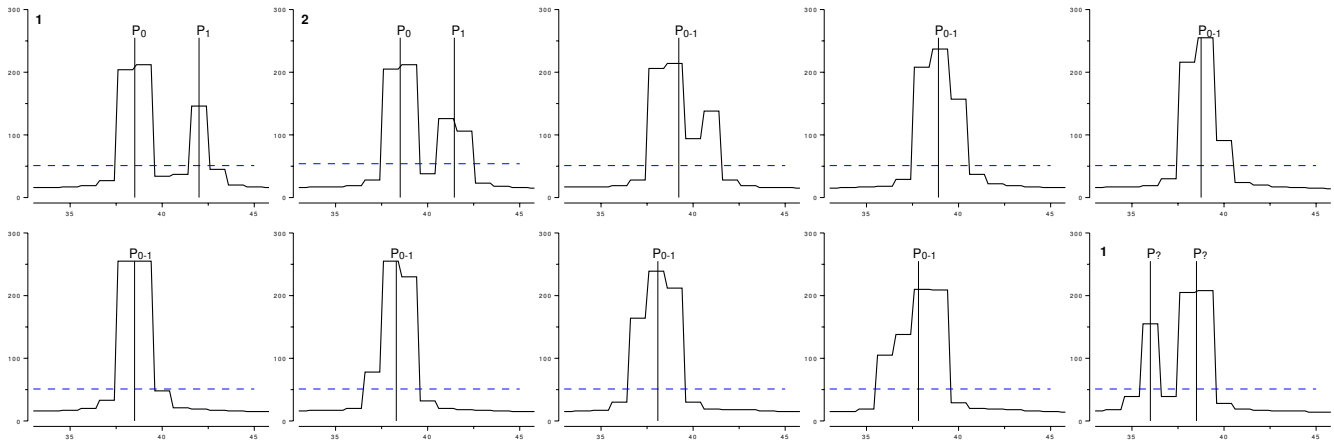

Figure 3. Shadowing occurring in 10 successive frames. Response on horizontal antennas #33 to #45 (black curve), dynamic threshold (dashed blue line), and computed peak locations (vertical lines). From frame #3 to #9 the two peaks shadow each other: the algorithm only computes one peak location.

then fit to the observed data which prevents the estimated peak location to drift from the true location. However, during a shadowing event, the position of a shadowed peak cannot be observed in the data and thus its estimation only relies on the prediction. The same is true for the estimated speed because it is computed as the difference between the current and the previous position of the peak. Hence, the incertitude of the prediction quickly increases as the last observation ages.

In order to minimize the number of data frames where the position of peaks cannot be observed during a shadowing event, we model the DiamondTouch antennas' outputs as a mixture of Gaussians.

MODELING RESPONSE AS A MIXTURE OF GAUSSIANS

When only one finger is in contact with the DiamondTouch, the shape of the peak in the signal is roughly similar to the shape of a Gaussian distribution function. When another finger comes close to the first one, the strength of the signal originating from the two fingers accumulate on each antennas and the overall output appears as the sum of the two peaks. This phenomenon is visible in Figure 3.

The model

We thus choose to model the signal S as a mixture of Gaussian: a sum of k weighted Gaussian functions M_j (also called *modes*):

$$S(i) = \sum_{j=0}^{k-1} M_j(i) \tag{1}$$

$$M_j(i) = s_j * N_{\mu_j,\sigma_j}(i) \tag{2}$$

where $N_{\mu,\sigma}$ is the normal (Gaussian) distribution function centered on μ and of standard deviation σ. s_j is the weight of mode j. A mode's weight models the difference in strength between touch responses: a strong pressure on the DiamondTouch generates a peak with a greater surface than a small pressure.

A mixture of Gaussians is a common mathematical tool used in many domains. The parameters of the model (center, standard deviation and weight of each mode) are usually optimized using the *Expectation Maximization* (EM) algorithm [1]. EM being an iterative algorithm, an initial parameter vector of the model must be provided. In addition, the number of modes in the mixture must be defined a priori.

Initialization

In the case of the modeling of a DiamondTouch 1D output, the initialization of the model is straightforward: one mode is created for each *peak* detected by the simple thresholding algorithm (see "Tracking with a simple peak model"). At the creation of a mode, the center is set to the centroid of the thresholded peak, the standard deviation is set to half the width of the thresholded peak (i.e. half the number of adjacent antennas passing the threshold), and the weight is set to the surface of the peak (i.e. the sum of the values measured on its antennas). The choice of initial parameters is not critical as it is the duty of the EM algorithm to start from coarse estimates and fit an accurate model.

