

# A Hybrid HMM/DPA Adaptive Gesture Recognition Method

Stjepan Rajko<sup>1,3</sup> and Gang Qian<sup>2,3,\*</sup>

<sup>1</sup> Dept. of Computer Science and Engineering

<sup>2</sup> Dept. of Electrical Engineering

<sup>3</sup> Arts, Media and Engineering Program,  
Arizona State University, Tempe, AZ 85287, USA

**Abstract.** We present a hybrid classification method applicable to gesture recognition. The method combines elements of Hidden Markov Models (HMM) and various Dynamic Programming Alignment (DPA) methods, such as edit distance, sequence alignment, and dynamic time warping. As opposed to existing approaches which treat HMM and DPA as either competing or complementing methods, we provide a common framework which allows us to combine ideas from both HMM and DPA research. The combined approach takes on the robustness and effectiveness of HMMs and the simplicity of DPA approaches. We have implemented and successfully tested the proposed algorithm on various gesture data.

## 1 Introduction

Both Hidden Markov Model (HMM) and Dynamic Programming Alignment (DPA) approaches have been used extensively for gesture recognition<sup>1</sup>. Using HMMs, many good results have been achieved, for example with sign language [1, 2, 3]. While HMM-based methods usually prove accurate and robust, they require some effort in the proper training of the model (with notable exceptions such as [1]), and the implementation of the algorithms is not trivial. DPA approaches are much simpler, but only seem adequate for less challenging tasks, for example when the gesture vocabulary is small [4]. Although, there have been many attempts to improve DPA methods for gesture recognition, such as [5, 6].

DPA approaches have a long history, and an enormous body of research regarding various application areas and different optimization methods. For example, the Edit Distance problem [7], different kinds of DNA and protein sequence alignment [8, 9], and sound pattern recognition [10] can all be solved using DPAs. All of these areas offer common insights on various improvements, such as space-efficiency [11], or optimal implementations for parallel computers [12].

As HMMs are slowly pushing out DPAs in many application areas (including gesture recognition), improvements that have been developed in the context

---

\* This material is based upon work supported by the National Science Foundation under CISE-RI No.0403428.

<sup>1</sup> Please note that many HMM algorithms can be implemented using Dynamic Programming, and that by DPA we refer only to non-HMM algorithms.

of DPAs have become less applicable. To facilitate the reuse of ideas from both bodies of work, we attempt to provide a common framework for the two methods. We do so by relying on the robustness of HMM theory, but deriving underlying algorithms that are very reminiscent of DPA approaches. Applying this to gesture recognition proved to result in a method both accurate and simple.

## 2 Problem Formulation

Since our method is general, we will formulate the problem in general terms, but give specific examples from our experiments. For the initial part of the discussion, we will assume that we are only trying to recognize a single gesture in the object being observed. Extending this to multiple gestures will be explained later.

Let  $\mathbb{O}$  be a set of observations that we can witness by monitoring the object whose gestures we are trying to recognize. For example, in one of our experiments we monitor the direction of movement of a point in a 2D plane. We represent the direction by a point on the unit circle, so  $\mathbb{O} = \{(x, y) | x^2 + y^2 = 1\} \cup \{(0, 0)\}$ , where  $(0, 0)$  indicates that the point is not moving.

We are initially given an example of the gesture to be recognized, expressed as a sequence of observations  $M = M_1, M_2, \dots, M_{n-1}$  with each  $M_j \in \mathbb{O}$ . We are also given a continuous stream or real-time observations of the object we are monitoring, denoted by  $O_1, O_2, \dots$  with each  $O_i \in \mathbb{O}$ . Both of the observation sequences are expected to be collected at a consistent frame rate, i.e. typical time series data. We will refer to times at which they are recorded as time steps.

The goal is to find continuous parts of the real-time observation sequence  $O$  that correspond to the gesture given by  $M$ . That is, we want to separate parts of  $O$  that are similar to the example gesture given by  $M$  (instances of the gesture) from parts that are not. We do so by modeling the gesture by a Hidden Markov Model (HMM)  $\lambda = (S, a, \pi)$ , defined by the set of states  $S$ , the state transition probability distribution  $a$ , and the initial state probability  $\pi$ .

