A $-Family Friendly Approach to Prototype Selection

Corey Pittman
University of Central Florida
4000 Central Florida Blvd
Orlando, FL 32816, USA
cpittman@knights.ucf.edu

Eugene M. Taranta II
University of Central Florida
4000 Central Florida Blvd
Orlando, FL 32816, USA
etaranta@gmail.com

Joseph J. LaViola Jr.
University of Central Florida
4000 Central Florida Blvd
Orlando, FL 32816, USA
jjl@cs.ucf.edu

ABSTRACT

We explore the benefits of intelligent prototype selection for $-family recognizers. Currently, the state of the art is to randomly select a subset of prototypes from a dataset without any processing. This results in reduced computation time for the recognizer, but also increases error rates. We propose applying optimization algorithms, specifically random mutation hill climb and a genetic algorithm, to search for reduced sets of prototypes that minimize recognition error. After an evaluation, we found that error rates could be reduced compared to random selection and rapidly approached the baseline accuracies for a number of different $-family recognizers.

Author Keywords

Classifier; Gesture recognition; Rapid prototyping; User interfaces; $-family; Prototype selection

ACM Classification Keywords

H.5.2 Information interfaces and presentation: User interfaces, input devices and strategies; I.5.5 Pattern recognition: Implementation, interactive systems

INTRODUCTION

$-family recognizers, such as $1 [17], $N [2], 1¢ [6], and $P [15], are 2D gesture recognizers designed to be accessible to all developers. They are suitable for rapid prototyping, have low coding overhead, rely only on simple geometry and nearest neighbor prototype matching, utilize intuitive internal representations of gestures, are easy to debug, and achieve high accuracy with as little as one prototype per gesture [15, 17]. By possessing these characteristics, it is not surprising that $-family recognizers have gained significant popularity. However, developers who employ these recognizers may also be interested in $-family like methods that address related topics in gesture recognition such as rejection criteria for malformed gestures, synthetic data generation, and prototype selection. In this paper, we begin to tackle the latter.

IUI'16, March 7–March 10, 2016, Sonoma, CA, USA.
 ISBN 978-1-4503-4137-0/16/03...$15.00.
DOI: http://dx.doi.org/10.1145/2856767.2856808

It is easy to acquire a large dataset. One can collect numerous gesture samples from friends, family, coworkers, and volunteers; record samples online and return them to a local repository; select gestures from publicly available datasets such as [3, 9, 16, 17]; or generate countless synthetic samples from an online service [7]. However, with $-family recognizers that are 1-nearest neighbors template matching algorithms, each new training sample incurs an additional penalty in recognition time and storage space, both of which are often restricted or must be minimized as much as possible. It is therefore necessary to winnow the dataset down to a small subset of samples that exhibit high discriminatory power—that best separate the gesture classes. This prototype (or instance) selection problem is well studied [4], but most solutions are complex (requiring expert knowledge), are slow, or have high coding overhead, and are consequently juxtaposed to $-family ideology in one way or another. Alternative recognizers, like Rubine's linear classifier [12], have also been used for 2D gesture recognition but feature more complex mathematical structures and are therefore less approachable.

In a $-family friendly context, a promising set of solutions for prototype selection are stochastic methods such as genetic algorithms and hill climbing [10, 11, 13]. These methods have been shown to achieve high accuracy, but involve only random template selection and recognizer evaluation. As a result, there is little bookkeeping and little coding overhead, and therefore can be added to any project with only minimum effort.

PROTOTYPE SELECTION

In this work, we propose two potential selection methods to be applied to $-family recognizers: Genetic Algorithms and Random Mutation Hill Climb. Both stochastically search for optimal solutions using online repeated cross-validation of the candidate solutions.

Genetic Algorithm

Genetic algorithms test the fitness of a population consisting of multiple candidate solutions for a number of generations. At the end of each generation, children are created from the higher fitness candidates (parents) from the population. Genetic algorithms leverage mutation and crossover operators to facilitate the exploration of new areas within the search space.

