MVA2007 IAPR Conference on Machine Vision Applications, May 16-18, 2007, Tokyo, JAPAN

Semi-supervised Incremental Learning of Manipulative Tasks

Zhe Li, Sven Wachsmuth, Jannik Fritsch[1], and Gerhard Sagerer
Applied Computer Science, Faculty of Technology
Bielefeld University, D-33594 Bielefeld, Germany
{lizhe, swachsmu, jannik, sagerer}@techfak.uni-bielefeld.de

Abstract

For a social robot, the ability of learning tasks via human demonstration is very crucial. But most current approaches suffer from either the demanding of the huge amount of labeled training data, or the limited recognition cabability caused by very domain-specific modeling. This paper puts forward a semi-supervised incremental strategy for the robot to learn the manipulative tasks performed by the user. The task models are extended Markov models, taking a set of pre-learned object-specific manipulative primitives as basic states. They can be initialized with few labeled data, and updated continously when new unlabeled data is available. Furthermore, the system also has the capability to reject unlabeled observation as unseen tasks and detect a new task model from a group of them. Thus, using this strategy, the robot only needs human teaching at every beginning, then elaborate the learned tasks, and even extend task knowledge by its own observation. The experimental results in an office environment show the applicability of this approach.

1 Introduction

Recently, robots that cooperate with humans are receiving more and more interest in the robotics as well as in the computer vision research community. It is an interesting application field that is expected to have a high future market potential. A couple of global and also mid-sized companies have come up with quite sophisticated robotic platforms that are designed for human-robot interaction. The ultimate goal is to place some robotic assistant or companion in the regular home environment of people, who would be able to communicate with the robot in a human-like fashion.

For human-centered robots, it is very important to achieve the awareness of the state of the user. The visual recognition of human actions provides a non-intrusive way for such kind of communication between a human and the robot, especially in passive, more observational situations. In the near past, much work has been done in this area [11]. Starting from recognizing short-term, predefined commands for the human-robot interaction, the research focus shifts to more complicate, semantically elaborated human gestures. In terms of Bobick's taxonomy of *movements, activities, and actions* [1] this can be characterized as a shift from movements to more structured activities. In this regard, object manipulations[2] add an additional complexity because the hand trajectory needs to be interpreted in relation to the manipulated object. Due to Bobick this kind of context characterizes *actions*.

Our focus is the vision-based learning and recognition of object manipulation tasks, which are sequential object manipulations. For example, "take a cup" and "take a tea can" should not only be recognized as two independent manipulative actions but also as an entity with the underlying human intention "prepare tea". But the realization of such a vision system is difficult. The reasons are twofold. Firstly, in the low-level image processing, the trajectory-based gesture recognition suffers from the notorious segmentation ambiguity and spatio-temporal variability. The approaches which are based on neat state descriptions also meet difficulties for robust object detection and tracking, especially when the mutual occlusion between the hand and the object happens during manipulation. On the higher level, because of the generality of the definition of "task", there does not exist any universal task model. In the approaches using semantic models, the prior knowledge of a certain kind of task must be well studied and summarized as a task-specific grammar, which limits the application of such model to a small and well-defined area. On the other side, the demand for a huge amount of training data prohibits the use of probabilistic approaches in learning large-scale tasks.

In this paper, we put our attention on the higher interpretation level and propose an online task-learning strategy based on prelearned object-specific manipulative primitives. The system is based on a two-layered structure (see Figure 1). In the top layer, the manipulative tasks are modeled by an extended Markov process. It uses the detected primitive output from the lower layer as states. In order to have comparable similarity measurements between the primitive sequences with different lengths, a random model is used to scale the probability. The task model can be initialized with few labeled data and updated incrementally when new unlabeled data becomes avaliable. Moreover, with the possibility to reject, the system is able to detect the unseen tasks during learning process and build up new task models. These two capabilities – to reject unmodeled sequences and to learn from unlabeled data – are essential for a human-like interaction style with a robot, which does not separate between a learning mode and a recognition mode. Learning of new task models can take place during normal interaction. If the robot does not understand a specific action sequence the human interaction partner can instantly react on it by repeating the sequence, so that the robot is able to establish a new model for it.

This paper is organized as follows: In the next section, the related work on task learning is discussed. Section 3 shows the system architecture, scratches the primitive detection, and presents the task model and the decision rule.

[1] J. Fritsch is now with the Honda Research Institute Europe GmbH in Offenbach, Germany.

