# Incorporating On-demand Stereo for Real Time Recognition

T. Deselaers[1,2], A. Criminisi[1], J. Winn[1] and A. Agarwal[1]
[1]Microsoft Research Ltd., Cambridge, United Kingdom
[2]Human Language Technology and Pattern Recognition, RWTH Aachen University, Germany
deselaers@cs.rwth-aachen.de, {antcrim,jwinn,ankagar}@microsoft.com

## Abstract

*A new method for localising and recognising hand poses* and *objects in real-time is presented. This problem is important in vision-driven applications where it is natural for a user to combine hand gestures and real objects when interacting with a machine. Examples include using a real eraser to remove words from a document displayed on an electronic surface.*

*In this paper the task of simultaneously recognising object classes, hand gestures and detecting touch events is cast as a* single *classification problem. A random forest algorithm is employed which adaptively selects and combines a minimal set of appearance, shape and stereo features to achieve maximum class discrimination for a given image. This minimal set leads to both efficiency at run time and good generalisation. Unlike previous stereo works which explicitly construct disparity maps, here the stereo matching costs are used directly as visual cue and only computed on-demand,* i.e. *only for pixels where they are necessary for recognition. This leads to improved efficiency.*

*The proposed method is assessed on a database of a variety of objects and hand poses selected for interacting on a flat surface in an office environment.*

## 1. Introduction

This paper presents a unified algorithm for the automatic recognition and localisation of hand poses and object classes in real-time. The algorithm has been designed for (but is not restricted to) scenarios where a person makes use of his/her hands together with common physical objects to drive an application either displayed on a tablet screen or projected onto a table top. Figure 1 shows an example application where a user edits an electronic document using common office objects such as pens and erasers while manipulating it via natural hand gestures.

Enabling intuitive and natural interaction in such a scenario requires accurate and efficient recognition of different hand poses and a variety of object classes[1], together with the ability to detect when the user touches the workspace surface. Current touch-screen technology and hand-based user interfaces do not support all these requirements [2].

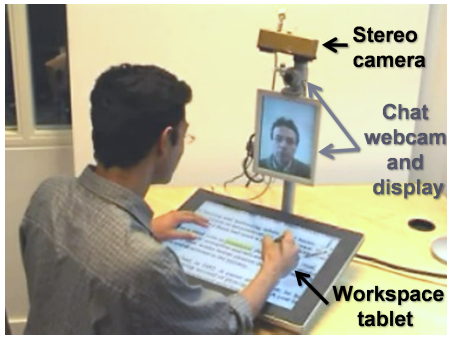



Figure 1. **An example hardware setup and application.** A user edits an electronic document by using his hands, real pens, and erasers. An overhead stereo camera is used both for gesture/object recognition and touch detection. A web cam and small display may also be used to aid collaboration with remote users.

Although within the seemingly constrained setup considered here (see fig. 1), the problem of recognising different hand poses and objects remains challenging due to the large variability in lighting conditions, skin colour, hand sizes, object appearance (*e.g.* different types of cell phones), the presence of sleeves etc. (fig. 3). Furthermore, touch vs. non-touch discrimination must be robust with respect to different camera setups, hand poses and cases where the hands may or may not be holding objects.

Previously, the problems of recognising object classes [4, 11, 16, 22], hand gestures [9, 19] and touch detection [18, 20, 21] have been treated separately. In contrast, the stereo- and learning-based approach proposed here addresses all three problems within a *unified* classification framework.

The discriminative classifier developed here builds upon recent advances in random forest learning [1, 8, 10, 24], and is capable of efficiently combining appearance, shape and depth cues to achieve accurate class discrimination. An illustration of the importance of depth cues both for touch detection and as an aid to recognition is shown in fig. 2.

Existing work in stereo vision has concentrated on the

[1]Unlike previous work on surface computing [20] here all physical objects are assumed to be untagged.

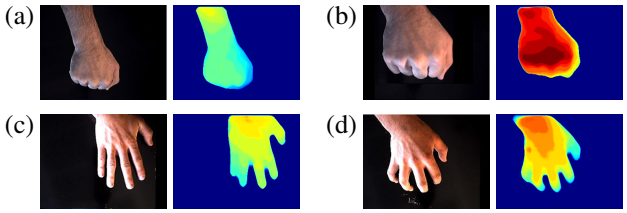



Figure 2. **Depth cues for touch detection and hand pose recognition.** Examples of different hand poses and the corresponding disparity maps (computed using a conventional stereo algorithm; red indicates large disparities and blue small ones). (a) A hand in the "fist" pose, *touching* the working surface, and (b) *above* the surface. Stereo cues help distinguish "touch" and "no-touch" cases. (c,d) Images of a hand in the "flat" and "spider" pose, respectively. In (c) the whole hand is touching the surface, whereas in (d) only the fingertips are touching. Although the shape of the two silhouettes (c,d) is very similar, the corresponding disparity maps contain sufficient information to disambiguate the two cases.

extraction of disparity maps [3, 7, 14, 17] from image pairs. However, accurate disparities require incorporating spatial smoothness priors, which often involve running computationally expensive procedures such as alpha expansion graph cuts, belief propagation or dynamic programming. Examples of papers which tackle the problem of efficient disparity map computation are [15, 23]. However, those approaches still produce *whole* disparity maps as output. Another step towards using stereo information efficiently was presented in [6] for purposes of foreground/background segmentation. It was shown that rather than solving the 'full stereo problem' and computing disparity maps, cheaper stereo likelihoods could be computed for each pixel. In this paper, the idea of cheap stereo is investigated further. In fact, instead of computing disparity maps, stereo matching costs are used directly as visual features for classification. Furthermore, those stereo features are calculated only at those pixels which require depth cues for improved class-discrimination. This, together with a new training-method penalising computationally expensive features enables real-time performance.