In the context of prototype selection, each candidate solution is a subset of the complete dataset. During initialization, a random collection of prototypes is selected to populate each of the candidate solutions and a recognizer is generated from

those prototypes. To evaluate the fitness of a candidate, a random selection of prototypes is recognized, and the resultant accuracy is the fitness of the candidate. This operation is carried out for each of the candidates in the population. Next, the highest fitness candidates are selected to be carried over into the next generation, while the remaining candidates are replaced with children generated by mutating (changing one of the selected prototypes of a candidate) and crossing over (randomly selecting prototypes from two different candidates) candidates from the current generation. This new generation then repeats the processes of the previous generations until either the maximum number of generations pass or a solution that can correctly recognize all the prototypes during fitness evaluation is found. In our implementation, we used single point mutation and uniform crossover as our genetic operators. We also used elitism to carry over the top 50% of each generation. For the fitness evaluation, each candidate solution was evaluated by attempting to recognize $32k$ prototypes from the dataset, where k is the total number of prototypes in each candidate solution. The fitness is the accuracy of this result.

Random Mutation Hill Climb

Of all the methods we considered, Skalak's [13] approach to prototype selection using *random mutation hill climb* (RMHC) is perhaps the most appropriate technique with respect to coding overhead, complexity, understandability, speed, and accuracy. In his initial formulation, prototypes are represented as integers that index samples in a training set, and the selection is encoded into a single bit string. The bit string is randomly mutated one bit at a time. On each iteration, the fitness of the encoded prototype bit string is evaluated utilizing a 1-nearest neighbor classifier. If the fitness of the mutated string is greater than all previously identified solutions, the mutated string becomes the new best (fittest) solution. This process terminates after a predetermined, but low, number of iterations. Our approach is identical except that we store an array of integers that index samples in a training set, and we randomly mutate whole integers rather than individual bits in order to reduce coding complexity. However, there are also a number of optimizations that we employ to speedup the process.

There is no sense in considering a solution in which not every class is represented. When the target prototype count is low, for example, this situation is easy to encounter, and significantly many more mutations are required to achieve high accuracy as compared to the following alternative. Instead, we ensure that there is at least one prototype per gesture when performing a mutation. For each class we keep a counter of selected prototypes that represent the class. When a prototype is to be mutated, if its associated class count is one, we randomly select another prototype from within the same class. Otherwise, there is no restriction. Finally, if the fittest solution wins its second mutation round and has classified a sufficient number of consecutive samples correctly, the process terminates (currently this is 64 times the class count).

We operate under a similar assumption as Skalak: that $-family recognizers are typically employed in problem domains where good prototypes appear in dense regions of the sample space. We also assume that most, if not all, gestures reside in a *sloppiness space* [5], so that rushed and sloppy instances are still distinguishable. Using these assumptions, instead of evaluating a candidate solution over the entire training dataset, we can randomly sample from the dataset. That is, samples are drawn with replacement and evaluated against the fittest solution and candidate solution, until either one solution clearly out performs the other, or a predetermine number of iterations pass (we use 256 times the class count in our evaluation). Also, to ensure good test coverage, in each iteration we test one sample from each class. Since each result is a Bernoulli trial (either a sample is classified correctly or not), the sequence of all results is a Binomial distribution. Therefore we can use an exact Binomial test to compare the solutions. The minimum variance unbiased estimator of the success rate is given by:

$$\hat{p} = \frac{x}{n}, \tag{1}$$

where x is the number of successes in a set of n Bernoulli trails. Let $\hat{p}_0$ be the success rate estimate for the fittest solution and $\hat{p}_c$ the estimate for a new candidate solution. As solutions are iteratively evaluated, we conduct a one-tailed hypothesis test to eliminate any candidate solutions that are unlikely to improve accuracy:

$$\begin{aligned} H_0 &: \hat{p}_c = \hat{p}_0 \\ H_1 &: \hat{p}_c < \hat{p}_0, \end{aligned} \tag{2}$$