Each state in  $S = \{S_0, \dots, S_n\}$  represents a phase in the execution of the gesture. The initial state is governed by  $\pi = \{\pi_0, \dots, \pi_n\}$ , where  $\pi_j$  is the probability that the HMM will begin in state  $j$ . At each time step, the HMM changes its current state according to the state transition probability distribution  $a = \{a_{k,j}\}$ , with  $a_{k,j}$  being the probability of transitioning from state  $k$  to state  $j$ .

$S_1 \dots S_{n-1}$  correspond to the gesture example  $M = M_1 \dots M_{n-1}$ . Upon entering one of these states, the HMM emits an observation  $o \in \mathbb{O}$ , according to a probability distribution function initialized from  $M$ . In our experiments, the p.d.f. for a state  $S_j$ , which we denote implicitly by  $p(o|S_j)$ , is initialized using a Gaussian-based model with mean  $M_j$  and variance determined from the average distance between successive observations in  $M$ .

States  $S_0$  and  $S_n$  do not produce an observation (and for consistency we set  $p(o|S_0) = p(o|S_n) = 1$  for all  $o \in \mathbb{O}$ ), but are used to indicate the beginning and completion of the gesture. Hence, we enforce that the HMM begins in state 0 by setting  $\pi_0 = 1$  and  $\pi_j = 0$  for  $j \neq 0$ . We also assume that the gesture is executed by going through the states in a left to right fashion, which we model

using a simplified form of the transition probabilities. In particular, we choose three constants,  $a_{stay}$ ,  $a_{next}$ , and  $a_{skip}$ , which determine all state transition probabilities.  $a_{stay}$  gives the probability of staying in the same state from one time step to another, so a high value will allow gestures executed slower than the model to be recognized.  $a_{next}$  gives the probability of going to the next state, while the probability of going to the  $s^{th}$  next state is given by  $a_{skip}^{s-1} a_{next}$ . Intuitively,  $a_{skip}$  is the probability of skipping a state, so a high value will allow gestures that have parts of the model missing to be recognized.

In terms of  $a$ , we can write for all  $k$  and  $j < n$

$$a_{k,j} = \begin{cases} 0, & \text{if } j < k \\ a_{stay}, & \text{if } j = k \\ a_{skip}^{j-k-1} a_{next}, & \text{if } j > k \end{cases}, \tag{1}$$

with the values for  $a_{k,n}$  chosen so that  $\sum_{j=1}^n a_{k,j} = 1$ . In our experiments, we use  $a_{stay} + a_{next} \sum_{j=k+1}^{\infty} a_{skip}^{j-k-1} = 1$ , in which case  $a_{k,n} = a_{next} \sum_{j=n}^{\infty} a_{skip}^{j-k-1}$ .

### 3 The Algorithm

With  $\lambda = (S, a, \pi)$  defined, the question of relating a part of the observation sequence to the model of the gesture becomes the question of finding the most likely state sequence within  $\lambda$  to have produced it. In Section 3.1, we derive a modification of the Viterbi algorithm, which is commonly used to accomplish that task. In Section 3.2 we explain how we find the recognized gestures in the real-time observation sequence  $O_1, O_2, \dots$ . Section 3.3 shows how the recognized gestures are used to improve the model of the gesture, and Section 3.4 explains how we deal with the recognition of multiple gestures.

#### 3.1 A Modification of the Viterbi Algorithm

Given a particular sequence of  $v$  observations  $O_{u+1}, O_{u+2}, \dots, O_{u+v}$ , we seek to find the most likely state sequence  $(s_0 = S_0), s_1, \dots, s_v, (s_{v+1} = S_n)$  of the HMM  $\lambda$  that would have produced it. The Viterbi algorithm does so using a dynamic programming variable, which captures the probability of partial state sequence generating a partial observation sequence, which we write as

$$\delta_{i,j}^{(u)} = \max p((s_0 = S_0), s_1, \dots, (s_{v'} = S_j), O_{u+1}, \dots, (O_{u+v'} = O_i) | \lambda) \tag{2}$$