The remainder of the paper is structured as follows; section 2 describes the database used to train the system and evaluate its performance; section 3 presents the algorithm for segmenting the foreground from the background surface. The main contribution of this paper is introduced in section 4 which describes the recognition and touch detection algorithms. Experimental results are in section 5.

## 2. The database

A fully labelled database of several different hand poses and objects was constructed for purposes of training and testing. Fig. 3 shows example images. The database consists of stereo image pairs of 45 classes, broadly divided into the following three different groups. (a) Hand poses (12 classes): fist, fist (side view), flat hand, flat hand (side view), picking, point with 1 finger, point with 2 fingers, right angle, ring, spider, thumb up and thumb up (side view). (b) Objects held by a hand (10 classes): black pen, blue pen, red pen, green pen, pencil, stylus (from the tablet display), eraser, scissors, post-it note and paper. (c) Objects on their own (2 classes): cell phone and sticky tape. For each of the classes from categories (a) and (b), examples where the hand (or the object) are touching the screen are labelled as being different from cases where they are above the screen. There are thus effectively twice the number of classes as the number of hand poses/objects. The two objects in (c) always occur on their own and are always touching the surface. Finally, scissors never touch the surface.

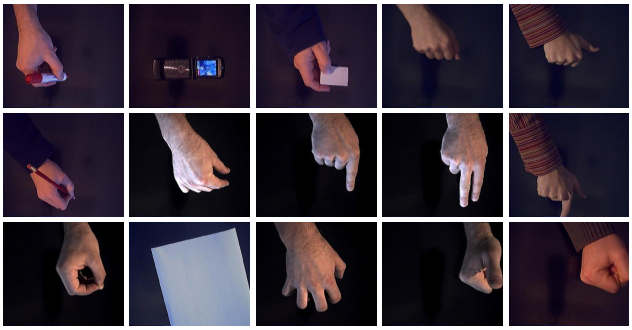


Figure 3. **Database of stereo images.** Example images (only left views shown here). The database comprises images of different people's hands in 12 different poses and images of 12 different objects classes. The large variability in lighting conditions, skin colour and the presence of sleeves makes recognition challenging.

For each class we have captured 100 images, divided randomly into 50 training and 50 test images. The images were captured under different lighting conditions, with different people's hands (3 female and 5 male) in different poses. For each class both "touch" and "no-touch" image pairs were recorded. Epipolar rectification of all stereo pairs is achieved using conventional techniques [5].

## 3. Foreground extraction

Before any object/gesture can be recognised the foreground needs to be segmented from the background. This section describes the segmentation algorithms employed.

In the case where the workspace surface is an LCD display (as in the example in fig. 1), placing polarising filters before the cameras (orthogonally to the display's own polarisation) achieves background suppression. However, our recognition system may also be employed with non-electronic surfaces such as a wooden desk top. In this case, in order to perform segmentation while allowing for possible camera jitter and changes in lighting conditions the following learning approach is employed.

**Training.** A small set of images containing foreground (hand/objects) exemplars are manually segmented (off-line)

and stored. Additionally, some images of the "clean" background (e.g. wooden desk) are captured under slightly different lighting conditions and different camera positions. A decision tree is then trained [12] on those segmentation masks, so as to learn to discriminate between background and foreground. The tests used in the tree nodes are the same appearance features that are employed in the classification stage and will be described in section 4.2. Foreground models and segmentation tree can now be refined by: i) adding more, *unsegmented* images containing foreground; ii) using the learnt tree to segment them and; iii) re-training of the tree on *all* obtained segmentation masks.

**Segmentation.** During testing the learned decision tree is applied to each pixel in an input image. Thus, each pixel is associated with a label, i.e. the index of the terminal node reached when the tree is applied at that pixel (fig. 4, top). A foreground/background segmentation mask is obtained by energy minimisation, similar to GrabCut [13], but with two important differences. First, in GrabCut appearance is modelled by means of Gaussian Mixture Models in RGB space. In contrast, in our system the output of the decision tree is used as unary potentials, i.e. histograms over the leaf labels are used to model appearance of foreground and background. Second, our system does not require manual initialisation; instead automatic initialisation is achieved by using aggregate foreground and background histograms across the training set described earlier. As in [13], we alternately update the segmentation and the foreground/background histograms for a number of iterations (typically just two for an optimal speed/accuracy trade off).

## 4. Fusing appearance, shape and depth for recognition

This section assumes that foreground/background separation has been achieved and focuses on the recognition of the foreground region[2].

### 4.1. Random Forests for classification

Recent work has demonstrated the effectiveness of Random Forests algorithms for classification (*e.g.* [1]), particularly with regard to speed and robustness to overfitting. In this paper a number of random trees are trained to select and combine various visual cues and achieve good class discrimination.

Given an input image and its foreground mask, a decision tree trained to discriminate between different classes using tests[3] $t_i$ is applied to each foreground pixel leading to a leaf label for each pixel. The leaf label corresponds to the index of the leaf reached in the tree (fig. 4, bottom). A histogram over all such labels for the foreground region is computed and classified using a nearest neighbour classifier with Bhattacharyya distance. This method is extended to use multiple decision trees (a decision forest) by concatenating the histograms prior to classification.

