# How Do We Teach Young Children New Concepts via Sketching?

Chau Thai Truong[1,2], Duy-Hung Nguyen-Huynh[1,2], and Minh-Triet Tran[1]

[1] Faculty of Information Technology, University of Science, VNU-HCM
[2] John von Neumann Institute, VNU-HCM
Ho Chi Minh city, Vietnam
chau.truong.ict@jvn.edu.vn, hung.nguyen.ict@jvn.edu.vn, tmtriet@fit.hcmus.edu.vn

**Abstract.** The authors propose a system that supports children to learn new concepts of familiar topics via their sketches on an interaction surface. The proposed system has two main subcomponents: a system of interaction surface with touch detection from depth images captured by a Kinect and a sketch recognition module based on the idea of bag-of-word model. The system provides a natural and intuitive interface for children because they can learn new concepts via sketching. With the dataset of 70 common concepts, the accuracy of the sketch recognition is 78.21% and the average response time to recognize a sketch is 0.86s. The sketch database can also be easily customized to teach new concepts to children.

**Keywords:** Human-computer interaction, sketch recognition, bag-of-word model, table-top, 3D interaction.

## 1 Introduction

Human-computer interaction (HCI) plays an important role in the evolution of computing society. Researches in the field of interaction between users and computers aim to enhance the ease of use, ergonomics, and portability as well as to save time for users. Recent researches in HCI also aim to create intelligent ambient environments that are suitable for users of different ages so that people can have much more exciting experience. Examples of these systems include the Dangerous Australians product that works as an interaction surface and shows the images of Australian species for tourists, especially children [1]; the Google Glass wearable device that operates as a mini computer and can display the information via the glass screen in front of user's eyes [2].

In the field of HCI, smart phones and tablets are becoming more and more popular. In 2013, the market share of smart phones and tablets takes up 79.7% of the worldwide smart connected device market [3]. With the appearance of these devices, children have new forms of entertainment. They spend their time playing games, drawing pictures, or surfing Internet on these devices. Therefore, we can utilize smart phones and tablets instead of books and magazines to teach kids of 10 or under new concepts in various familiar topics such as animals, plants, transportations, buildings, etc. This motivates our proposal to devise a system on these devices to support children in learning new concepts via sketching. Our system allows children to do many interesting tasks such as drawing a colorful picture just by sketching, practicing to sketch an item, or write a letter beautifully with decoration.

To deploy our system in a large area, such as a playground for children, we intergrate our learning-via-sketching system into an interaction surface using a Kinect to transform any regular surface into an interactive one [4]. Additionally, our proposed system can also be further developed to run on many common operating systems on personal computer (Windows, Linux, Mac) or on mobile device (Android, iOS, Windows Phone).

To perform the sketch recognition function, we use the dense version of SIFT to extract descriptors and apply the spatial pyramid scheme to represent the final image feature. The accuracy of our sketch recognition method is evaluated by testing on a dataset containing 70 classes, which is a part of the dataset by Eitz et.al. in [5]. Experimental results show that our sketch recognition method has the accuracy of 78.21%. Besides, with this dataset, the average response time to recognize an arbitrary sketch is 0.86s, which is acceptable for the system to run in real time.

The content of the paper is as follows. In Section 2, the authors briefly review the researches in the field of HCI and sketch recognition. Our proposed system and experimental results are presented in Section 3 and 4 respectively. Finally, conclusions and further development are discussed in Section 5.

## 2 Related Work

### 2.1 Sketch Recognition

Sketch is an intuitive image type in real life. Because of its simplicity and ease to draw, sketch is often used to represent or remember something visible. Therefore, using a computer to teach children via sketches of some common concepts can bring the excitement and motivation to children so that they can learn more effectively. Sketch recognition approaches can be divided into two groups:

**Gesture-Based Approaches:** These approaches are related to gestures that people use to draw a sketch. Akshay Bhat et.al. use entropy value of the angles caused by drawing gestures to determine whether a sketch is text or hand-drawn diagram [6]. Dean Rubine uses 13 features of a gesture to encode a sketch [7]. The main disadvantage of these methods is that the accuracy depends mainly on the order of the drawn sketches. This significantly reduces the flexibility of drawing a sketch because the image is constrained to be drawn in the order of its smaller parts.