$\delta_{i,j}^{(u)}$  can be thought of as an instance of  $\lambda$  that started its execution just prior to producing observation  $O_{u+1}$ , with  $\delta_{i,j}^{(u)}$  representing the maximum probability of any partial state sequence starting with  $S_0$  and ending with  $S_j$  producing the observation sequence  $O_{u+1}, O_{u+2}, \dots, O_{u+v'} = O_i$ . It can be calculated as a table with initialization and recursive relationship

$$\delta_{u,j}^{(u)} = \begin{cases} 1, & \text{if } j = 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$\delta_{i,j}^{(u)} = \max_{k=1}^n \delta_{i-1,k}^{(u)} a_{k,j} p(O_i | S_j), \text{ for } i > u. \tag{4}$$

Using (1) and some simplification, we can separate (4) into

$$\delta_{i,j}^{(u)} = p(O_i|S_j) \max \left\{ \begin{array}{l} a_{stay} \delta_{i-1,j}^{(u)} \\ a_{next} \max_{k=1}^{j-1} \delta_{i-1,k}^{(u)} a_{skip}^{j-k-1} \end{array} \right. \quad (5)$$

However, we have no knowledge of when an intended instance of the gesture starts in the observation sequence  $O = O_1, O_2, \dots$ , and calculating the  $\delta^{(u)}$  table for all  $u$  is prohibitive. To that end we define  $\delta_{i,j} = \max \delta_{i,j}^{(u)}$ , which corresponds to choosing the optimal starting point ( $u$  value) at each point of the table, i.e. if  $\delta_{i,j} = \delta_{i,j}^{(u)}$  then observations  $O_{u+1} \dots O_i$  are best explained by the HMM that started its execution just prior to  $O_{u+1}$  being observed. Conviniently, it is easy to show that  $\delta_{i,j}$  can be calculated by compromising equations (3) and (4) with the superscript  $^{(u)}$  dropped:

$$\delta_{0,j} = \begin{cases} 1, & \text{if } j = 0 \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

$$\delta_{i,j} = \begin{cases} \max_{k=1}^n \delta_{i-1,k} a_{k,j} p(O_i|S_j), & \text{for } i > 0 \text{ and } j > 0 \\ 0, & \text{for } i > 0 \text{ and } j = 0 \end{cases} \quad (7)$$

We now define  $\gamma_{i,j} = \max_{k=1}^{j-1} \delta_{i,k} a_{skip}^{j-k-1}$ , and note that

$$\gamma_{i,j} = \max \left\{ \begin{array}{l} \gamma_{i,j-1} a_{skip} \\ \delta_{i,j} \end{array} \right. \quad (8)$$

Hence,  $\delta$  values for row  $i$  can now be calculated by first computing  $\gamma$  values for row  $i - 1$ , and then using

$$\delta_{i,j} = p(O_i|S_j) \max \left\{ \begin{array}{l} a_{stay} \delta_{i-1,j} \\ a_{next} \gamma_{i-1,j-1} \end{array} \right. \quad (9)$$

Instead of calculating  $\delta$  we set  $\alpha_{i,j} = \log \delta_{i,j}$ , and work with

$$\alpha_{i,j} = \max_{k=1}^n \{ \alpha_{i-1,k} + \log a_{k,j} + \log p(O_i|S_j) \} \quad (10)$$

This gives us

$$\alpha_{i,0} = 0 \quad (11)$$

$$\alpha_{i,j} = \log p(O_i|S_j) + \max \left\{ \begin{array}{l} \log a_{stay} + \alpha_{i-1,j} \\ \log a_{next} + \beta_{i-1,j-1} \end{array} \right., \text{ for } j > 0 \quad (12)$$

$$\beta_{i,j} = \max \left\{ \begin{array}{l} \beta_{i,j-1} + \log a_{skip} \\ \alpha_{i,j} \end{array} \right., \quad (13)$$

which is more efficient and numerically stable, and has similarities with certain forms of DNA alignments and local alignments in particular[8].

Because  $a_{k,n}$  is an exception to the rules given by (1),  $\alpha_{i,n}$  needs to be calculated according to (4). Hence, we need three passes for each row of the table. One is to calculate  $\alpha$  for all columns but the last, one to calculate  $\beta$  values, and for the last column of  $\alpha$ . Since the work for each cell is constant, the time complexity to calculate a row of  $\alpha$  and  $\beta$  is  $\Theta(n)$ .