The key strength of random forests is that the trees in the forest capture different cues due to the randomness in the learning process. Every tree is trained independently of the others which allows for parallel training of several trees. In each training iteration, a pool of 200 randomly selected test candidates $\mathcal{T} = \{t_1, \ldots, t_I\}$ is generated and at each leaf the test candidate $t_i$ with the highest entropy gain[4] is appended as a node in the tree. This procedure is repeated until no further leaves are appended.

In Random Forests, each tree is allowed to overfit during training; but averaging over outputs of multiple trees leads to good testing generalisation. In many cases Random Forests have been proved to achieve better generalisation than AdaBoost despite higher training errors [1, 24]. The tests used to build the tree are described in the next section.

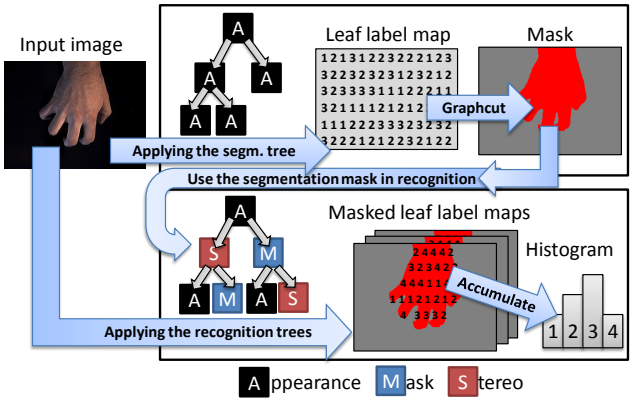



Figure 4. **Our segmentation and recognition system. (top box)** The segmentation phase. A decision tree trained to discriminate foreground and background is applied to the image to get a "leaf label map". See text. The likelihoods of foreground and background are then computed from the label map and a min-cut algorithm applied to obtain a segmentation mask. Only appearance features are used in this phase. **(bottom box)** The recognition phase. A Random Forest with $T$ trees, trained to discriminate between different object classes is applied to an input stereo pair to produce $T$ new label maps. For each connected foreground region a histogram of leaf labels is accumulated and classified via nearest neighbours. Several visual features are combined in this phase.

### 4.2. Visual features

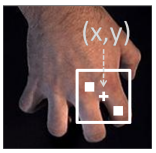
This section describes the different visual cues, the features that are used, and the tests that are made available to the decision tree learning.

We use a combination of appearance, shape and depth visual cues. For each of the different visual cues, texture-

[2] Since the recognition process is independent of segmentation, a large part of our training dataset actually consists of pre-segmented images obtained using a black background.

[3] A *test* here refers to a boolean decision rule applied to a feature vector.

[4] Provided the gain is above a certain threshold.

(a) 

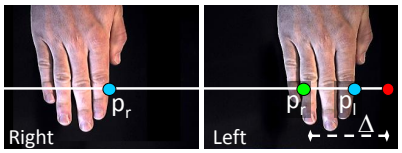
(b) 


Figure 5. **Visual features.** (a) An example for a possible test in a decision tree. The test is applied to the pixel marked with the cross and the difference between two randomly selected pixels chosen within a bounding box of 50×50 pixels is compared to two thresholds. (b) For a pair of corresponding pixels $p_l$ and $p_r$ from a stereo image pair the disparity $d$ is computed as $d = p_l - p_r$.

like filters are applied to obtain features $f_i$ which are used in the tests $t_i$ of the decision tree nodes (one test per node). Generally, the tests compare the values of the features $f_i$ with two randomly selected thresholds (see fig. 5a). Thus, the tests for a pixel in $(x, y)$ are of the following form:

$$t(x, y) \quad = \quad \theta_1 < f_i(x, y) < \theta_2, \qquad (1)$$

where $f_i(x, y)$ is a feature function for position $(x, y)$ in the image, and $\theta_1$ and $\theta_2$ denote thresholds.

The features $f_i$ can be of different forms, either they are the difference of two randomly selected pixels $p_1$ and $p_2$ within a bounding box centred in $(x, y)$ *(type 1)* or they are just the value of one randomly selected pixel in this area *(type 2)*:

$$\begin{aligned} f_1(x, y) \quad &= p_1(x + x', y + y') \\ &\quad -p_2(x + x'', y + y'') \quad \text{(type 1)} \\ f_2(x, y) \quad &= p_1(x + x', x + y') \quad \text{(type 2)} \end{aligned}$$

where $p_1(x, y)$ and $p_2(x, y)$ denote the values of pixels at these positions in the specified colour channels. All parameters and the type of the tests are randomly chosen in the generation of the test candidates. An example test is shown in fig. 5a. It has been shown, that using sufficiently large bounding boxes allows to capture long-range interactions in the images [16]. These tests are applied to different visual cues and slightly adapted to the specific character of the cue. Next we describe details of the different visual cues used here:

**Appearance.** To capture appearance information, tests are applied to RGB colour channels. In the training process, 90% of all randomly generated appearance (colour) features are of type 1 (*i.e.* pixel differences), and the remaining ten percent are of type 2 (*i.e.* absolute values). The advantage of pixel differences for appearance cues is a better invariance with respect to lighting changes.