**Free-sketch Approaches:** these methods are based on vision-based or geometric-based features of sketches to learn and classify the group to which a sketch belongs. Heeyoul Choi et.al. use Isomap kernel to compute the dissimilarity between two sketches [8]. Paul Corey et.al. integrate both gesture-based and geometric-based techniques to perform a hybrid method for sketch recognition [9]. Specifically Mathias Eitz et.al. use bag-of-features representation of a sketch and run multi-class support vector machines to classify sketches [5]. Unlike gesture-based systems, these systems do not depend on how a sketch is drawn. Therefore, we follow this trend, particularly the bag-of-feature representation to develop our method.

### 2.2 Interactive Surface

Interactive products have become more and more popular for users to communicate with digital devices and systems. Depending on the needs of users, there are many products with many different sizes, designs, materials and technologies. Medium-sized products such as tablets, ATM vending machines, smartphones are designed with small touch screens for the sake of mobility.

Many research approaches of interactive products have been presented, especially tabletops. Tobias Schwirten et.al. use laser to determine touch events in radarTOUCH system [10]. Nicolai Marquardt et.al. detect the actions of hands and fingers with or without markers/gloves via single or multiple traditional cameras [11]. Andrew Wilson uses depth data from a Kinect touch sensor to find out touched points [12]. Especially, Björn Hartmann et. al. propose eight interaction methods that are used in a working desk that supports interactions with real keyboards and mice [13]. We apply this idea to develop our interaction surface. Users interact through virtual devices projected by a projector over any relatively-flat surface and touch events are detected from depth data captured from a Kinect.

## 3 Proposed System

This section shows the main components and operations of our system. The overview of our system is presented in Sec.3.1. The touch event detection module is described in Sec.3.2. Finally, Sec.3.3 shows the sketch recognition method using bag-of-word model to recognize sketches drawn by a user on an interaction surface.

### 3.1 Overview of the System

Fig.1. demonstrates the overview and main structures of our proposed system. User draws sketches on a flat normal surface. In our system, during the interaction process, the surface and a Kinect device are held in fixed positions. A single computer collects depth data images captured from a Kinect to filter the drawn sketches and classify those sketches. The image of sketches and the result class name are shown on the surface via a projector. These functions are divided into two parts: touch detection and sketch recognition.

***Touch detection:*** A Kinect device operates continuously to capture depth images of the surface. Depth information is processed by the computer and touched points are detected from this process. Then, appropriate touch events are generated. Finally, the projector shows the images of sketches.

***Sketch recognition:*** this subcomponent is the main function of our system. When a user draws a virtual image of a sketch, the touch detection subcomponent shows the real image while the sketch recognition subcomponent classifies the class to which this sketch belongs. With the recognition result, a child can learn new concepts by doing the following tasks:

– Drawing and completing a picture by the corresponding image returned after each time a sketch is drawn. Corresponding images can be retrieved from the local storage of the system or from the Internet via web APIs. The returned image can then be resized and moved to the appropriate place (see Fig.2. (b), (c), (d), and (e)).
– Practicing how to sketch beautifully an item (a tree, a plane, a car…) or a letter. To support this task, the system lists many predefined concepts in the order of the similarity with the sketch.