We add a "noise" mode that models the noise level on the antennas when there is no user contact, so that this residual signal does not influence the shape of actual "peak" modes. The noise mode is initialized with its center in the middle of the signal, a very large standard deviation of half the length of the signal, and a weight defined as what remains from the entire signal surface once the surface of all peak modes have been removed.

As a result, the prior knowledge of the thresholded peaks allows us to define an initial state of the Gaussian mixture model that is close to the optimal solution. Indeed, we measure that the EM algorithm converges quickly with an average of 6 iterations when using convergence thresholds of 0.01 antennas for centers, 0.05 antennas for standard deviations and 10 units of signal strength for the weights.

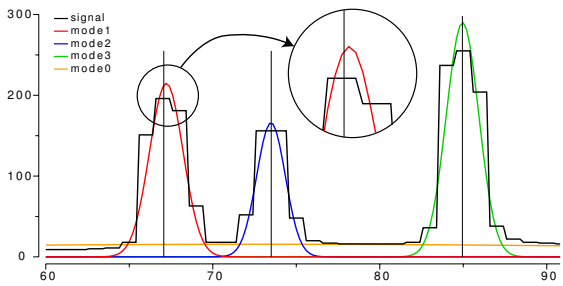

Figure 4. Signal (black curve) modeled by a mixture of Gaussians (colored Gaussian shaped curves).

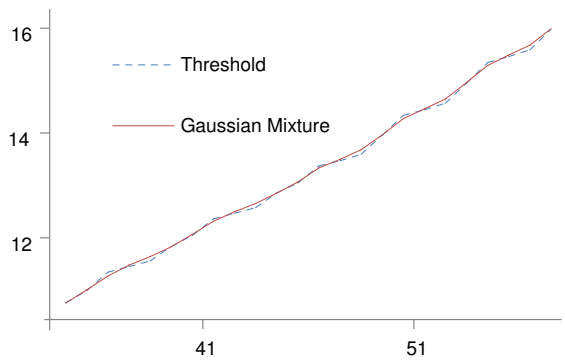

Figure 5. Horizontal position (y-axis) of a finger moving on the surface as estimated by the thresholded peaks (dashed blue line) and the Gaussian mixture (plain red line) across data frames (x-axis).

Results

A simple example of signal modeled by a mixture of Gaussians is shown in Figure 4. The orange mode, labelled mode0, is the "noise" mode. The three other modes model touch responses. The first benefit of the Gaussian mixture is its accuracy. This is illustrated on the leftmost peak in Figure 4: the position estimated by the threshold approach (vertical line) and the mixture (center of the Gaussian shape) do not match. In order to determine which is the most accurate, we conducted a simple experiment: we moved a finger continuously on a straight line on the surface, and we plotted both positions estimated by the thresholded peaks and the mixture. The result is shown on Figure 5. The curve from the mixture appears more regular than the one from the thresholded peak. Instantaneous speed was estimated at each data frame, except the first one, by the difference between the current position and the previous one. While the average of the instantaneous speed is equivalent using the two approaches, at 0.22 antennas/s, the standard deviation is 40% smaller using the mixture estimation (sttdev = 0.067) compared to thresholded peaks (stddev = 0.11). This improvement in the estimated position and speed contributes to better predictions in the tracking algorithm.

But the main benefit from the mixture model is shown on Figure 6: provided that the mixture is initialized with the correct number of modes, especially during a shadowing event, it is able to localize overlapping peaks from the accumulated signal. This is a significant improvement when compared to Figure 3: there is no frame where peaks position cannot be recovered. This requires, however, that the simple initialization scheme presented in section "Initialization" be refined to handle the case where one thresholded peak can be modeled by more than one mode. This is taken care by our tracking algorithm.

THE TRACKING ALGORITHM

Our tracking algorithm begins by tracking peaks in 1D in the signal of the two arrays (vertical and horizontal) of the DiamondTouch. It then processes the associations of two 1D peaks in order to keep track of the 2D touch locations.