**Shape.** The binary mask obtained from the segmentation stage is used as an additional channel in the recognition phase to capture shape information. The tests applied to this (binary) mask are of the same type as those applied to the appearance channels.

**Depth.** Depth information is essential for reliable touch/no touch discrimination. Given a pair of rectified stereo images, depth information can be obtained by finding the correspondence of the pixels in the left image to the pixels in the right image for every scanline as depicted in fig. 5b. For every pixel $p_r$ in a scanline of the right image, the corresponding pixel $p_l$ in the left image is determined within a certain disparity range $\Delta$. The displacement of $p_r$ to $p_l$ is the disparity ($d = p_l - p_r$) and is inversely proportional to the distance of the observed point in the image to the camera. Usually, to determine the correspondences of the pixel pairs, the sums of squared distances (SSD) between all pairs of pixels of a scanline is calculated and the optimal alignment of all pixels of the left scanline to all pixels of the right scanline is determined by algorithms such as dynamic programming, belief propagation or graph cuts.

Direct incorporation of high quality disparity maps in our framework is straightforward but highly inefficient. Using low resolution disparity maps, would result in more efficient algorithms but with high costs in the accuracy of touch detection.

Yet another way to incorporate stereo information efficiently is to use mean and variance disparities or winner-takes-all (WTA) disparities. For the latter, the calculation does not require any spatial coherence and therefore the computation is cheap. The mean disparity and its variance are calculated as $\overline{d} = \sum_{d=0}^{\Delta} d \cdot p(d)$ and $\sigma_d^2 = \sum_{d=0}^{\Delta} (d - \overline{d})^2 \cdot p(d)$, respectively, where $p(d) = \exp\left(-\gamma SSD(p_r, p_r + d)\right)/Z$ and $SSD$ is the sum of squared distances of 3×7 patches around the pixels in the current scanline of the left and right images. $Z$ is a normalisation factor. WTA disparities are calculated as $\hat{d} = \arg\max_d p(d)$. To calculate the mean, variance, and WTA disparities, for each pixel matching costs for the whole disparity range $\Delta$ must be computed.

To reduce the number of necessary computations further we develop new stereo features by subsampling the search interval $\Delta$ into a small number of allowed disparities $d_i$ and extracting the corresponding slices from the whole 3D matching cost space (an illustration is shown in fig. 6). We call these '*disparity cost slices*' (DCS). The idea is to use stereo matching costs *directly* as visual features in our learning framework.

The stereo features used in the test candidates for training of trees are chosen such that 50% of the features are of type 2 (*i.e.* absolute disparity/SSD cost) and the remaining 50% of the features are absolute disparity/cost differences.

### 4.3. On-demand stereo and cost-aware training

**On-demand stereo.** In decision trees, in contrast to other classifiers such as AdaBoost, not every test is applied to every observation (here: pixel). Instead, for every pixel, only the tests along one path through the tree (from root to leaf) are evaluated. Therefore, we can avoid computing the full cost space volume (or the selected full slices) which would

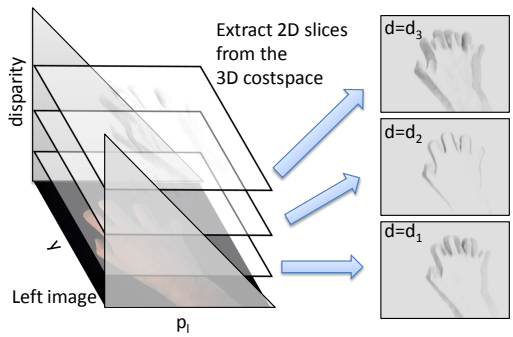



Figure 6. **Disparity Cost Slice stereo features**. Sampled stereo matching costs are used directly as visual features in our learning framework. Note how the slice corresponding to $d = d_2$ shows a near perfect alignment of left and right images, suggesting $d_2$ as the correct depth for the hand.

lead to many unnecessary operations. We only compute stereo matching costs for those pixels that require depth information, as dictated by the learned decision tree. This concept applies to all features, but is most important for the stereo ones because of their computational cost.

**Cost-aware training.** A further speedup is obtained by training the tree *cost aware*. Conventional tree training minimises an entropy criterion, here we modify this criterion to optimise with respect to efficiency *and* classification accuracy jointly. This is achieved by adding a weighted version of the expected cost of evaluating the test. Hence tree learning now minimises a trade off of discriminatory performance and speed. This is implemented by dividing the entropy gain for a test by a penalty value accounting for its computational cost. Thus, more expensive tests are pushed toward the bottom of the tree and applied to fewer pixels.

In our case, the evaluation of tests for the stereo cues (even for the DCS stereo slices) is more costly than evaluating appearance or shape tests. The penalisation of stereo tests allows us to smoothly blend between using stereo features and not using them at all.

**Memory caching.** Finally, a caching mechanism is implemented to avoid repeating calculations. An informal experimental evaluation with a subset of images has shown that the cache is used in 16% of all accesses to the cost space and leads to a significant performance increase. More details are given in section 5.