### 3.2 Recognizing Gestures

The issues to address in relation to the gesture recognition are determining where a recognized gesture ended, where it started, and the most likely state sequence that generated the observations in between. To determine where a recognized gesture ends, we monitor the values in the last column of the  $\alpha$  table. Whenever we encounter a local maximum at some row  $i$ , i.e.  $\alpha_{i-1,n} \leq \alpha_{i,n} > \alpha_{i+1,n}$ , we examine the value  $\alpha_{i,n}$ . If it is large enough, it means that a complete gesture (ending with  $S_n$ ) has been executed with high enough probability.

To decide the threshold, consider the probability of the HMM generating a fictitious sequence of observations  $o$  such that  $\log p(o|S_j) \geq r$  for some constant  $r$  and all  $j = 1, \dots, n-1$ , with  $n_{skip}$  states getting skipped and  $n_{stay}$  times that the HMM stays in the same state from time step to time step. Then,

$$\alpha_{i,n} \geq n_{skip} \log a_{skip} + (n - n_{skip})(r + \log a_{next}) + n_{stay}(r + \log a_{stay}) \quad (14)$$

After selecting appropriate values for  $r$ ,  $a_{stay}$ ,  $a_{next}$ ,  $a_{skip}$ , different values for  $n_{stay}$ , and  $n_{skip}$  will give us the probabilities of different fictitious observation sequences, which can be used to determine a reasonable value for the threshold. In our experiments, we scale the probability distributions over  $\mathbb{O}$  so that for an observation  $o$  that is one half of a standard deviation away from the mean of the Gaussian p.d.f. of a state  $S_j$ ,  $\log p(o|S_j)$  is expected to be 0 (it can vary slightly depending on the particular p.d.f.), so we set  $r = 0$ . We would also like to allow the gesture to be executed somewhat slower than the example  $M$ , and not allow a significant portion of the gesture to be omitted, so we set  $a_{stay} = 0.75$  and  $a_{next} = 0.25$ , which gives  $a_{skip} = \frac{1}{11}$ . A borderline acceptable instance of the gesture would have  $n_{skip} = \frac{1}{11}n$ , and  $n_{stay} = 0.75n$ , yielding  $\alpha_{i,n} \geq \frac{1}{11}n \log \frac{1}{11} + \frac{10}{11}n \log 0.75 + 0.25n \log 0.25$ . Anything more probable than this will be accepted as a recognized instance of the gesture, and anything less probable will be rejected.

Once a local maximum in the rightmost column of  $\alpha$  exceeds the threshold, we employ a traceback technique to extract the correspondence between the observations and the HMM states. This is a simple modification of the traceback commonly used with the Viterbi algorithm. Basically, we maintain pointers along with the  $\alpha$  and  $\beta$  table calculations which tell us the progression of the states. In our version of the algorithm, the pointers are maintained as follows:

$$\beta_{i,j}^* = \begin{cases} j, & \text{if } \beta_{i,j} = \alpha_{i,j} \\ \beta_{i,j-1}^*, & \text{otherwise.} \end{cases} \quad (15)$$

$$\alpha_{i,j}^* = \begin{cases} j, & \text{if } \alpha_{i,j} = a_{stay}\alpha_{i-1,j} \\ \beta_{i-1,j-1}^*, & \text{otherwise.} \end{cases} \quad (16)$$

If the end of the gesture was detected at  $\alpha_{i,n}$ ,  $\alpha_{i,n}^*$  tells us the previous state,  $\alpha_{i-1,\alpha_{i,n}^*}$  the state before that, etc. When we reach  $S_0$ , that tells us where the gesture started. If the maximum length of a gesture instance is  $m$ , this requires us to keep  $m$  rows of the  $\alpha$  table in memory, i.e.  $\Theta(mn)$  space. However, the method of Hirschberg [11] (given in the context of the maximal common subsequence problem) can be adapted to reduce the space requirement to  $\Theta(m+n)$ .

### 3.3 Updating the Model

Once it has been detected that a gesture has been completed in the observation sequence, the traceback method tells us not only the most likely state sequence, but also what observation was generated by what state. This induces a mapping from the observations to the states of the model. In cases where the model was predicted to produce multiple observations from the same state, all of these observations are mapped to this state.