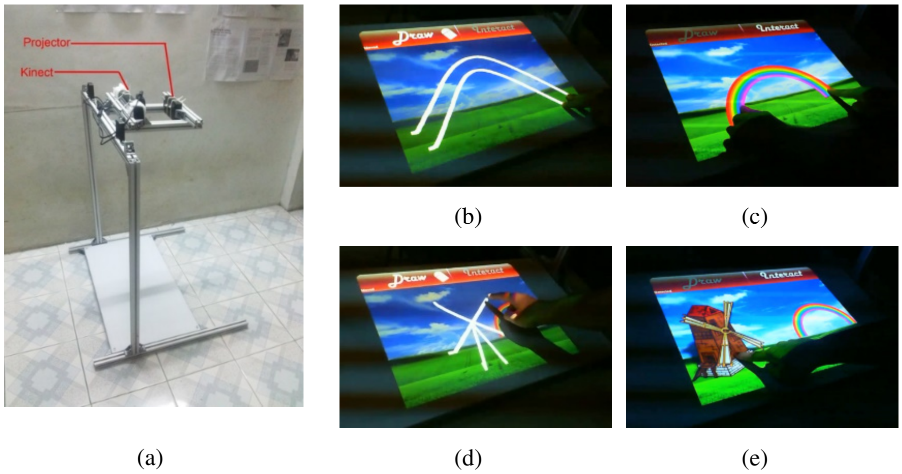


**Fig. 1.** Drawing sketches with a touch system using a Kinect and a projector (a) Overview of the system (b) A user draws a sketch on the touch surface (c) This sketch is recognized and replaced by the real image (d) More sketches are drawn. (e) The drawing is completed gradually by many images recognized from the sketches.

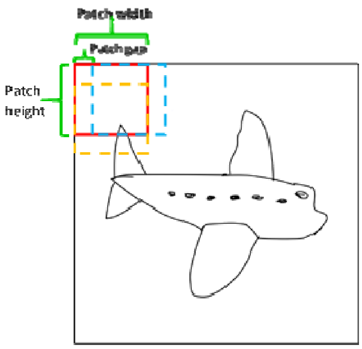
## 3.2 Touch Event Detection

In this module, we reuse our touch event detection module proposed in [4] to perform the interaction process. In reality, this module can be substituted by any device that has touch function such as smart phones or tablets. The sketch recognition module is designed as a program that can be modified to run on common operating systems such as Windows, Android, or iOS.

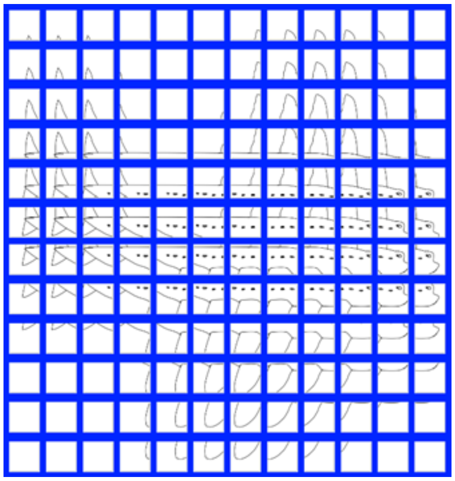
## 3.3 Sketch Recognition Method

In section 3.3, we present the process to recognize sketches. The main structure is based on the concept of bag-of-word model [14]. The main steps of this method are densely sampling on images, building visual word dictionary or codebook, descriptor vector quantization, building final image descriptor based on the codebook and using Support Vector Machines learning model to train and test data.

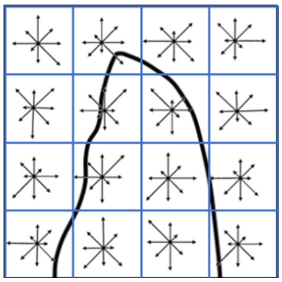
*Densely sampling on images*



(a)



(b)



(c)

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| $v_1$ | $v_2$ | $v_3$ | $\cdots$ | $v_{N-1}$ | $v_N$ |

$N$ = number of orientations × number of cells

(d)

**Fig. 2.** Computing patch descriptors. (a) Example of 3 overlapping patches with different border colors (red, blue, orange); (b) All patches in an image; (c) The gradient orientations and magnitudes in the cells of each patch; (d) The total descriptor vector from (c).

To represent an image as a feature vector, the common approach is to locate regions of interest, or keypoints in the image, with different keypoint detectors, among which SIFT detector [15] is widely used. However, in sketch images, there are very few keypoints as an image mostly contains strokes and curves, not textures. Therefore, instead of using regular SIFT detector, we calculate the dense SIFT descriptor, which sample in all regions of an image to generate overlapping 16-by-16 pixels sub-images or patches. Each patch is subdivided equally into non-overlapping $4 \times 4$ cells.

Two adjacent patches have a gap of 4 pixels horizontally or vertically. The number of patches in each image is:

$$\frac{(width_{image} - width_{patch})}{patch\ gap} \cdot \frac{(height_{image} - height_{patch})}{patch\ gap}.$$

The descriptor vector is the gradient orientation and amplitude map of that $16 \times 16$ pixel patch. In each cell, the image orientation previously computed is quantized into 8 bins depicting 8 different gradient directions over the range $0 - 360^o$ resulting in a histogram of 8 orientations weighted by the pixel's gradient amplitude and the pixel's distance from the center of the patch. Finally, histograms of $4 \times 4 = 16$ cells are concatenated to make a final descriptor of the patch as a single vector of 128 dimensions (16 cells $\times$ 8 dimensions). The descriptor is normalized afterwards. This process is demonstrated in Fig.2.