1D trackers

Each 1D tracker (one for each array) manages a set of *tracked peaks* characterized by their center, standard deviation, weight, and speed. The first three parameters are used to initialize the modes when running the EM algorithm. The algorithm of the 1D tracker is composed of the following steps:

- *Predictions update*. The prediction of the location in the current frame of every tracked peak ($\hat{p}_t$) is estimated from the position (p) and speed (s) at the previous frame: $\hat{p}_t = p_{t-1} + s_{t-1} * \Delta t$.
- *Thresholded peaks computation*. The simple peak approach (see "Tracking with a simple peak model") is used to locate *thresholded peaks* in the signal. This step outputs a list of thresholded peaks described by their center, width, and surface.
- *Thresholded peaks association*. Every tracked peak is associated, using its predicted position $\hat{p}_t$, with the closest thresholded peak. The output may contain thresholded peaks associated with 0, one, or more tracked peaks. Tracked peaks may not be associated if their distance to the closest thresholded peak is greater than a threshold (defined by the maximum reachable speed).
- *Association processing*. Unassociated tracked peak are marked as disappearing and will be processed by the 2D tracker.

  Thresholded peaks being associated with a single tracked peak are simply used to update the estimated position and weight of the tracked peak.

  If more than one tracked peak is associated with a single thresholded peak, this indicates a shadowing. The prediction of the tracked peaks are not updated from the location of the single thresholded peak as it represents some kind of centroid of the shadowed peaks. The update is left to the mixture of Gaussians (next step).

  Thresholded peaks that are not associated denote a new touch event. A new tracked peak is created using the parameters of the thresholded peak: its center, half its width for the standard deviation, and its surface for the weight. New tracked peak speeds are set to 0.
- *Mixture modes computation*. The EM algorithm is ran with one mode per tracked peaks, plus the noise mode. Peak modes are initialized with the tracked peak parameters.

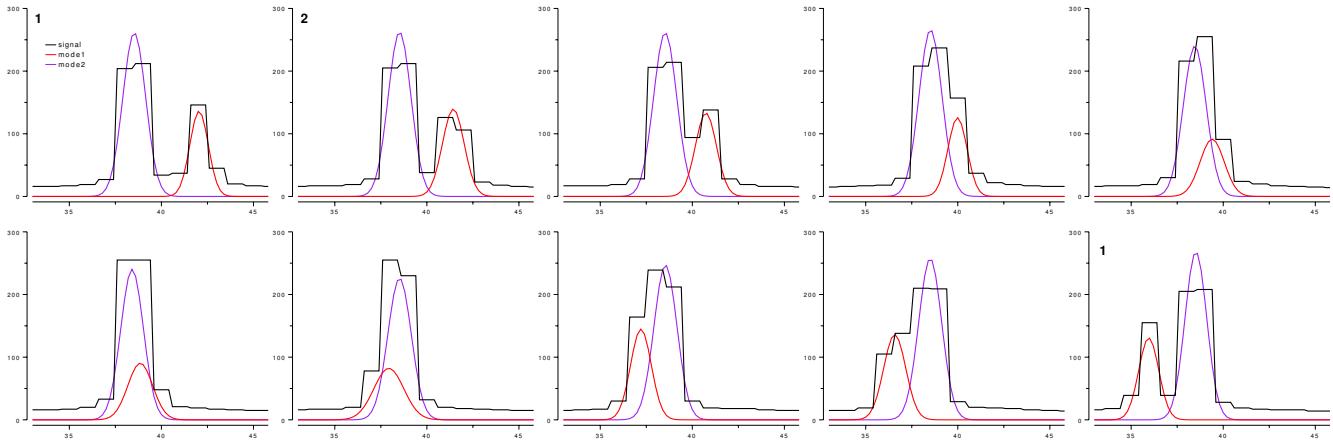

Figure 6. The same shadowing event as Figure 3 modeled as a mixture of Gaussians. The Gaussian model correctly localize the peaks even during shadowing.

- *Mixture modes association*. The combination of all possible couples of mode and tracked peak is computed. The optimal combination (the one minimizing the sum of distances between the mode and tracked peak centers) is then used.

  Tracked peak centers are updated with the center of their associated mode, unless they are involved in a shadowing event with the mode centers distant of less than one antenna. In this case, the signal shape is mostly symmetrical (as shown on Figure 6, frame #6) and does not allow for a correct recovery of the peak locations. The tracked peaks are thus left on their predictions (see "Prediction update" above).