## 5. Results

**A look at the selected features.** Fig. 7 shows a typical tree being learnt from our training data, with colour coding indicating the extent to which each cue is used. Appearance, shape and depth features are all used throughout the tree. Table 1 provides a quantitative overview on the features that

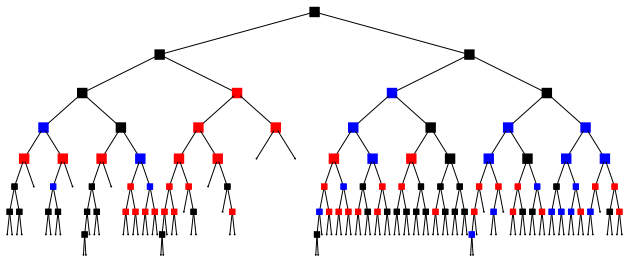


Figure 7. **Selected visual cues**. A decision tree trained with all three visual cues enabled. Like in fig. 4 black indicates appearance, blue denotes shape and red stereo tests.

Table 1. **Overview of the tests selected by tree training.** The figures are averaged over six random decision trees and subdivided based on their visual cues.

| visual cue | tests [%] | type-2 tests [%] | avg. pixel dist. |
|---|---|---|---|
| appearance | 55 | 47 | 26.8 |
| stereo | 32 | 78 | 13.7 |
| shape | 13 | 46 | 37.8 |

are selected when training decision trees: the percentage of tests of each type, the percentage of tests for absolute values (type 2), and the average distance of the pixel pair considered in the tests. The features used most often are appearance features, followed by stereo and then shape. For stereo features approximately 80% of the selected tests are of type 2 (absolute SSD matching cost); for appearance and shape cues, type-1 and type-2 tests are nearly chosen equally often. The mean distance between test pixel pairs (fig. 5a) is largest for the shape cue. Unsurprisingly, this indicates that shape cues are best encoded by longer-range pixel interactions. On the contrary, for stereo features short-range interactions are sufficient for discrimination.

**Comparing the different stereo features.** Table 2 shows the recognition error for each of the stereo features used with a single tree of different sizes. The three stereo features considered here (WTA disparities, mean & variance disp. and DCS slices) all lead to lower recognition errors than the no-stereo case (especially for small trees) and without much difference between them. In the remainder of the paper we use DCS slices because they are computationally cheaper than the other stereo features. Moreover, they show no overfitting for large trees. An evaluation of the number of used DCS stereo slices shows that high numbers of slices are more likely to overfit and that too few slices underperform. Six slices are found to be optimal. These are extracted at the disparities values 0, 10, 20, 30, 40, and 50 for all following experiments. Note that the tree size is critical since too small trees do not carry sufficient discrimination power and too large trees tend to overfit. Both effects (especially overfitting) are weakened when using decision forests instead of single trees, but it is still desirable to avoid very

Table 2. **Error rates** [%] using different stereo features for the 45 classes task using different numbers of nodes on the decision tree.

| number of nodes in tree | 100 | 500 | 1500 |
|---|---|---|---|
| no stereo | 12.5 | 2.9 | 2.4 |
| WTA disp. | 3.9 | 2.1 | 2.4 |
| Mean&Var disp. | 3.9 | 2.5 | 2.6 |
| 6 DCS stereo slices | 4.1 | 2.3 | 2.3 |

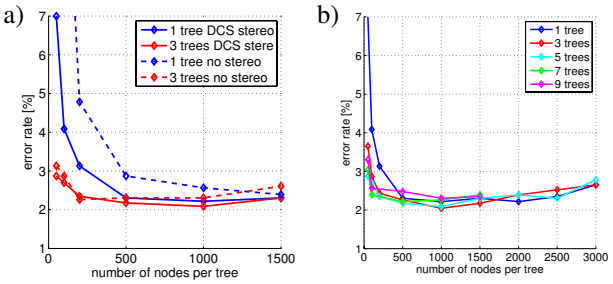



Figure 8. **Classification Error**. (a) Error curves for: different nodes per tree, different numbers of trees per forest and for both stereo and no-stereo cases. (b) Error as a function of the number nodes for different numbers of trees using 6 DCS stereo slices.

large trees to keep the runtimes low.

**Effect of forest size and number of nodes.** Fig. 8a compares the recognition error using DCS stereo slices with the case where no stereo information is used, for varying tree size and varying forest size. The trees using stereo outperform trees without stereo information by a factor of 2 for small tree sizes. As expected, the recognition error decreases with increasing number of trees and with increasing number of nodes. With 3 trees and more than 500 nodes per tree, the classification accuracy using appearance and shape only almost matches that obtained by adding stereo information. However, as pointed out earlier, large trees involve significant extra computation. Forests with many small trees tend to work as well as forests with a few large trees, allowing us to choose an optimal operational point based on our efficiency requirements. The fact that stereo features lead to a significant gain in accuracy when working with a few small trees suggests that generalisation capabilities are highly improved by using stereo features.

Furthermore it can be observed that in the experiments that use stereo features, the loss of accuracy is smaller for very high numbers of nodes per tree than the experiments that use no stereo information at all.

The surprisingly good performance of the non-stereo method with large trees might partly be due to too little variation in the evaluation setup. Next, to investigate further the influence of stereo on generalisation, we reduced the size of the training set.