*Visual dictionary construction*
The main purpose of this step is to find the patch patterns that appear most frequently in the dataset and then calculate the frequent of each pattern in the dataset. These patches form the visual dictionary of the dataset or the codebook. The training step, testing step as well as the final product use this codebook to quantize the patch descriptors of input images. We use a common algorithm for clustering problems – the *k*-means algorithm.

*Vector quantization*
The purpose of this step is to quantize the large set of the visual descriptors to a smaller set of visual words, i.e. consider an image in terms of visual words rather than visual descriptors. Each *D*-dimensional descriptor is encoded to a *K*-dimensional vector that belongs to the space of the codebook. The main criterion to encode a descriptor is the Euclidean distance between that descriptor and each visual word in the codebook. We use the soft vector quantization method for each descriptor *V*. Let distance $D_i$ be the distance from the $i^{th}$ visual word to *V*, we get *k* nearest visual words and assign the following Gaussian function values of $D_i$'s to the corresponding dimensions of the code vector:

$$d_i = e^{\frac{-D_i^2}{\sigma^2}}$$

Therefore, the weight for each dimension of a patch's code vector depends mainly on the similarity between that patch and each visual word.

*Using spatial pyramid scheme and average pooling to build an image descriptor*
Pyramid matching [16] is a scheme used to estimate the approximate similarity between two sets of feature vectors, i.e. the distance between these two set in the same feature space. The idea of this scheme is to combine the representations of smaller set partitions in the feature space at multiple resolutions and compare resulting multi-resolution representations of sets. More specially, this scheme makes a sequence of increasingly coarser grids in the feature space and takes a weighted sum of the number of matches for each resolution. Two points are considered as a match if they belong to the same grid cell at the same resolution. Matches found at finer resolutions

are weighted more because those matches are more highly expected than matches at coarser resolutions are.

After the quantization step, for each image $s^{(t)}$, we achieve the set of codes $C^{(t)} = \left[c_1^{(t)}, \dots, c_N^{(t)}\right]$ corresponding to the set of descriptors $V^{(t)} = \left[c_1^{(t)}, \dots, c_N^{(t)}\right]$ where *N* is the number of descriptors for $s^{(t)}$. The histogram of each spatial region of the spatial pyramid is calculated by the average pooling on the codes as follows:

$$z_j^{(t)} = \frac{1}{Q}\sum_{i=1}^{Q} c_{ij}^{(t)}, j = 1, \dots, K$$

where *Q* is the number of descriptors in that region of $s^{(t)}$ and *K* is the dimensions of image code $c_i^{(t)}$. The final image descriptor is formed by concatenating the histograms of all spatial regions and has the number of dimensions as:

$$M = K\sum_{l=1}^{L} 4^l = K\frac{1}{3}(4^{L+1} - 1)$$

where *L* is the number of spatial pyramid levels.

*Training and testing steps using Support Vector Machines (SVM) model*

The two main processes in this method are training and testing. In the training process, the final sketch descriptors of training images are learned by SVM. The testing images are then verified on the trained SVM model to determine the accuracy of the system. In real application, the trained SVM models are also used to predict the class of an input sketch from a user.

## 4 Experimental Results

Experiments of the touch detection accuracy are already mentioned in [4]. In this section, we present three experiments. The first experiment in Sec.4.1 determines the accuracy of the sketch recognition module. The feasibility of the system to run in real time is mentioned in Sec.4.2. Finally, Sec. 4.3 shows the efficiency of the system in teaching children. The first two experiments are performed on the system using CPU core i7 2.2Ghz with 8GB RAM.