We can now update the model by incorporating the recently recognized gesture, which is similar to the approach taken by Lee and Xu[1]. We do so by updating each state which has an observation mapped to it. For each mapped observation we modify the p.d.f. related to the state to incorporate information given by the new observation. This can be done in many ways, for example by running a few iterations of the Expectation-Maximization algorithm with the new observation included. Here, we present a simpler approach as an example.

Suppose we represent the p.d.f. for each state by a mixture of Gaussians, with a given state  $S_j$  having  $g$  Gaussians, 1 through  $g$ , and each Gaussian  $i$  having mean  $\mu_k$ , variance  $\sigma_k^2$ , weight  $\gamma_k$ , and  $p(o|S_j) = \sum_{k=1}^g \gamma_k \mathcal{N}(o|\mu_k, \sigma_k)$ .

To update  $S_j$  given a new observation  $o$ , we use a simple probabilistic technique that modifies the  $\mu$ ,  $\sigma$ , and  $\gamma$  parameters. We first choose a Gaussian from the mixture whose mean and variance we will update. The probability of choosing Gaussian  $k$  is proportional to  $\gamma_k \mathcal{N}(o|\mu_k, \sigma_k)$ . If Gaussian  $i$  is chosen, we decide the new  $\mu_i$  and  $\sigma_i$  by randomly drawing  $R$  points from the old Gaussian and adding  $o$  to the mix. Hence,  $R$  indicates how flexible the model is - a high  $R$  results in slow changes in the model, while a low  $R$  modifies the model more quickly. Furthermore, for well behaved observation sets the expected result of the above probabilistic method can be evaluated or approximated analytically.

Finally, we can update the weights so that

$$\gamma'_k = \begin{cases} \frac{R+1}{R+\gamma_k} \gamma_k, & \text{if } k = i \\ \frac{R}{R+\gamma_k} \gamma_k, & \text{otherwise.} \end{cases} \quad (17)$$

### 3.4 Recognizing Multiple Gestures

Up to this point in our discussion, we have assumed that we are only trying to find instances of a single gesture in the observation sequence. To recognize multiple gestures, we simply run multiple instances of the single gesture algorithm.

Each instance of the algorithm is initialized by an example of its gesture, and at each time step, a new row of each of the  $\alpha$  tables is calculated using the newly recorded observation. The time complexity of the algorithm becomes  $\Theta(\sum n_i)$ , where  $n_i$  is the number of observations given in the example of gesture  $i$ .

Once a gesture is found, each instance of the algorithm reinitializes its  $\alpha$  table, which corresponds to enforcing that no two recognized gestures overlap.

## 4 Experimental Results

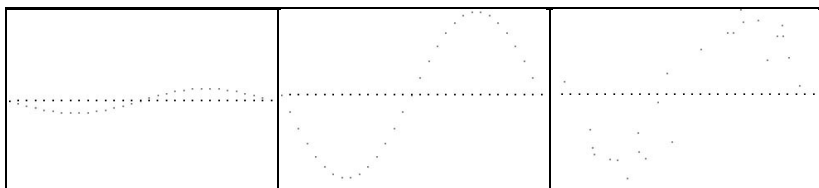
We have implemented the proposed algorithms into the Motion Analysis and Visualization Engine (MAVE, <http://randomaxis.info/research/mave/>), which is released under the GNU GPL. Our initial tests dealt with the recognition of gestures of a point in a 2D plane, with the observations given to the algorithm being the point position (scenario A,  $\mathbb{O} = \mathbb{R}^2$ ), velocity (scenario B, again  $\mathbb{O} = \mathbb{R}^2$ ), or the direction of movement (scenario C,  $\mathbb{O} = \{(x, y) | x^2 + y^2 = 1\} \cup \{(0, 0)\}$ ).

The scenarios were each tested in a task involving recognition of 10 gestures representing digits 0-9, with the results summarized as follows. Velocity as the observation (scenario B) worked poorly, probably due to even slight changes in the speed of the gesture execution having dramatic effects on the observations. Position (scenario A) worked well, but only after incorporating a grid into the user interface, allowing the user to place the gestures in a similar position as the example gesture. Angle (scenario C) worked the best. In scenarios A and C, nearly all expressed gestures were recognized, and no misclassifications occurred.