RANDOM-MUTATION-HILL-CLIMB ($Samples$, k)

for $i \leftarrow 0$ **to** k **do**
    $Fittest[i] \leftarrow$ RANDOM-SAMPLE($Samples$)
$FittestRecognizer \leftarrow$ GENERATE-RECOGNIZER($Fittest$)
$count \leftarrow 0$
$max \leftarrow 32 * k$
$TestMutations \leftarrow 256$
while $count < max$ **do**
    $Alternate \leftarrow Fittest$
    MUTATE-ONE($Samples$,$Alternate$)
    $AlternateRecognizer \leftarrow$ GENERATE-RECOGNIZER($Alternate$)
    $c1, c2, n \leftarrow 0, 0, 0$
    for $i \leftarrow 0$ **to** $TestMutations$ **do**
        for $j \leftarrow 0$ **to** *Gesture-Type-Count(*$Samples$*)* **do**
            $S \leftarrow$ RANDOM-SAMPLE-OF-TYPE($Samples$,j)
            $c1 \leftarrow c1+$IS-RECOGNIZED($FittestRecognizer$, S)
            $c2 \leftarrow c2+$IS-RECOGNIZED($AlternateRecognizer$, S)
        $k \leftarrow$ FIND-CRITICAL-VALUE($c1$, $c2$, n)
        if $c2 < k$ **then**
            $break$
    if $c2 > c1$ **then**
        $Fittest \leftarrow Alternate$
        $FittestRecognizer \leftarrow AlternateRecognizer$
return $Fittest$

FIND-CRITICAL-VALUE ($correct1$, $correct2$, n)

$\alpha, pf, comb \leftarrow 0.8, 1.0, 1.0$
$psuccess \leftarrow correct1/n$
$pfail \leftarrow 1.0 - psuccess$
$ps \leftarrow psuccess^n$
for $k \leftarrow n$ **to** 0 ***and*** $\alpha > 0$ **do**
    $\alpha \leftarrow \alpha - comb * ps * pf$
    $comb \leftarrow (comb * k)/(n - k + 1)$
    $ps \leftarrow ps/psuccess$
    $pf \leftarrow pf * pfail$
return k

where the null hypothesis is that candidate solution is approximately as good as fittest solution. Note, however, that the usual normal approximation to the binomial distribution is inappropriate because the success rate is often too high. For this reason we must conduct an exact test and manually find the critical value of the rejection region.

The cumulative distribution function (CDF) of the binomial distribution is as follows:

$$P(X \leq k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}, \tag{3}$$

where random variable X is the number of successes and p, again, is the probability of success. Given a level of significance, such as $\alpha = .05$, it is actually more efficient to solve the inverse problem: to find the critical value for the upper tail, $1 - \alpha$, and calculate the summation backward from n down to k for $P(X > k)$. Why? First note that Equation 3 contains a binomial coefficient, which can be solved incrementally as the CDF is being calculated by using the following relation:

$$\binom{n}{k-1} = \binom{n}{k} \frac{k}{n-k+1}, \tag{4}$$

where k represents the number of successes in n trials, and the initial condition is $\binom{n}{k=n} = 1$. Second, note that because the success rate is high, the k containing $\alpha = .05$ of the lower tail (or equivalently the k containing $\alpha = .95$ of the upper tail) is close to n. For example, with $n = 1500$ and $p = .95$, then 5% of the distribution falls under $k = 1439$.

EVALUATION

We ran a large scale evaluation of the proposed selection techniques and their effect on error rates. We compared to the baseline, using all available prototypes, and the current trend of random selection for reduction. In this way, we could determine if there was an improvement over random selection and how close the results came to the best possible result. We chose to evaluate the proposed selection methods on three different datasets, with details shown in Table 3.

Recognizers

We used a number of $-family recognizers to insure that the results were consistent regardless of which recognizers were used:

Protractor. Protractor [8] is a speed-optimized version of the traditional $1 recognizer. Protractor aligns the candidate gesture and dataset prototype to minimize the distance between the two unistroke and uses the inverse minimum cosine distance between the samples.

$N-Protractor. $N-Protractor [3] is a speed-optimized version of the $N recognizer which leverages the optimizations proposed in Protractor to decrease the overall computation time of $N while maintaining similar error rates.

Penny Pincher. The Penny Pincher recognizer [14] is a reduced complexity recognizer that emphasizes speed and simplicity while maintaining high recognition rates. Penny Pincher reduces the matching process to addition and multiplication, allowing for significantly shorter matching times.

	Penny Pincher								
	$1-GDS			SIGN			MMG		
k	GA	RM	Rand	GA	RM	Rand	GA	RM	Rand
1	2.0	1.5	8.8	4.8	4.1	14.1	23.4	22.6	37.8
2	0.9	0.6	4.8	2.4	1.5	8.8	11.3	8.7	25.5
3	0.5	0.5	3.4	2.0	1.1	6.0	8.6	4.5	20.2
4	0.4	0.5	2.5	1.8	0.9	4.5	6.3	3.2	16.2
5	0.2	0.5	2.3	1.3	0.9	4.2	5.2	2.4	14.0
n	<0.1	<0.1	<0.1	0.4	0.4	0.4	0.8	0.8	0.8
	Protractor								
	$1-GDS			SIGN			MMG		
k	GA	RM	Rand	GA	RM	Rand	GA	RM	Rand
1	2.3	1.6	12.0	16.0	16.0	32.7	23.4	23.7	35.6
2	1.0	0.8	6.7	14.5	13.2	27.8	9.6	6.9	23.3
3	0.8	0.7	4.8	13.8	13.1	24.8	7.0	4.3	18.5
4	0.6	0.5	3.6	13.9	12.2	23.4	5.8	3.6	15.5
5	0.6	0.5	3.1	13.9	12.7	23.1	5.1	3.2	13.4
n	0.2	0.2	0.2	13.5	13.5	13.5	3.0	3.0	3.0
	$N-Protractor								
	$1-GDS			SIGN			MMG		
k	GA	RM	Rand	GA	RM	Rand	GA	RM	Rand
1	5.0	4.2	14.7	15.9	16.2	31.1	12.0	12.0	20.4
2	1.3	0.9	7.6	12.6	11.1	24.3	5.6	4.1	12.0
3	0.9	0.5	5.5	11.7	10.7	20.8	3.9	2.1	9.3
4	0.7	0.8	4.1	10.7	10.1	19.0	3.5	1.7	7.4
5	0.6	0.7	3.7	11.1	10.6	18.4	2.7	1.7	6.1
n	0.1	0.1	0.1	9.4	9.4	9.4	0.5	0.5	0.5

Table 1: Absolute error rates (in %) for all datasets, selection modes, and recognizers for template counts $k = [1, 5]$ and baseline ($k = n$) using remaining samples as templates after removing the candidates.

	$1-GDS		SIGN		MMG	
Recognizer	Mem	Speed	Mem	Speed	Mem	Speed
Penny Pincher	98.3	95.7	99.7	99.5	97.5	95.2
Protractor	98.3	97.7	99.7	99.7	97.5	96.8
$N-Protractor	98.3	97.4	99.7	99.6	97.5	97.7

Table 2: Percent reduction in memory consumption and runtime for $k = 5$ compared to baseline. Note that metadata overhead is excluded from the memory consumption calculation.

Procedure

In order to determine the effect of the reduced datasets on the error rates of the $-family recognizers, we randomly generated a number of tests for each combination of recognizer, dataset, and selection mode. The evaluations were user independent, so samples for each gesture type from each user were combined together. A prototype was selected from each gesture class within the selected dataset and was designated a candidate to be matched. The candidate was matched against the best reduced dataset generated using one of the selection methods. The candidate was not included in the gesture selection process for that particular test. Error rates for this cross-validation were recorded for $k = [1, 5]$, where $k * T$ is the number of prototypes to which the dataset should be reduced, and T is the number of distinct classes in the complete dataset. The variance of the resulting error rates were minimized by running tests with each configuration 500 times.

Name	Ref	Multistroke	Gestures	Participants	Total Samples
$1-GDS	[17]	No	16	10	4800
SIGN	[1]	No	17	20	33154
MMG	[3]	Yes	16	20	3200

Table 3: Datasets used to evaluate selection methods. Additional details can be found in the associated references.

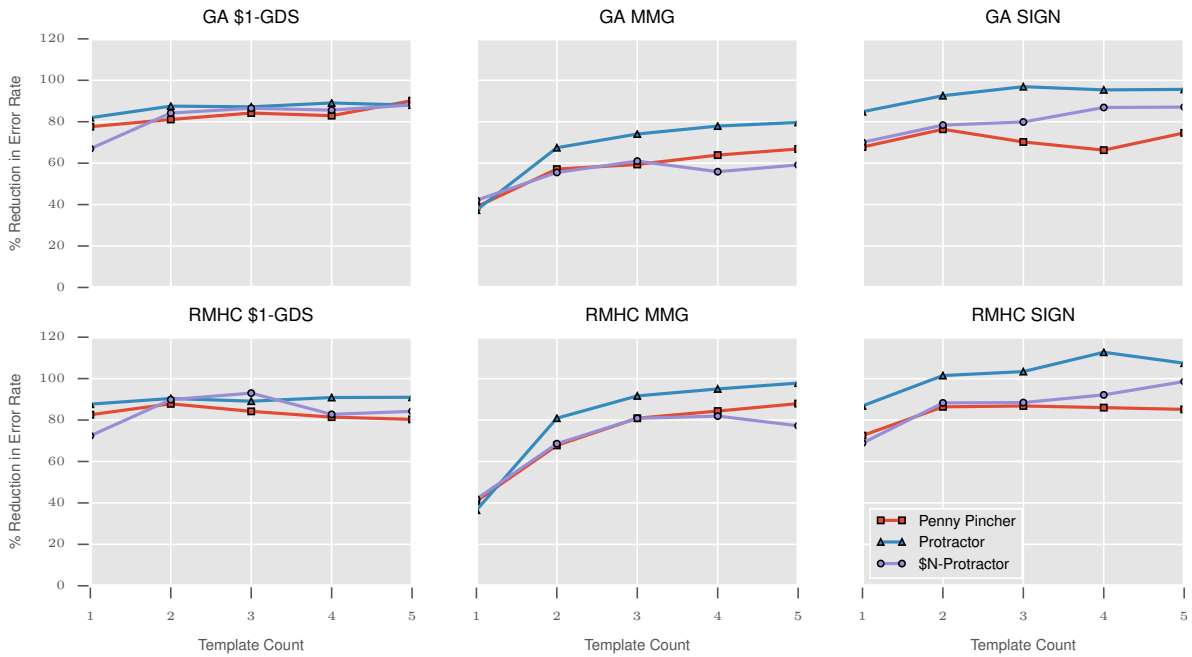

Figure 1: Percent reduction in error rates relative to random selection error rates and baseline for different sample counts per gesture class for each dataset. The baseline is represented by 100% reduction in error rate and random selection is represented by 0%; that is, charts were generated using the equation $100 * (random - method)/(random - baseline)$. In all cases, RMHC and GA were significantly closer to baseline accuracy than random selection with substantial reductions in error rate.

Results

Results for the evaluation are shown in Table 1. For visualization purposes, we elected to report the percentage decrease in error rate in lieu of the absolute error rate because we are not comparing recognizers. We wanted to see how close to baseline error rates the selection algorithms were able to reach. These results are presented in Figure 1. Each of the recognizers improved dramatically over random selection in all configurations. We compared the RMHC and GA error distributions to random selection for each dataset, recognizer and level of k, which resulted in 90 pairwise comparisons. Using Welch's unequal variances t-test with a Bonferroni correction, all results were significant ($p < .05/90$). Error rates were reduced by over 70% in all cases for $1-GDS, 35% for SIGN, and 25% for MMG. Table 2 also shows the reduction in time and space efficiency for $k = 5$. It is clear that reducing the size of the dataset reduces the computational load of the recognition process while still reaching similar recognition rates to baseline, according to Table 1.

DISCUSSION AND CONCLUSIONS

The results of the experiment point toward a clear benefit in using a supervised selection algorithm to reduce the number of prototypes while still maintaining only slightly worse error rates than when using the entire dataset. This idea is compounded in situations when there is a temporal constraint on recognition. Each configuration rapidly approached the baseline accuracy of the dataset for that particular recognizer.

Also notable is the fact that RMHC and GA performed similarly. Both methods are heavily influenced by the mutation genetic operator. Mutation is the primary method of exploration in GA with crossover being used to consolidate good candidates into potentially higher fitness individuals. RMHC is able to reach similarly optimal prototype subsets without the use of crossover or multiple candidates in each iteration. For this reason, we recommend practitioners use RMHC in lieu of GA. We have provided pseudocode for RMHC.

There are almost no downsides to carrying out this calculation, save the brief additional implementation time and time to carry out the preprocessing and selection, which is negligible for RMHC. However, we found that $N-Protractor was slow to carry out selection, particularly when working with a large, multistroke dataset (MMG). One possible way to solve this is to use a faster, more efficient recognizer like Penny Pincher during the selection process and then use the resultant subset of strokes to build a different $-family recognizer.

We proposed two alternative techniques for prototype selection that intelligently select prototypes that will best maintain the low error rates of the recognizer. Previous research [10] has found that there are methods to improve the performance of GA relative to RMHC. However, both techniques perform better than random selection in all cases tested here, and RMHC is simpler to code with similar execution time.

ACKNOWLEDGEMENTS

This work is supported in part by NSF CAREER award IIS-0845921.

REFERENCES

1. Almaksour, A., Anquetil, E., Quiniou, S., and Cheriet, M. Personalizable pen-based interface using lifelong learning. In *2010 International Conference on Frontiers in Handwriting Recognition (ICFHR)* (Nov 2010), 188–193.
2. Anthony, L., and Wobbrock, J. O. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010*, GI '10, Canadian Information Processing Society (Toronto, Ont., Canada, Canada, 2010), 245–252.
3. Anthony, L., and Wobbrock, J. O. $n-protractor: A fast and accurate multistroke recognizer. In *Proceedings of Graphics Interface 2012*, GI '12, Canadian Information Processing Society (Toronto, Ont., Canada, Canada, 2012), 117–120.
4. García, S., Luengo, J., and Herrera, F. *Data preprocessing in data mining*. Springer, 2015.
5. Goldberg, D., and Richardson, C. Touch-typing with a stylus. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, ACM (New York, NY, USA, 1993), 80–87.
6. Herold, J., and Stahovich, T. F. The 1-cent: Recognizer: A fast, accurate, and easy-to-implement handwritten gesture recognition technique. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '12, Eurographics Association (Aire-la-Ville, Switzerland, Switzerland, 2012), 39–46.
7. Leiva, L. A., Martín-Albo, D., and Plamondon, R. Gestures À go go: Authoring synthetic human-like stroke gestures using the kinematic theory of rapid movements. *ACM Trans. Intell. Syst. Technol. 7*, 2 (Nov. 2015), 15:1–15:29.
8. Li, Y. Protractor: A fast and accurate gesture recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, ACM (New York, NY, USA, 2010), 2169–2172.
9. Llorens, D., Prat, F., Marzal, A., Vilar, J. M., Castro, M. J., Amengual, J.-C., Barrachina, S., Castellanos, A., Boquera, S. E., Gómez, J., et al. The ujipenchars database: a pen-based database of isolated handwritten characters. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, and D. Tapias, Eds., European Language Resources Association (ELRA) (Marrakech, Morocco, may 2008). http://www.lrec-conf.org/proceedings/lrec2008/.
10. Mitchell, M., and Holland, J. H. When will a genetic algorithm outperform hill-climbing?
11. Park, H.-S., and Jun, C.-H. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications 36*, 2 (2009), 3336–3341.
12. Rubine, D. Specifying gestures by example. *SIGGRAPH Computer Graphics 25*, 4 (July 1991), 329–337.
13. Skalak, D. B. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Machine Learning: Proceedings of the Eleventh International Conference*, Morgan Kaufmann (1994), 293–301.
14. Taranta, II, E. M., and LaViola, Jr., J. J. Penny pincher: A blazing fast, highly accurate $-family recognizer. In *Proceedings of the 41st Graphics Interface Conference*, GI '15, Canadian Information Processing Society (Toronto, Ont., Canada, Canada, 2015), 195–202.
15. Vatavu, R.-D., Anthony, L., and Wobbrock, J. O. Gestures as point clouds: A $p recognizer for user interface prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, ACM (New York, NY, USA, 2012), 273–280.
16. Vatavu, R.-D., Vogel, D., Casiez, G., and Grisoni, L. Estimating the perceived difficulty of pen gestures. In *Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part II*, INTERACT'11, Springer-Verlag (Berlin, Heidelberg, 2011), 89–106.
17. Wobbrock, J. O., Wilson, A. D., and Li, Y. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, ACM (New York, NY, USA, 2007), 159–168.