### 4.1 Sketch Recognition Accuracy

In the first stage of our system, we conduct the experiment of the sketch recognition module. We reuse a part of the dataset published by Mathias Eitz in SIGGRAPH 2012 [5]. Our dataset consists of 70 classes with total 5600 1111-by-1111 images of many familiar topics. Each class contains 80 images. Some images in our dataset and their corresponding real images are demonstrated in Fig.3. We resize all images to 128×128 pixels and extract 16-by-16 patches and the numbers of overlapping pixels

8. Each class is divided into 2 parts for training and testing. The training part is also divided into 4 equal parts for cross-validation (CV) testing.
Since the images in each class can be very different (see Fig.4), we find the class of a test image by voting from 10 different SVM models. The final predicted class is the one that has the largest sum of probabilistic estimate scores of 10 models. Each SVM model is chosen randomly from the CV parts as well as the final training part.

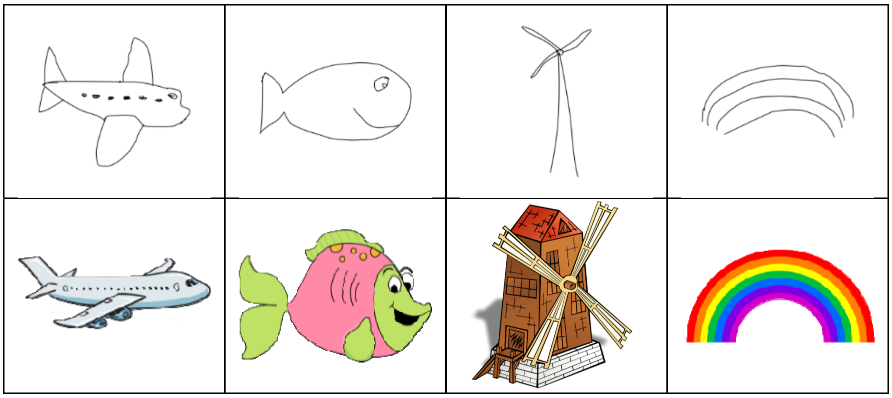


**Fig. 3.** Some sketch images in the dataset (1st row) and corresponding images (2nd row)

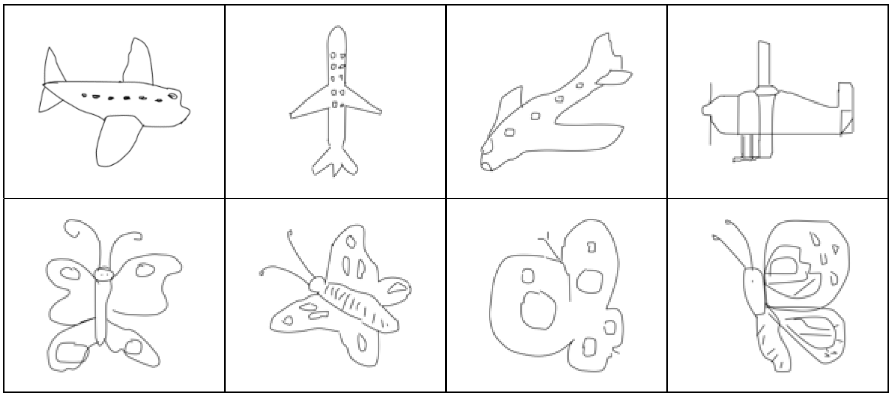


**Fig. 4.** Some classes that have very different images: the plane (1st row) and the butterfly (2nd row)

To build the visual dictionary, we use the $k$-means algorithm with 300 clusters. The final image descriptors are calculated at 3 spatial pyramid levels, which results in $300 \times (4^0 + 4^1 + 4^2) = 6300$ dimensions for each descriptor. The cross-validation accuracy of $k$-fold test (with $k$=4) is presented in Fig.5.

After the cross-validation test, we train the system with the whole training set and evaluate the accuracy of sketch recognition function with the test set. The accuracy for each of 70 classes is demonstrated inFig.6.. Although there are still 4 classes with low accuracy of less than 50%, most of the classes can achieve high accuracy and the average accuracy of our system is 78.21%. The confusion matrix for 70 classes is illustrated in Fig.7.

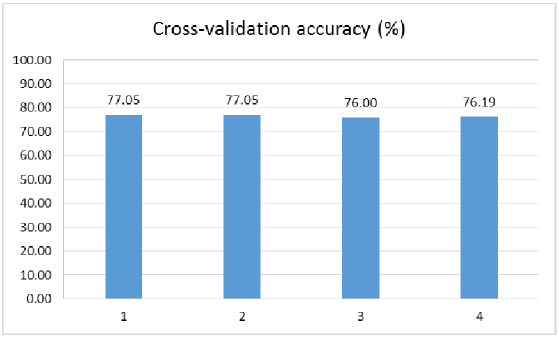


**Fig. 5.** The cross validation accuracy of *k*-fold test (with *k*=4)

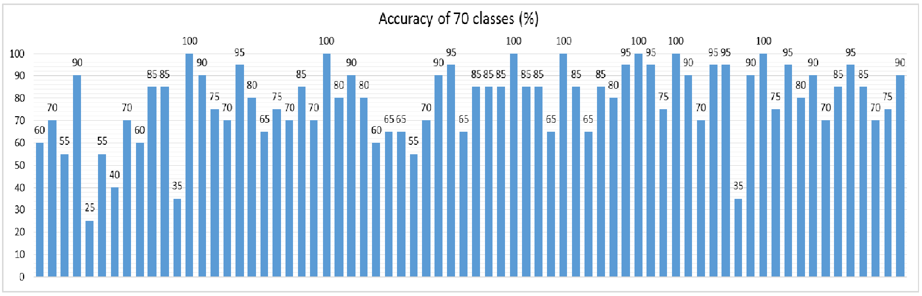


**Fig. 6.** The accuracy of 70 classes in the total test. There are 4 difficult classes with the accuracy below 50%.

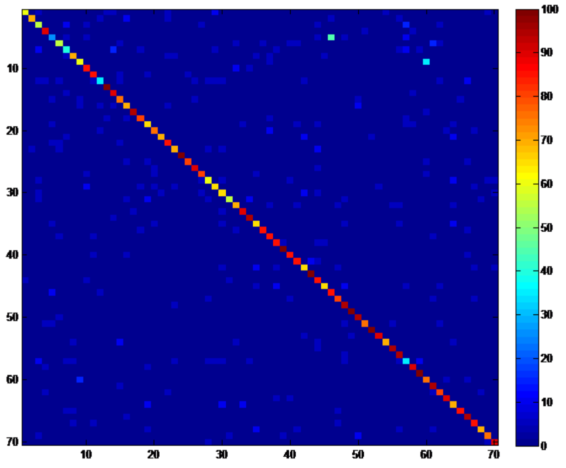


**Fig. 7.** Confusion matrix for 70 classes

### 4.2 Feasibility of the System to Recognize the Sketches in Real Time

We test the feasibility of the system to run in real time by taking the average running time of test images in the final test (after the CV test). The running time of the stages are shown in Table 1. This experiment is conducted in the same system configuration and parameters as in Sec.4.1.

**Table 1.** The average running time of sketch recognition

| | |
|---|---|
| Calculating patch descriptors | 0.458s |
| Calculating final image descriptor | 0.139s |
| Testing on SVM models | 0.263s |
| **Total** | **0.860s** |

### 4.3 Efficiency of the System in Teaching Children

The second stage of our proposed system is to evaluate the attractiveness and efficiency in teaching new concepts for young children. 20 young Vietnamese children of the ages from 6 to 8 are divided into two groups to learn English words of common concepts. Children in the first group learn with traditional teaching method and printed books of pictures while those in the second group learn by playing with our system and sketches. From the observation, kids in the second group can quickly remember new words and can save up to 30% time to learn words in a category in comparison with those in the first group. Furthermore, children in the second group spend 50-70% more time in studying than those in the first group and can study more concepts.

## 5 Conclusions

In this paper, we propose a system that can be intergrated to an interactive product to teach children new concepts via their sketch images. The system can be deployed on our interaction surface using Kinect or on any other mobile device.

To perform the sketch recognition module, we extract the dense SIFT descriptors that is computed by densely sampling all regions of a sketch; the idea of bag-of-word model and spatial pyramid scheme to represent the image feature. Additionally, we also use Support Vector Machine for the training and testing steps and real application.

Experiments of this system are about the accuracy of a large set of sketch images with 70 different types and the time respond that enables the feasibility to run in real time. Results from the experiment show that our system meets the requirement to be a real-time system that runs with high accuracy. The dataset can also be changed to diversify the concepts that can be taught to children.

Further developments of this system include the system functions such as: sharing on social networks, more intuitive interfaces; and the sketch recognition algorithms such as: more sketch types, instant feedback from users to improve the accuracy.

## References

1. Lightwell, `http://lightwell.com.au/projects/dangerous-australians/`
2. Fig. 8, `http://www.google.com/glass/start/`
3. Enterprise Irregulars, `http://www.enterpriseirregulars.com/66218/idc-87-connected-devices-2017-will-tablets-smartphones/`
4. Truong, C.T., Nguyen-Huynh, D.-H., Tran, M.-T., Duong, A.-D.: Collaborative smart virtual keyboard with word predicting function. In: Kurosu, M. (ed.) Human-Computer Interaction, Part IV, HCII 2013. LNCS, vol. 8007, pp. 513–522. Springer, Heidelberg (2013)
5. Eitz, M., Hays, J., Alexa, M.: How do humans sketch objects? ACM Transactions on Graphics (TOG) - SIGGRAPH 31(4) (July 2012)
6. Bhat, A., Hammond, T.: Using entropy to distinguish shape versus text in hand-drawn diagrams. In: IJCAI 2009 Proceedings of the 21st International Jont Conference on Artifical Intelligence, San Francisco (2009)
7. Rubine, D.: Specifying gestures by example. In: SIGGRAPH (July 1991)
8. Hammond, T., Heeyoul, C.: Sketch recognition based on manifold learning. In: AAAI 2008 Proceedings of the 23rd National Conference on Artificial Intelligence (2008)
9. Corey, P., Hammond, T.: GLADDER: combining gesture and geometric sketch recognition. In: AAAI 2008 Proceedings of the 23rd National Conference on Artificial Intelligence (2008)
10. Schwirten, T.: radarTOUCH., `http://www.radar-touch.com/`
11. Marquardt, N., Kiemer, J., Greenberg, S.: What caused that touch?: Expressive interaction with a surface through fiduciary-tagged gloves. In: ITS 2010 ACM International Conference on Interactive Tabletops and Surfaces (2010)
12. Andrew, W.D.: Using a depth camera as a touch sensor. In: ITS 2010 ACM International Conference on Interactive Tabletops and Surface (2010)
13. Hartmann, B., Morris, M.R., Benko, H., Wilson, D.A.: Augmenting interactive tables with mice & keyboards. In: UIST 2009 Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology (2009)
14. Li, F.-F., Fergus, R., Torralba, A.: Recognizing and Learning Object Categories
15. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
16. Grauman, K., Darrell, T.: The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In: ICCV 2005 Proceedings of the Tenth IEEE International Conference on Computer Vision (2005)
17. Lin, L., Luo, P., Chen, X., Zeng, K.: Representing and recognizing objects with massive local image patches. Pattern Recognition 45(1), 231–240 (2012)
18. Cortes, C., Vapnik, V.: Support-Vector Networks. Machine Learning 20(3), 273–297 (1995)
19. Tuddenham, P., Davies, I., Robinson, P.: WebSurface: An interface for co-located collaborative information gathering. In: ITS 2009 Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (2009)
20. Klinkhammer, D., Nitsche, M., Specht, M., Reiterer, H.: Adaptive personal territories for co-located tabletop interaction in a museum setting. In: ITS 2011 Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (2011)
21. Dippon, A., Echtler, F., Klinker, G.: Multi-touch Table as Conventional Input Device. In: Stephanidis, C. (ed.) Posters, Part II, HCII 2011. CCIS, vol. 174, pp. 237–241. Springer, Heidelberg (2011)

22. Grauman, K., Darrell, T.: The Pyramid Match Kernel: Efficient Learning with Sets of Features. The Journal of Machine Learning Research 8, 725–760 (2007)
23. Lazebnik, S., Schmid, C., Ponce, J.: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In: CVPR 2006 Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2006)
24. Bosch, A., Zisserman, A., Muñoz, X.: Scene classification via pLSA. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 517–530. Springer, Heidelberg (2006)
25. Morris, M.R., Lombardo, J., Wigdor, D.: WeSearch: Supporting collaborative search and sensemaking on a tabletop display. In: CSCW 2010 Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work (2010)