To illustrate the effects of model updating, we created some more detailed tests using scenario A and synthetic gestures of the form  $y = a \sin t, x = t$  for  $0 \leq t < 2\pi$  (see Figure 1). The gesture sequence to be recognized was generated by repeating the gesture  $y = a \sin t$  with noise. Before every ten trials of the noisy gesture, non-gesture data of the form  $y = 0, x = t$  was inserted. Also, we slowed down the model updating to see its effects over a longer period of time.

With  $a = 0.3$ , and very low noise (uniform distribution noise of  $\pm 0.05$  on both  $x$  and  $y$ ) in the first test, the gesture data and the non-gesture data were very similar (Figure 1, left). The first 9 times the non-gesture was observed, it was misclassified as an instance of the gesture. However, after the 9<sup>th</sup> time (after observing 90 highly precise true gestures) it was no longer misclassified, showing the ability of the model updating to discriminate between very similar gestures.

In the second test, we used  $a = 2.0$  (Figure 1, middle and right), and very high noise ( $\pm 0.75$ ). In this case, the gesture was barely recognizable, but the recognition improved with time. The number of times it was recognized in each set of ten gesture trials was (in order) 1, 0, 1, 9, 10, 10, 10, 10, ... However, after recognizing these 51 instances of highly noisy data, the model became so flexible that it misclassified the non-gesture straight line as the gesture, showing that the model updating can adapt even to high noise, but at a cost in misclassification.



**Fig. 1.** gestures  $y = 0.3 \sin t$  (left),  $y = 2 \sin t$  (middle), and recognized portions of  $y = 2 \sin t$  with large noise (right)

## 5 Conclusions and Future Work

We have presented a hybrid method applicable to gesture recognition, based on Hidden Markov Model theory but reminiscent of other Dynamic Programming Alignment methods. Our solution has both good accuracy (typical of HMM), and minimal training/initialization requirements (typical of DPA).

In addition to the 2D point results presented here, we have also successfully applied the algorithm to 3D motion capture data, allowing us to recognize full body gestures. We are currently working on adapting our algorithm to sign language data[2], which will allow us to compare our approach to other methods.

## References

1. Lee, C., Xu, Y.: Online, interactive learning of gestures for human/robot interfaces. In: 1996 IEEE International Conference on Robotics and Automation. Volume 4. (1996) 2982–2987
2. Vogler, C., Metaxas, D.: Handshapes and movements: Multiple-channel american sign language recognition. In: Lecture Notes in Computer Science. Volume 2915. (2004) 247–258
3. Hienz, H., Bauer, B., Kraiss, K.F.: Hmm-based continuous sign language recognition using stochastic grammars. In: GW '99: Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction, London, UK, Springer-Verlag (1999) 185–196
4. Corradini, A.: Dynamic time warping for off-line recognition of a small gesture vocabulary. In: Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'01), Washington, DC, USA, IEEE Computer Society (2001) 82
5. Keogh, E., Pazzani, M.: Derivative dynamic time warping. In: First SIAM International Conference on Data Mining, Chicago, IL, USA (2001)
6. Wu, H., Kido, R., Shioyama, T.: Improvement of continuous dynamic programming for human gesture recognition. In: International Conference on Pattern Recognition. Volume 02. (2000) 945–948
7. Masek, W.J., Paterson, M.S.: A faster algorithm for computing string edit distances. *Journal of Computer and System Sciences* **20** (1980) 18–31
8. Setubal, J., Meidanis, J.: Introduction to computational molecular biology. PWS Publishing Company, Boston, MA (1997)
9. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. *Journal of Molecular Biology* **147** (1981) 195–197
10. Kruskal, J.B., Liberman, M.: The symmetric time warping algorithm: From continuous to discrete. In: Time Warps, String Edits and Macromolecules: The Theory and Practice of String Comparison, Addison-Wesley (1983)
11. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. In: Communications of the ACM. Volume 18(6). (1975) 341–343
12. Rajko, S., Aluru, S.: Space and time optimal parallel sequence alignments. *IEEE Transactions on Parallel and Distributed Systems* **15** (2004) 1070–1081

Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF).