# Random ferns method implementation for the general-purpose machine learning

Miron Bartosz Kursa[*]
Interdisciplinary Centre for Mathematical and Computational Modelling
University of Warsaw

February 7, 2012

**Abstract**

In this paper I present an extended implementation of the Random ferns algorithm contained in the R package *rFerns*.

It differs from the original by the ability of consuming categorical and numerical attributes instead of only binary ones. Also, instead of using simple attribute subspace ensemble it employs bagging and thus produce error approximation and variable importance measure modelled after Random forest algorithm.

I also present benchmarks' results which show that although Random ferns' accuracy is mostly smaller than achieved by Random forest, its speed and good quality of importance measure it provides make *rFerns* a reasonable choice for a specific applications.

## 1 Introduction

Random ferns is a machine learning algorithm proposed first by Özuysal et al [9] for patch matching, as a faster and simpler alternative to the Random forest algorithm. Since introduction, it has been used in numerous computer vision application, like image recognition [1], action recognition [8] or augmented reality [11].

However, it has not gathered attention outside this field; this work aims to bring this algorithm to much wider spectrum of applications by adapting it to use general information systems composed of continuous and categorical attributes.

Moreover, by further exploiting Random ferns' ensemble structure I have modified it so that it could produce OOB error approximation and attribute importance measure. Those modifications are described in Section 2, among a brief derivation of the Random ferns algorithm.

I have implemented this version of the Random ferns algorithm as an R [10] package *rFerns*; Section 3 contains benchmarks of *rFerns* accuracy and the quality of error

[*]E-mail: `M.Kursa@icm.edu.pl`

approximation and importance measure in several well known machine learning problems. Those results and computational performance of the algorithm are compared with Random forest implementation contained in *randomForest* package [6].

The conclusions are given in Section 4.

## 2 Algorithm

For simplicity, let's consider a classification problem based on an information system $(X_{i,j}, Y_i)$ with $M$ binary attributes $X_{\cdot,j}$ and $N$ objects equally distributed over $C$ disjoint classes. The generic *Maximum a Posteriori* (MAP) Bayes classifier classifies the sample $X_{i,\cdot}$ as

$$Y_i^p = \arg\max_y P(Y_i = y|X_{i,1}, X_{i,2}, \ldots, X_{i,M}); \qquad (1)$$

because of the Bayes theorem, it is equal to

$$Y_i^p = \arg\max_y P(X_{i,1}, X_{i,2}, \ldots, X_{i,M}|Y_i = y). \qquad (2)$$

This formula is strict, however it is not practically usable due to a combinatorial explosion of the number of possible $X_{i,\cdot}$ value mixes.

The simplest solution to this problem is to assume complete independence of the attributes, what brings us to the Naïve Bayes classification where

$$Y_i^p = \arg\max_y \prod_j P(X_{i,j}|Y_i = y). \qquad (3)$$

The original Random ferns classifier [2] was an in-between solution defining a series of $K$ subsets of $D$ features ($\vec{j}_k$, $k = 1, \ldots, K$) treated using a corresponding series of simple exact classifiers (ferns), which predictions are assumed independent and thus combined in a naïve way, i.e.

$$Y_i^p = \arg\max_y \prod_k P(X_{i,\vec{j}_k}|Y_i = y); \qquad (4)$$

this way one can still represent some non-linear interactions in the system, possibly achieving better accuracy than in purely naïve case. On the other hand, such defined classifier is still very simple and computationally manageable for a range of $D$ values. $\vec{j}_k$ are usually generated randomly to omit the impact of arbitrary order of attributes in the information system; thus $K$ is considered a parameter of the training procedure.

In the original implementation of Random ferns, the probabilities $P_{i,\vec{j}_k}(y) := P(X_{i,\vec{j}_k}|Y_i = y)$ are estimated on a training system $(X^t_{i,j}, Y^t_i)$ by counting the frequency of each class in each subspace of attribute space defined by $\vec{j}_k$ with a Dirichlet prior, so

$$P_{i,\vec{j}_k}(y) = \frac{1}{\#L_{i,\vec{j}_k} + C}\left(1 + \#\left\{m \in L_{i,\vec{j}_k} : Y^t_m = y\right\}\right), \qquad (5)$$

where # denotes set size and

$$L_{i,\vec{j}_k} = \left\{ l \in [1; N^t] : X^t_{l,\vec{j}_k} = X_{i,\vec{j}_k} \right\} \tag{6}$$

is the set of objects in the same leaf of fern $k$ as object $i$.

In order to prevent numerical problems, instead of $P_{i,\vec{j}_k}(y)$ one can work on scores $S_{i,\vec{j}_k}(y)$ defined as

$$S_{i,\vec{j}_k}(y) := \log P_{i,\vec{j}_k}(y) + \log C. \tag{7}$$

This representation is monotonic, so the MAP rule persists as a selection of class with the maximal sum of scores,

$$Y^p_i = \arg\max_y \prod_k P_{i,\vec{j}_k}(y) = \arg\max_y \sum_k S_{i,\vec{j}_k}(y); \tag{8}$$

moreover a fern which knows nothing about some objects give it a score vector of 0s instead of $C^{-1}$s, what greatly reduces the magnitude of final votes.

Furthermore, by using additive scores we can move from a Bayesian legacy of Random ferns to a ensemble of classifiers language — in this view we have a random subspace [4] battery of classifiers combined by mean voting.

Going further in that direction, we can replace random subspace ensemble with bagging, i.e. by building each fern on a set of objects build by sampling with replacement $N_t$ objects from an original training system, thus changing Equation 6 into

$$L_{i,\vec{j}_k} = \left\{ l \in B_k : X^t_{l,\vec{j}} = X_{i,\vec{j}} \right\}, \tag{9}$$

where $B_k$ is the set of indexes of objects sampled for training, called *bag*. This way, each fern has a set of at average $0.368N^t$ objects which was not used to build it; it is usually called *out-of-bag* (OOB) subset and will be denoted here as $B^*_k$.

This setup allows one to migrate out-of-bag error approximation and permutational importance measure.

## 2.1 Generalisation beyond binary attributes

Both continuous and categorical attributes can be exactly represented as a series of binary attributes, thus any information system containing those type of features is equivalent to a larger binary information system.

We can omit the necessity of building it before training by generating effective binary attributes on-the-fly. Namely, when a continuous attribute $X_{\cdot,i}$ is selected for creation of a fern level a random threshold $\xi$ is generated and this attribute is treated as a binary $(X_{\cdot,i} < \xi)$. Correspondingly, for a categorical attribute $X_{\cdot,j}$ a random subset of levels $\Xi$ is generated and this attribute is treated as $(X_{\cdot,j} \in \Xi)$. In my implementation, I have used a heuristic to select $\xi$ as a mean of two randomly selected values of $X_{\cdot,i}$; $\Xi$ is drawn uniformly from all possible subsets except empty and containing all levels.

Obviously, completely random generation of splits is usually less effective in terms of the accuracy of a final classifier than selecting thresholds in a greedy way via some optimisation procedure (for instance based on information gain or Gini impurity). It is neither significantly more computationally effective approach because such optimisation may be implemented with even linear complexity in the number of objects.

However, this way the classifier can escape certain overfitting scenarios and unveil more subtle interaction. This and the more even usage of attributes may be beneficial both for the robustness of the model and the accuracy of the importance measure it provides.

While the scores depend on $\xi$ and $\Xi$ values in this generalisation, from now on I will denote them as $S_{i,F_k}$ where $F_k = (\vec{j}_k, \vec{\xi}_k, \vec{\Xi}_k)$.

## 2.2 Error estimate

While each fern was build only using its in-bag objects, one can use the OOB subset to reliably test its classification performance. Thus one can calculate an error approximation of the whole ensemble by comparing true classes of objects $Y_i$ with corresponding OOB predictions $Y_i^*$ defined as

$$Y_i^* = \arg\max_y \sum_{k:i\in B_k^*} S_{i,F_k}(y). \qquad (10)$$

Note that for small values of K some objects may not manage to appear in any bag, and thus get an undefined OOB prediction.

## 2.3 Importance measure

The core idea behind the permutational importance score is to measure how the performance of the classification decreases due to a permutation of values within an investigated attribute.

Instead of a simple approach of measuring the misclassification rate, I have postulated a measure that will provide information even for cases when the object is misclassified by unperturbed classifier, namely the difference in the OOB score of a correct class of an object. Precisely, the importance of attribute $a$ equals

$$I_a = \left\langle \left\langle S_{i,F_k}(Y_i) - S^p_{i,F_k}(Y_i) \right\rangle_{i\in B_k^*} \right\rangle_{k:a\in\vec{j}_k}, \qquad (11)$$

where $S^p_{i,\vec{j}_k}$ is $S_{i,\vec{j}_k}$ calculated on a permuted $X$ in which values of attribute $a$ have been shuffled.

One should also note that the fully stochastic nature of selecting attributes for building partial ferns guarantees that the attribute space is evenly sampled and all, even marginally relevant attributes are included in the model for large enough number of ferns.

## 2.4 Unbalanced classes case

When the distribution of the classes in the training decision vector is not uniform, the classifier structure gets biased in recognition of the most represented classes. While this phenomenon may lead to a better accuracy in sharply unbalanced cases, it is usually undesired because it strips the model from information about smaller classes.

Thus, *rFerns* is internally balancing class' impacts by diving the counts of objects of a certain class in a current leaf by the fraction of objects of that class in the bag set of the current fern — this is roughly equivalent to oversampling underrepresented classes so that the amounts of objects of each class are equal within bag.

One should note that the knowledge of the class distribution prior $P(y)$ for the test data (for instance the assumption that it will be the same as in training data) can still be used to improve the prediction accuracy. To this end, one must request the prediction code to return the raw scores and correct them by adding logarithm of respective priors, i.e.

$$S^c_{i,F_k}(y) = S_{i,F_k}(y) + \log P(y), \tag{12}$$

and use so corrected scores in the MAP rule.

## 3 Benchmarks

I have tested *rFerns* on 7 classification problems from the R's *mlbench* [5] package, namely *DNA* (DNA), *Ionosphere* (ION), *Pima Indian Diabetes* (PIM), *Satellite* (SAT), *Sonar* (SON), *Vehicle* (VEH) and *Vowel* (VOW).

### 3.1 Accuracy

For each of the testing sets, I have built 10 Random ferns models for each of the depths in range $[1; 15]$ and number of ferns equal to 5000 and collected the OOB error approximations.

Next, I have used those results to find optimal depths for each set ($D_b$) — for simplicity I selected value for which the mean OOB error from all iterations was minimal.

Finally, I have verified the error approximation by running 10-fold stochastic cross-validation. Namely, the set was randomly slit into test and training subsets, composed respectively of 10% and 90% of objects; the classifier was then trained on a training subset and its performance was assessed using the test set. Such procedure has been repeated two times.

As a comparison, I have also built and cross-validated 10 Random forest models with 5000 trees. The ensemble size was selected so that both algorithm would manage to converge for all problems.

The results of those tests are collected in the Table 1. One can see that as in case of Random forest, OOB error approximation is a good estimate of the final classifier error. It is also well serves as an optimisation target for the fern depth selection — only in case of the Sonar data the naïve selection of the depth giving minimal OOB error led to a suboptimal final classifier, however one should note that the minimum was not significant in this case.

Based on the OOB approximations, forest outperforms ferns in all but one case; yet the results of cross-validation show that those differences are in practice masked

|  | Set | DNA | ION | PIM | SAT |
|---|---|---|---|---|---|
|  | Set size | $3186 \times 180$ | $351 \times 34$ | $392 \times 8$ | $6435 \times 36$ |
| OOB [%] | Ferns 5 | $6.03 \pm 0.18$ | $7.32 \pm 0.23$ | $24.69 \pm 0.48$ | $18.40 \pm 0.13$ |
|  | Ferns 10 | $6.56 \pm 0.11$ | $7.35 \pm 0.22$ | $27.93 \pm 0.30$ | $15.46 \pm 0.06$ |
|  | Ferns $D_b$ | $6.03 \pm 0.18$ | $7.07 \pm 0.40$ | $23.95 \pm 0.31$ | $14.33 \pm 0.05$ |
|  | Forest | $4.13 \pm 0.09$ | $6.55 \pm 0.00$ | $21.76 \pm 0.36$ | $7.87 \pm 0.06$ |
| CV [%] | Ferns 5 | $6.52 \pm 1.66$ | $7.78 \pm 3.41$ | $24.50 \pm 6.75$ | $18.60 \pm 1.32$ |
|  | Ferns 10 | $6.96 \pm 1.30$ | $8.61 \pm 3.81$ | $29.50 \pm 6.10$ | $15.92 \pm 1.30$ |
|  | Ferns $D_b$ | $5.92 \pm 1.41$ | $5.00 \pm 3.88$ | $24.00 \pm 6.99$ | $14.32 \pm 0.88$ |
|  | Forest | $4.20 \pm 0.99$ | $6.11 \pm 4.68$ | $21.00 \pm 3.94$ | $7.75 \pm 1.54$ |
|  | $D_b$ | 5 | 3 | 7 | 15 |

|  | Set | SON | VEH | VOW |  |
|---|---|---|---|---|---|
|  | Set size | $208 \times 60$ | $846 \times 18$ | $990 \times 10$ |  |
| OOB [%] | Ferns 5 | $19.71 \pm 0.60$ | $31.17 \pm 0.49$ | $13.70 \pm 0.52$ |  |
|  | Ferns 10 | $14.18 \pm 1.12$ | $29.52 \pm 0.23$ | $4.42 \pm 0.26$ |  |
|  | Ferns $D_b$ | $13.13 \pm 0.64$ | $28.83 \pm 0.49$ | $2.41 \pm 0.19$ |  |
|  | Forest | $15.38 \pm 0.64$ | $25.48 \pm 0.18$ | $2.13 \pm 0.11$ |  |
| CV [%] | Ferns 5 | $22.38 \pm 6.37$ | $32.94 \pm 4.15$ | $17.07 \pm 3.10$ |  |
|  | Ferns 10 | $14.29 \pm 5.94$ | $29.41 \pm 7.48$ | $5.25 \pm 1.64$ |  |
|  | Ferns $D_b$ | $18.10 \pm 4.92$ | $28.71 \pm 5.69$ | $2.22 \pm 1.77$ |  |
|  | Forest | $19.52 \pm 8.53$ | $22.35 \pm 4.33$ | $2.22 \pm 1.70$ |  |
|  | $D_b$ | 12 | 15 | 15 |  |

Table 1: OOB and cross-validation error of the Random ferns classifier for 5000 ferns of a depth equal to 5, 10 and optimal over $[1;15]$, $D_b$. Those results are compared to the accuracy of a Random forest classifier composed of 5000 trees. Prediction errors are given as a mean and standard deviation over 10 repetitions of training for OOB and 10 iterations for cross-validation.

by the natural variability of both classifiers. Only in case of the Satellite data Random forest clearly gives almost two times smaller error.

## 3.2 Importance

To test importance measure, I have used two sets for which importance of attributes should follow certain pattern.

Each objects in DNA set [7] represent 60-residue DNA sequence in a way so that each consecutive triple of attributes encode one residue. Some of the sequences contain a boundary between exon and intron (or intron and exon[1]) and the objective is to recognise and classify those sequences. All sequences were aligned in a way that the boundary always lies between 30th and 31st residue; while the biological process of recognition is local, the most important attributes should be those in vicinity of the boundary.

[1]The direction of a DNA sequence is significant, so those are separate classes.

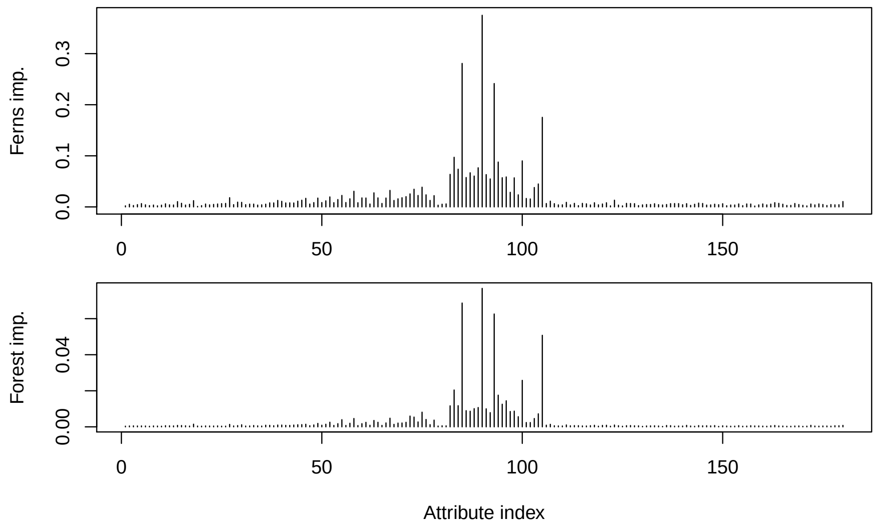

Figure 1: Attribute importance for DNA data, generated by Random ferns (*top*) and, for comparison, by Random forest (*bottom*). Note that the importance peaks around 90th attribute, corresponding to an actual splicing site.

Objects in Sonar set [3] correspond to echoes of a sonar signal bounced off either a rock or a metal cylinder (a model of a mine). They are represented as power spectra, thus each next attribute corresponds to a consecutive frequency interval. This way, one may expect that there exist frequency bands in which echoes significantly differ between classes — and this would manifest as a set of peaks in the importance measure vector.

For both of this sets, I have calculated the importance measure using 1000 ferns of a depth 10. As a datum, I have also calculated permutational importance measure using Random forest algorithm with 1000 trees.

The results are presented on Figure 1 and 2. The importance measures obtained is both cases are consistent with the expectations based on the sets' structures — for DNA, one can notice a maximum around attributes 90–96, corresponding the actual cleaving location. For Sonar, the importance scores reveal a band structure which likely corresponds to the actual frequency intervals in which the echoes differ between stone and metal.

The both results are also qualitatively in agreement with those obtained from Random forest models. Qualitative difference comes form the completely different formulations of both measures and possibly the higher sensitivity of ferns resulting from its fully stochastic construction.

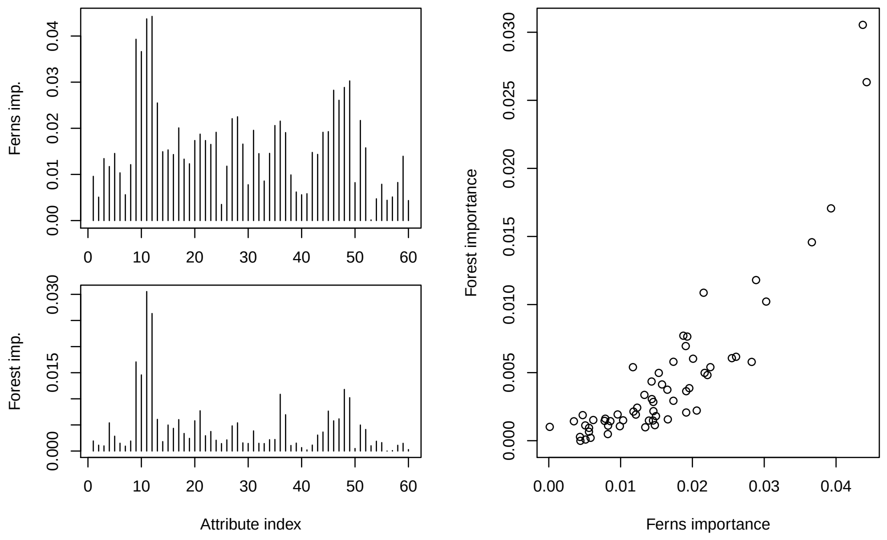

Figure 2: Importance measure for Sonar data, generated by Random ferns (*top right*) and, for comparison, by Random forest (*bottom right*). Those two measures are compared on a scatterplot (*left*).

| | Set | Forest [s] | Ferns 10 [s] | Speedup | Ferns $D_b$ [s] | Speedup | $D_b$ |
|---|---|---|---|---|---|---|---|
| Just training | DNA | 30.13 | 1.96 | 15.41 | 0.65 | 46.28 | 5 |
| | ION | 3.41 | 0.81 | 4.21 | 0.06 | 55.35 | 3 |
| | PIM | 2.45 | 0.97 | 2.53 | 0.24 | 10.29 | 7 |
| | SAT | 136.91 | 4.08 | 33.55 | 75.55 | 1.81 | 15 |
| | SON | 3.40 | 0.78 | 4.37 | 2.95 | 1.15 | 12 |
| | VEH | 8.00 | 1.67 | 4.80 | 46.04 | 0.17 | 15 |
| | VOW | 93.92 | 4.77 | 19.69 | 140.26 | 0.67 | 15 |
| With importance | DNA | 456.16 | 4.37 | 104.38 | 1.87 | 244.31 | 5 |
| | ION | 7.34 | 1.46 | 5.02 | 0.25 | 29.32 | 3 |
| | PIM | 3.53 | 1.06 | 3.34 | 0.35 | 10.17 | 7 |
| | SAT | 289.26 | 8.77 | 32.98 | 82.80 | 3.49 | 15 |
| | SON | 4.83 | 0.93 | 5.18 | 3.13 | 1.54 | 12 |
| | VEH | 17.26 | 2.22 | 7.77 | 47.00 | 0.37 | 15 |
| | VOW | 99.26 | 5.40 | 18.39 | 141.13 | 0.70 | 15 |

Table 2: Training times of the *rFerns* and *randomForest* models made for 5000 base classifiers, with and without importance calculation. Times are given as a mean over 10 repetitions.

## 3.3 Computational performance

In order to compare training times of *rFerns* and *randomForest* codes, I have trained both models on all 7 benchmark sets for 5000 ferns/trees, and, in case of ferns, depths 10 and $D_b$. Than I have repeated this procedure, this time making both algorithms calculate importance measure during training.

I have repeated both tests 10 times to stabilise the results and collected the mean execution times; the results are shown in the Table 2. The results show that the usage of *rFerns* may result is significant speedups in certain applications; best speedups are achieved for the sets with larger number of objects, which is caused by the fact that Random ferns' training time scales linearly with the number of objects, while Random forest $\sim N \log N$.

Also the importance can be calculated significantly faster by *rFerns* than by *randomForest*, and the gain increases with the size of the set.

*rFerns* is least effective for sets which require large depths of the fern — in case of Vowel and Vehicle sets it was even slower than Random forest. However, one should note that while the complexity of Random ferns $\sim 2^D$, its accuracy usually decreases much less dramatically when decreasing $D$ from its optimal value — this way one may expect an effective trade-off between speed and accuracy.

## 4 Conclusions

In this paper, I have presented *rFerns*, a general-purpose implementation of the Random ferns classifier. Slight modifications of the original algorithm allowed me to additionally implement OOB error approximation and permutational attribute importance measure.

Benchmarks showed that such algorithm can achieve accuracies comparable to Random forest algorithm while usually being much faster, especially for large datasets.

Also the importance measure proposed in this paper can be calculated very quickly and proved to behave in a desired way and be in agreement with the results of Random forest; however the in-depth assessment of its quality and usability for feature selection and similar problems requires further research.

## Acknowledgements

This work has been financed by the National Science Centre, grant 2011/01/N/ST6/07035. Computations were performed at ICM, grant G48-6.

## References

[1] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image Classification using Random Forests and Ferns. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[2] Leo Breiman. Random Forests. *Machine Learning*, 45:5–32, 2001.

[3] Paul R. Gorman and Terrence J. Sejnowski. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. *Neural Networks*, 1:75–89, 1988.

[4] T.K. Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, 1998.

[5] Friedrich Leisch and Evgenia Dimitriadou. *mlbench: Machine Learning Benchmark Problems*, 2010. R package version 2.1-0.

[6] Andy Liaw and Matthew Wiener. Classification and Regression by randomForest. *R News*, 2(3):18–22, 2002.

[7] Michiel O. Noordewier, Geoffrey G. Towell, and Jude W. Shavlik. Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences. *Advances in Neural Information Processing Systems*, 3, 1991.

[8] Olusegun Oshin, Andrew Gilbert, John Illingworth, and Richard Bowden. Action recognition using randomised ferns. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference*, pages 530–537. IEEE, 2009.

[9] Mustafa Özuysal, Pascal Fua, and Vincent Lepetit. Fast Keypoint Recognition in Ten Lines of Code. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[10] R Development Core Team. R: A Language and Environment for Statistical Computing, 2010.

[11] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE transactions on visualization and computer graphics*, 16(3):355–68, 2010.