Results for up to 9 trees and up to 3000 nodes per tree are presented in fig. 8b. In this application Using more than 3 trees does not lead to an improvement in accuracy for small

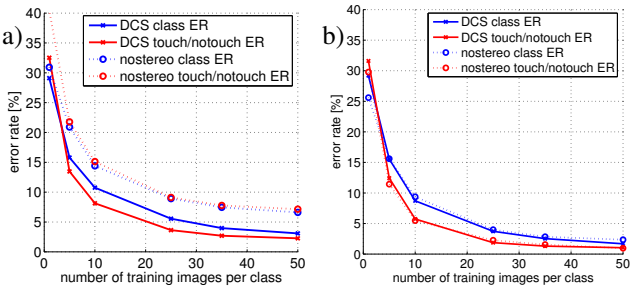



Figure 9. **Generalisation properties.** The influence of the amount of training data used on the classification performance: a) for 1 tree with 100 nodes, and b) for 1 tree with 500 nodes. The red lines denote touch/no touch error rates, and the blue lines denote error rates for the discrimination of different objects and pose classes. Dotted lines denote experiments without stereo information and solid lines denote experiments with 6 DCS stereo slices. Results are averaged over three runs.

numbers of nodes, but rather weakens the overfitting for large numbers of nodes. The fact that small trees with stereo information outperform trees that use appearance and shape only, suggests that trees that use stereo features generalise better.

**Generalisation.** In order to assess generalisation properties we reduced the amount of training data that is available for the classification of the leaf histograms while using the same trees (trained on 20% of all training data) for all experiments. Fig. 9 shows the results of these experiments for (a) one tree with 100 nodes and (b) one tree with 500 nodes. As we noticed a high variability in classification accuracy for small trees due to randomisation effects, the results were averaged over three runs.

Figure 9a shows how the use of stereo leads to a clear improvement over the experiments which use only appearance and shape for a reduced training set. This observation confirms the added generalisation brought in by the depth cues, both for the touch/no-touch discrimination task and for the class recognition task. The results with and without stereo information are very similar for larger trees (fig. 9b), which also emphasises that generalisation is improved by incorporating stereo.

**Effect of stereo on accuracy.** To assess further the improvement obtained by using stereo information as opposed to appearance and shape alone, we analysed the class and touch/no-touch discrimination separately. Tables 3a,b show confusion matrices for class discrimination and for touch/no-touch discrimination for identical setups with and without stereo features. These results were obtained using 3 trees with 100 nodes each. Additionally, the classification error rates are given for 1, 3, and 5 trees respectively. In the 1-tree case, stereo improves the class error rate (ER) by 76% relatively and the touch/no touch error rate by 58%. In the 3-tree case the class ER and the touch/no touch ER are both

Table 4. **Cost-aware tree training.** The effect of penalties for stereo features on both efficiency and classification error. The figure given here are averages over 3 runs for the case of 1 tree with 100 nodes.

| penalty | 1 | 1.2 | 2 | $\infty$ |
|---|---|---|---|---|
| stereo tests in tree | 25.0 | 8.7 | 4.3 | 0.0 |
| avg. depth stereo tests | 4.9 | 5.1 | 5.4 | 0.0 |
| st. tests on train data [%] | 13.2 | 7.5 | 4.5 | 0.0 |
| st. tests on 45 test img [%] | 9.7 | 4.9 | 0.1 | 0.0 |
| time to apply a tree [ms] | **12.4** | 6.3 | **6.2** | 5.8 |
| overall error rate [%] | **4.6** | 4.4 | **6.1** | 10.4 |

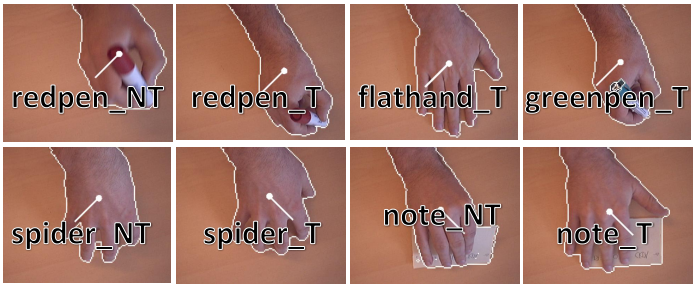



Figure 10. Example frames of hands and objects segmented and recognised correctly on a wooden background. "_T" indicates detected touch and "_NT" no touch.

improved by 16% relatively. Using 5 trees, the touch/no touch error rate is improved by 8.3% relatively. Using 3 trees, there is a significant increase in accuracy when stereo information is used both for the touch/no touch recognition and for the discrimination of the different hand poses and objects. In particular the number of touch events being missed is reduced from 3.0% to 2.1%. Also among the individual classes, several improvements are visible: *e.g.* by using stereo, the "rightangle" gesture is confused less often with the "thumb up (side view)" gesture (row 15, col 22). The same for the "fist" and "pick" gestures. Also the recognition of the "cellphone" class is improved. For the experiments with 1 tree and 100 nodes, the improvements are due to reduced confusions among the classes "pick", "point with 1 finger", and "point with 2 fingers", and among the classes "spider" and "flathand" (fig. 2c,d). Stereo information proves useful in class recognition, and even more in distinguishing touch from no-touch cases.

**The effect of cost-aware training.** Next, we assess the effects of cost-aware training. We have experimentally evaluated this for 1 tree with 100 nodes. Assuming that a non-stereo feature has a cost (penalty) of 1, we tested costs 1, 1.2 and 2.0 for stereo features. The results are presented in Table 4. As expected, a higher stereo penalty significantly reduces the number of stereo tests to be run (both in training and testing), while increasing the tree average depth. When using a penalty of 2 running times are halved (from 12.4 ms to 6.2 ms) but the classification accuracy is hardly reduced (from 4.6% to 6.1%) [5]. The smooth blending between using stereo features and using only appearance and shape allows us to choose an operating point that suits our efficiency requirements with little effect on the classification accuracy.

In the current framework, implemented in C#, the system processes 15 fps when only appearance and shape are used; and up to 10 fps when DCS stereo is also enabled. The experiments are performed on a 3.4 GHz machine.

**Pixel-wise vs on-demand stereo.** A final test we performed concerns the efficiency of stereo feature computation during training. To estimate the advantage of on-demand stereo over pre-calculating all possible stereo costs, we observe that in order to calculate 6 DCS slices for a pair of $320\times240$ stereo images, $320 \cdot 240 \cdot 6$ SSD costs have to be calculated. When applying a tree with on-demand calculation enabled, on a subset of images, in average only 41.7% of stereo matching costs were required with an average of 16% of these being required at least twice and thus coming from cache. Using on-demand stereo with memory caching reduced the number of computations to 34.4% of the total for the complete DCS slices. For WTA disparities, the effect is slightly stronger, with only 29.7% of all WTA disparities of the images being evaluated.

Finally, fig. 10 shows results of automatic segmentation, classification and touch detection obtained by our algorithm. Demo videos are available from `http::/research.microsoft.com/vision/cambridge/C-Slate/demos.htm`.

## 6. Conclusion

This paper has presented a unified algorithm for the simultaneous recognition of hand poses and object classes, and the detection of touch events. The algorithm developed here is being used to build new vision driven applications that can be controlled using natural gestures and common physical objects in real-time.

Various visual cues such as appearance, shape and depth (stereo) are combined efficiently by means of random forest learning. Stereo has been shown to improve the accuracy of both touch/no touch detection and class discrimination, with interesting generalisation properties. The structure of our learning algorithm has enabled expensive stereo features to be computed on-demand; with great reduction in the number of necessary computations and consequent increase in recognition efficiency.

Next, the use of dynamic information for the recognition of complex, time-dependent gestures promises to be interesting and challenging.

[5] the observed slight error reduction for penalty=1.2 is probably due to randomness in the training.

Table 3. **Confusion matrices** for class discrimination (top row) and touch/no touch discrimination (bottom row) using 3 trees with 100 nodes for the (a) 6 DCS stereo slice case and (b) no stereo case.

(a) DCSSTEREO

| | bigblackpen | bigbluepen | biggreenpen | bigredpen | cellphone | eraser | fistside | fist | flathand | flathandstanding | pencil | pick | point1 | point2 | rightangle | ring | scissors | sheetofpaper | spider | stickytape | stylus | thumbupside | thumbup | yellownote |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bigblackpen | 100 | | | | | | | | | | | | | | | | | | | | | | | |
| bigbluepen | | 99 | | | | | | 1 | | | | | | | | | | | | | | | | |
| biggreenpen | 1 | | 97 | | | | | | | | | | | | | | | | | | 2 | | | |
| bigredpen | | | | 100 | | | | | | | | | | | | | | | | | | | | |
| cellphone | | | | | 98 | | | | | | | 2 | | | | | | | | | | | | |
| eraser | | | | | | 100 | | | | | | | | | | | | | | | | | | |
| fistside | | | | | | | 96 | | | | | 1 | 3 | | | | | | | | | | | |
| fist | | | | | | | | 99 | | | | 1 | | | | | | | | | | | | |
| flathand | | | | | | | | | 100 | | | | | | | | | | | | | | | |
| flathandstanding | | | | | | | | | | 99 | | | | | 1 | | | | | | | | | |
| pencil | | | | | | | | | | | 100 | | | | | | | | | | | | | |
| pick | | | | | | | | | | | | 97 | | 2 | | 1 | | | | | | | | |
| point1 | | | | | | | 1 | | | | | | 97 | 2 | | | | | | | | | | |
| point2 | | | | | | | | | | | | | 1 | 99 | | | | | | | | | | |
| rightangle | | | | | | | | | | | | | | | 99 | | | | | | | 1 | | |
| ring | | | | | | | | | | | | | | | | 100 | | | | | | | | |
| scissors | | | | | | | | | | | | | | | | | 100 | | | | | | | |
| sheetofpaper | | | | | | | | | | | | | | | | | | 100 | | | | | | |
| spider | | | | | | | | | | | | | | | | 1 | | | 99 | | | | | |
| stickytape | | | | | | | | | | | | | | | | | | | | 100 | | | | |
| stylus | | | | | | | | | | | | | | | | | | | | | 100 | | | |
| thumbupside | | | | | | | | | | | | | | | | 1 | | | | | | 99 | | |
| thumbup | | | | | | | | | | | | | | | | | | | | | | | 100 | |
| yellownote | | | | | | | | | | | | | | | | | | | | | | | | 100 |

| | touch | notouch |
|---|---|---|
| touch | 97.9 | 2.1 |
| notouch | 2.5 | 97.5 |

| number of trees | 1 | 3 | 5 |
|---|---|---|---|
| class ER | 2.1 | 0.9 | 0.8 |
| touch/no touch | 3.3 | 2.3 | 2.2 |

(b) NOSTEREO

| | bigblackpen | bigbluepen | biggreenpen | bigredpen | cellphone | eraser | fistside | fist | flathand | flathandstanding | pencil | pick | point1 | point2 | rightangle | ring | scissors | sheetofpaper | spider | stickytape | stylus | thumbupside | thumbup | yellownote |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bigblackpen | 100 | | | | | | | | | | | | | | | | | | | | | | | |
| bigbluepen | | 99 | | | | | | 1 | | | | | | | | | | | | | | | | |
| biggreenpen | 1 | | 99 | | | | | | | | | | | | | | | | | | | | | |
| bigredpen | | | | 100 | | | | | | | | | | | | | | | | | | | | |
| cellphone | | | | | 94 | | | | | 2 | | | | 2 | 2 | | | | | | | | | |
| eraser | | | | | | 100 | | | | | | | | | | | | | | | | | | |
| fistside | | | | | | | 99 | | | | | | 1 | | | | | | | | | | | |
| fist | | | | | | | | 97 | | | | 3 | | | | | | | | | | | | |
| flathand | | | | | | | | | 99 | | | | | | 1 | | | | | | | | | |
| flathandstanding | | | | | | | | 1 | | 98 | | | | | | | | | 1 | | | | | |
| pencil | | | | | | | | | | | 100 | | | | | | | | | | | | | |
| pick | | | | | | | | | 1 | | | 97 | 1 | 1 | | | | | | | | | | |
| point1 | | | | | | | 1 | | | | | | 99 | | | | | | | | | | | |
| point2 | | | | | | | | | | | | | 2 | 98 | | | | | | | | | | |
| rightangle | | | | | | | | | | | | | | | 96 | | | | | | | 4 | | |
| ring | | | | | | | | | | | | 1 | | | | 99 | | | | | | | | |
| scissors | | | | | | | | | | | | | | | | | 100 | | | | | | | |
| sheetofpaper | | | | | | | | | | | | | | | | | | 100 | | | | | | |
| spider | | | | | | | | | | 1 | | | | | | | | | 99 | | | | | |
| stickytape | | | | | | | | | | | | | | | | | | | | 100 | | | | |
| stylus | | | | | | | | | | | | | | | | | | | | | 100 | | | |
| thumbupside | | | | | | | | | | | | 1 | | | | | | | | | | 99 | | |
| thumbup | | | | | | | | | | | | | | | | | | | | | | | 100 | |
| yellownote | | | | | | | | | | | | | | | | | | | | | | | | 100 |

| | touch | notouch |
|---|---|---|
| touch | 97.0 | 3.0 |
| notouch | 2.5 | 97.5 |

| number of trees | 1 | 3 | 5 |
|---|---|---|---|
| class ER | 8.8 | 1.1 | 0.8 |
| touch/no touch | 8.0 | 2.8 | 2.4 |

## References


[1] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[2] W. Buxton. Multi-touch systems that I have known and loved. http://www.billbuxton.com/multitouchOverview.html.

[3] A. Criminisi, J. Shotton, A. Blake, C. Rother, and P. H. Torr. Efficient dense stereo with occlusion by four-state dynamic programming. *IJCV*, 71(1):89–110, Jan. 2007.

[4] R. Fergus, P. Perona, and A. Zissermann. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, pages 264–271, Blacksburg, VG, June 2003.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2. Ed. 2004.

[6] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-layer segmentation of binocular stereo video. In *CVPR*, volume 2, page 1186, San Diego, CA, June 2005.

[7] V. Kolmogorov and R. Zabih. Graph cut algorithms for binocular stereo with occlusions. In N. Paragios, Y. Chen, and O. Faugeras, editors, *The Handbook of Mathematical Models in Computer Vision*. Springer, 2005.

[8] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR*, volume 2, pages 775 – 781, San Diego, CA, June 2005.

[9] R. Lockton and A. W. Fitzgibbon. Real-time gesture recognition using deterministic boosting. In *BMVC*, pages 817–826, Cardiff, UK, Sept. 2002.

[10] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*, in press, 2006.

[11] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *TPAMI*, 28(3):416–431, Mar. 2006.

[12] J. Quinlan. Bagging, boosting, and C4.5. In *National Conference on Artificial Intelligence*, pages 725–730, 1996.

[13] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans. on Graphics*, 23(3):309–314, Aug. 2004.

[14] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1–3):7–42, 2002.

[15] J. Schmidt, H. Niemann, and S. Vogt. Dense disparity maps in real-time with an application to augmented reality. In *Workshop on Applications in Computer Vision (WACV)*, pages 225–230, Orlando, Fl, USA, dec 2002.

[16] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV 2006*, *LNCS 3951*, pages 1–15, Graz, Austria, May 2006.

[17] J. Sun, N.-N. Zheng, and H.-Y. Shum. Stereo matching using belief propagation. *TPAMI*, 25(6):1–14, July 2003.

[18] C. Tomasi, A. Rafii, and I. Torunoglu. Full-size projection keyboard for handheld devices. *Communications of the ACM*, 46(7):70–75, 2003.

[19] O. Williams, A. Blake, and R. Cipolla. Sparse and semi-supervised visual mapping with the $S^3$GP. In *CVPR*, volume 1, pages 230 – 237, New York, June 2006.

[20] A. D. Wilson. PlayAnywhere: A compact tabletop computer vision system. In *Symposium on User Interface Software and Technology (UIST)*, pages 83–92, Seattle, WA, Oct. 2005.

[21] A. D. Wilson and N. Oliver. Multimodal sensing for explicit and implicit interaction. In *HCII*, 2005.

[22] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, volume 2, pages 1800 – 1807, Beijing, China, Oct. 2005.

[23] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *CVPR*, 2003.

[24] P. Yin, A. Criminisi, J. Winn, and I. Essa. Tree-based Classifiers for Bilayer Video Segmentation In *CVPR*, 2007.