- *Speed estimation*. The speed of every tracked peaks is estimated as the difference between the new center position and the one at the previous frame multiplied by the data frame update frequency.

2D tracker

The 2D tracker processes the tracked peaks of the two 1D trackers in order to associate them into a 2D *touch locations*. The output of the 2D tracker is a stream of events having one of the three types: Appear, Motion, or Disappear.

- When a single tracked peak on each array is available as not being associated to any touch location, a new touch location is created, and an *Appear* event is generated. The two tracked peaks are then marked as associated with the new touch location.

- When the 1D trackers update the position of at least one of the two tracked peaks associated with a touch location, then the 2D location of the touch is updated and a *Motion* event is generated.

- If one of the two tracked peaks associated with a touch location is marked as disappearing, then the touch is deleted, a *Disappear* event is generated, and the other tracked peak is returned to the set of unassociated tracked peaks of its 1D tracker.

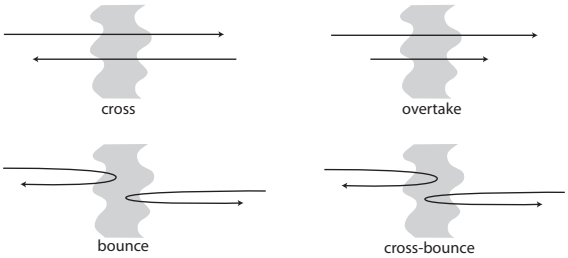

Figure 7. The 4 kinds of gesture in the evaluation corpus. The grey cloud represents the area of shadowing.

EVALUATION

Formal evaluation

In a formal evaluation, we recorded a corpus of DiamondTouch outputs while performing a set of two finger gestures that all exhibited some shadowing. Four kinds of gesture were used (Figure 7). In a *cross* gesture, the trajectories of the two fingers cross while moving in opposite directions. In an *overtake*, the trajectories cross while moving in the same direction: one finger is moving faster than the other one. In a *bounce*, the trajectories come close to each other and then move away. In a *cross-bounce*, a cross gesture occur, followed by a bounce gesture while the two fingers are still close to each others. For each kind of gesture and each of three finger motion speeds (fast, medium, slow), 20 trajectories were recorded. Hence $4 * 3 * 20 = 240$ trajectories where recorded in total. The corpus was batch-processed by our tracking algorithm with and without Gaussian mixture modeling activated. A failure was recorded when the tracking algorithm mis-interpreted a gesture or when it lost track of one of the finger, even if it was recovered in a subsequent data frame. It should be noted that these gestures were artificially chosen as being difficult cases to process: they are not representative of the most frequent trajectories observed in normal usage.

Table 1 summarizes the success rates. The Gaussian mixture modeling clearly improves the robustness of the track-

Table 1. Success rates

Kind	Speed	Without Gaussians	With Gaussians
cross	fast	100%	100%
	medium	100%	100%
	slow	45%	75%
overtake	fast	20%	90%
	medium	90%	100%
	slow	40%	65%
bounce	fast	65%	80%
	medium	65%	100%
	slow	75%	90%
cross-bounce	fast	40%	60%
	medium	25%	65%
	slow	15%	60%

ing. This is particularly visible in the most difficult case: the success rate is 4 times higher on cross-bounce at slow speed.

In order to get a feeling of the usability of the system in real-world usage, we conducted an informal evaluation with external participants.

Informal evaluation

We tested our system on a real application: the well-known photo shuffler. We asked a group of participants to spatially organize a set of pictures to their liking. We observed different two finger gestures such as the simultaneous translation of two pictures with two fingers from both hands; or the simultaneous rotation and scaling of pictures with two fingers of the same hand or two fingers from both hands. Some participants even rotated and scaled one picture with two fingers of one hand, while translating another picture with one finger of the other hand (i.e. a 3 finger action). Participants were told that if they touched the surface with two fingers at the same time, then the system may exhibit incoherent behavior, as when touching with a second finger aligned horizontally or vertically with the first.

Participant quickly learned not to touch the surface with two fingers at the same time. This was helped by the fact that the delay between two contacts can be very small (in the order to 2 data frames = $50ms$). The non-alignment constraint was more problematic in general as it took longer for participant to learn. However, after a short training period (a few minutes), participants were able to cooperate with the system so as to avoid the bootstrapping problems. They were then only confronted to tracking failures when their fingers exhibited shadowing at slow speed. When such a failure occurred, the system generated an incoherent output, such as a picture rotating in the wrong direction. Participants quickly noticed the incoherent behavior and stopped their gesture by lifting their fingers. They then restarted a new gesture and carried on with their intended action. All in all, there were few tracking failure and participants did not seem to be much disturbed in their task.

CONCLUSION AND FUTURE WORK

Overcoming the single touch per user limitation of the DiamondTouch would bring its qualities even closer to the ideal tabletop device. In this paper, we have presented a software-only approach that comes close to this goal. Through the precise modeling of the device's output using mixture of Gaussians, we were able to implement a robust touch tracking system that makes single user multi-touch viable on the DiamondTouch. The software is available both in source and binary forms as part of the GIL library[1].

We would like to investigate how this work can generalize to other kind of multi mono-dimentional sensor devices, such as the DViT system (see the "Related work" section). In this particular case, we need to evaluate if the mixture of Gaussians is an adequate model for the vector of occluded light.

Our system still requires a little bit of cooperation from its users when initiating their gestures. We envision two approaches to mitigate this limitation. In a software only solution, we plan to detect ambiguous situations and report both hypothesis to a higher level interaction layer. This layer would be in charge of presenting the two hypothesis to the user through graphical feedback. This would entice the user to simply lift one finger and touch again on one of the displayed hypothesis, in effect removing the ambiguity.

Alternatively, a completely autonomous solution could be achieved by augmenting the DiamondTouch with an overhead camera feeding a computer vision system. While implementing a robust vision system is hard to achieve when full detection and tracking in the whole video image is required, the requirements here are much more manageable: the vision system would simply have to assess the presence of a finger at the hypothesized locations and only when ambiguity appears in the DiamondTouch data. This would make possible the handling of all multi-touch situation on the DiamondTouch, but at the cost of some of the ease of setup.

ACKNOWLEDGEMENTS

The authors wish to thank Renaud Blanch for his help. This work was partially funded by the Agence Nationale de la Recherche for the DIGITABLE Project (ANR05-RNTL00503).

REFERENCES

1. J. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report TR-97-021, U.C.Berkeley, April 1998.

2. S. Chatty, A. Lemort, and S. Valès. Multiple input support in a model-based interaction framework. In *IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP)*, pages 179–186, 2007.

3. P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *ACM Symposium on User*

[1] http://gil.imag.fr/

Interface Software and Technology (UIST), pages 219–226. ACM, 2001.

4. J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 115–118. ACM, 2005.

5. E. Hornecker, P. Marshall, N. S. Dalton, and Y. Rogers. Collaboration and interference: awareness with mice or touch input. In *ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 167–176. ACM, 2008.

6. J. Letessier and F. Berard. Visual tracking of bare fingers for interactive surfaces. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 119–122, 2004.

7. P. Marshall, E. Hornecker, R. Morris, N. S. Dalton, and Y. Rogers. When the fingers do the talking: A study of group participation with varying constraints to a tabletop interface. In *IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP)*. IEEE Computer Society, 2008.

8. K. A. Mohamed, S. Haag, J. Peltason, F. Dal-Ri, and T. Ottmann. Disoriented pen-gestures for identifying users around the tabletop without cameras and motion sensors. In *IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP)*, pages 43–52. IEEE Computer Society, 2006.

9. K. Nakakoji, K. Jo, Y. Yamamoto, Y. Nishinaka, and M. Asada. Reproducing and re-experiencing the writing process in japanese calligraphy. In *IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP)*, volume 0, pages 75–78, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

10. A. M. Piper and J. D. Hollan. Supporting medical conversations between deaf and hearing individuals with tabletop displays. In *ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 147–156. ACM, 2008.

11. J. Rekimoto. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *ACM Conference on Human Factors in Computing Systems (CHI)*, pages 113–120. ACM, 2002.

12. J. Rick and Y. Rogers. From digiquilt to digitile: Adapting educational technology to a multi-touch table. In *IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP)*. IEEE Computer Society, 2008.

13. SmartTechnologies. Dvit digital vision touch technology white paper. Technical report, SMART Technologies, 2003.

14. P. Wellner. Interacting with paper on the digitaldesk. *Communications of the ACM (CACM)*, 1993.

15. A. D. Wilson. Touchlight: an imaging touch screen and display for gesture-based interaction. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 69–76. ACM, 2004.

16. M. Wu and R. Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 193–202. ACM, 2003.