[2] Nehaniv refers to them as *manipulative gestures* [8].

The learning strategy is described in detail in Section 4. Section 5 is dedicated to system evaluation. After that, the paper is closed by a summery.

2 Related work

Over the past years, a bunch of work has been carried out to learn and recognize the long-term human actions. In order to recognize the Japanese tea ceremony, a Stochastic Context Free Grammar (SCFG) is build up by Yamamoto according to the performing rules in the ceremony [12]. In Chan's work, a simple feature vector is used for modeling the interaction primitive, e.g. *approach*. The transition of the semantic primitives are modeled by HMM [2]. These semantic approaches can only be learned with prior structural knowledge given by a human. Thus, as good as they are, they are only applicable in pre-defined domains.

In the smart homes project, the inhabitant actions of the user are predicted according to task models, which are Markov models that have been generated from an unsupervised clustering of the data recorded in 1250 days [10]. For a household robot, it may be feasible to group similar activities after observing a certain amount of tasks for many times. But more naturally, users will expect that the robot is able to learn some tasks from only a few demonstrations. Pardowitz developed a task precedence graph (TPG) for such purpose. The TPG could be initialized as a most restrictive model with one instance and incrementally generalized by including more logical transition possibilities [9].

Our scenario prohibits us to use a large amount of data as Rao and Cook [10]. Similar to Pardowitz et al. [9], our approach chooses an incremental way to learn the Markov model of the task. But what goes beyond it is that our approach has a probabilistic similarity measure of the unlabeled sequence given a learned model, which leads to the capability of the rejection of unseen tasks and group them into new tasks unsupervisedly.

3 System Architecture

In our definition, the manipulative task has two semantic layers. The bottom layer consists of the object-specific manipulative primitives. Each object has its own set of manipulative primitives because we argue that different object types serve different manipulative functions and even manipulations with the same functional meaning are performed differently on different objects. The top layer is used for representing the manipulative task, which are modeled by typical transitions between certain manipulative primitives. The system architecture is shown in Figure 1. From bottom to top, a processing thread is created for each detected object. Thus, the feature computation and HMM-based recognition are performed in parallel for different objects. The task level takes all detected primitives from different threads as input. The task decision is based on matching the total sequence with the different task models. For recognition, a top-down process utilizes the task-level prediction of possible primitives for a task-driven attention filter on the low-level image processing.

3.1 The manipulative primitive detection

The manipulative gesture is different to the face-to-face interactional gesture because it reflects the interaction between the human hand and the objects, not the pure hand

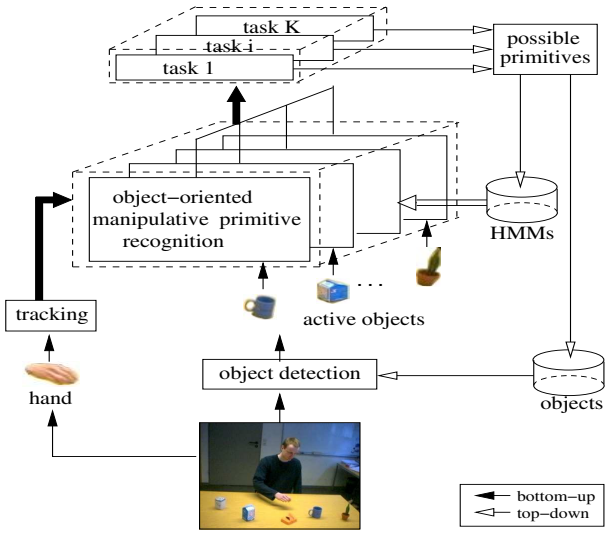

Figure 1: System architecture

movement with a meaningful trajectory. The hand is detected in a color image sequence by an adaptive skin-color segmentation algorithm (see [3] for detail) and tracked over time using Kalman filtering. The hand observation $\mathbf{o}_t^{\text{hand}}$ is represented by the hand position $(h_x, h_y)_t$ at time t. In order to avoid occlusion problems with interacting hands, we use an object recognizer based on the Scale-invariant Feature Transform (SIFT) [6] on the static scene. Then, object-dependent primitive actions are purely defined based on the hand trajectory that approaches an object instead of considering the object in the hand as a context. If a moved object is applied to another object, the second object defines the object context. The observation vector of a detected object $\mathbf{o}_i^{\text{obj}}$ contains its position (o_x, o_y), a unique identifier (ID) for each different object type in the scene and its height o_h and width o_w. So the object observation vector for a single object is:

$$\mathbf{o}_i^{\text{obj}} = (o_x, o_y, \text{ID}, o_h, o_w). \quad (1)$$

As mentioned before, a processing thread is created for each detected object. In the processing thread for object i. The five-dimensional feature vector $\mathbf{v}_f$ which represents the interaction of the hand and the object is calculated from $\mathbf{o}^{\text{hand}}$ and $\mathbf{o}_i^{\text{obj}}$. It contains the features: magnitude of hand speed v, change of the hand speed Δv, change of speed direction $\Delta\alpha$, distance r between the object and the operative hand scaled by object size, as well as the angle γ of the line connecting object and hand relative to the direction of the hand motion.

$$\mathbf{v}_f = (v, \Delta v, \Delta\alpha, r, \gamma) \quad (2)$$

The features are invariant with regard to translations, minor scale, and small rotations.

In the object manipulations, we argue that the more meaningful hand movements happens in the vicinity of the object. In our setting it is centered in the middle of the detected object and limited by the ratio β of its radius and the object size. Because of possible occlusions during the manipulation and the local uncertainty while moving an object, we concentrate on the semantic information of typical trajectories in the vicinity of a static objects. These are termed object-oriented manipulative primitives, e.g.,

"take a cup". They are modeled by HMMs. Different to the normal parameter set (the initial, transitional, and observational probabilities of a HMM), terminal probabilities are added. Each of them reflects the probability of a HMM to terminate given a hidden state and is calculated similar to the initial probabilities, except using the last states.

In order to spot the primitive from the trajectories, a PF called Sampling Importance Resampling (SIR) is used (better known as CONDENSATION introduced by Isard and Blake [4]). The resampling step in the particle propagation is able to adapt the starting point of the model matching process if the beginning of the primitive does not match the beginning of the segment. The terminal probability gives an estimation of the primitive's ending point. This combination to a certain extent solves the problem of the forward-backward algorithm which needs a clear segmentation of the pattern. Therefore the manipulative primitives can be detected from a long-term observation. The details of PF based HMM matching can be found in [5].

3.2 Model of the task

The manipulative tasks are modeled as a first-level Markovian process which is the same as Moore's definition [7]. Although this assumption violates certain domain dependencies, it is an efficient and practical way to deal with task knowledge. All the tasks share a set of possible manipulative primitives. The model Λ_i for a manipulative task i contains the transition matrix A_i, the initial probability Π_i, the terminal probability E_i, and a threshold Th_i. Suppose the result from the manipulative primitive recognition is the sequence P_o. To calculate the acceptance of a task $\Lambda_i = (\Pi_i, A_i, E_i, \mathrm{Th}_i)$, a random model Λ_r is used, which has no associated threshold and is learned from all the training data from different tasks. The similarity of the sequence and a task model $s(i, P_o)$ is calculated as:

$$s(i, P_o) = \log(\frac{p\{P_o|\Pi_i, A_i, E_i\}}{p\{P_o|\Lambda_r\}}) \quad (3)$$

Taking the possible rejection into consideration, the task decision $d(P_o)$for recognition is:

$$d(P_o) = \begin{cases} \arg\max_i(s(i, P_o)|s(i, P_o) > \mathrm{Th}_i) \\ \text{null} \end{cases} \quad (4)$$

In the following section, the learning process of the model will be described.

4 Task Learning

In the task learning, it is supposed that the robot has some basic recognition capabilities. That means the models of object-specific manipulative primitives are trained already. The primitive detector in the lower layer turns the observation of a manipulative task to a sequences of primitives and feeds them into the learning process as input. What's worth mentioning is that the primitive detection is not perfect. Consequently, both the labeled and unlabeled training data for task learning could have possible deletions, insertions, and substitutions in them.

The semi-supervised learning process is shown as pseudocode in Figure 2. It starts from a small set of labeled sequences. Suppose there are n different labels, n manipulative task models will be constructed. The model for

```
/* Supervised Learning */
1. construct Λ_{1...n} and Λ_r from labeled data
2. find P_i^th for each task model Λ_i (i = 1...n)

/* Unsupervised Learning */
for each new unlabeled sequence P_m do
  1. update random model Λ_r
  2. compute d(P_m) (see Eq. 4)
  if d(P_m) = j then
    update task model j
  else {d(P_m) = null}
    i* = arg max_i(s(i, P_m))
    if s(i*, P_m) ≥ Th_o then
      1. update task model Λ_{i*}
      2. choose new P_{i*}^th
    else {s(j*, P_m) < Th_o}
      /* reject P_m as unseen task */
      insert P_m into buffer
      if buffer is full then
        1. build a task model Λ_{n+1}
        2. k* = arg min_{k=1...l_b}(s(n+1, P_k^buffer))
        if s(n+1, P_{k*}^buffer) ≥ Th_o then
          1. take Λ_{n+1} as a valid task model
          2. n ← n+1
        else {s(n+1, P_{k*}^buffer) < Th_o}
          delete the sequence P_{k*}^buffer
```

Figure 2: Pseudocode of semi-supervised task learning

task i that is learned from u labeled data is represented as $\Lambda_{i,u}$. $\tilde{A}_{i,u}$ is the matrix recording the counts of all transitions between the primitives which happened in the labeled data. In order to account for unseen events, we use a simple "adding one" method. Consequently, the matrix $A_{i,u}$ is the resulting matrix $1 + \tilde{A}_{i,u}$ normalized in column. $\Pi_{i,u}$, $E_{i,u}$ and the parameters in Λ_r are computed in the same way. But the random model Λ_r is learned using all the sequences from different tasks. Given Λ_r and Λ_i, the similarity measurements of the labeled sequences from one task can be calculated according to Eq. 3. Then, $\mathrm{Th}_{i,u}$ is set as the minimum of them and the corresponding sequence is saved in memory as the instance of worst matching P_i^{th}.

When an unlabeled sequence P_m is perceived, the random model is updated first. As a consequence, the $\mathrm{Th}_{i,u}$ is renewed to $\hat{\mathrm{Th}}_{i,u}$. Afterwards, the decision of P_m is made based on Eq. 4. There are two possibilities. Case 1, the sequence is sorted into task j. The task model Λ_j will be updated. Case 2, the decision is null which means no $s(i, P_m)$ greater than $\hat{\mathrm{Th}}_{i,u}$. Because this decision is only made according to the current task models, in order to group similar sequences and update a task model during the learning process, a general lowest matching threshold Th_o is introduced, which defines the allowed loosest match between the sequence and its task model. So when the process entered case 2, the best matching task model $i^* = \arg\max_i(s(i))$ is computed. Dependent of the comparison of $s(i^*, P_m)$ and Th_o, the case 2 is divided into two subcases. Case 2.1, if $s(i^*, P_m)$ is greater than Th_o, the sequence is labeled as i^*. The task model Λ_{i^*} will be updated and the worst matching sequence of this model will be evaluated again. The case 2.2 means the all $s(i, P_m)$ are less than Th_o. This means it is far away from known models and just rejected as an unseen task.

A new sequence is labeled when the process goes into case 1 or case 2.1. The corresponding task model will be updated. Suppose $\tilde{A}_{i,\text{one}}$ represents the transition of the primitives in the new sequence, $A_{i,u+1}$ is the normalized $1+\tilde{A}_{i,u}+\tilde{A}_{i,\text{one}}$. The update of Π_i, E_i is in the same vein.

When the process has entered the case 2.1, the task threshold will also be updated. The similarity measurements of the sequence in memory and the new observation given the updated model are calculated again, the task threshold $\text{Th}_{i,u+1}$ and the sequence in memory will be set to the smaller matching result and the corresponding sequence.

Because of the task rejection, this learning strategy could be extended to learn new tasks unsupervisedly based on current task models. Considering also the possible false negative task rejection, a buffer with a pre-defined length l_b is used to save the rejected sequences. Once the buffer is full, a temporary model is built up based on the sequences in it. If all similarity measurements between the sequences and the new model are above the Th_o, this model is taken as a valid learned new task model and the buffer is emptied for next new model. Otherwise, the sequence with lowest similarity will be deleted from the buffer. Then, the next task rejection refills it and triggers the process again.

5 Experiments

In our experiment, a scenario in an office environment is set as the image in Figure 3. A person sits behind a table and manipulates the objects that are located on it. She or he is assumed to perform one of three different manipulation tasks: (1) *water plant*: take cup, water plant, put cup; (2) *prepare tea*: consists of take/put cup, take tea can, pour tea into cup, put tea can; (3) *prepare coffee*: consists of take/put cup, take milk/sugar, pour milk/take sugar into cup, put milk. In the experiment, each task is performed 4-5 times by 8 different persons resulting in 36 sequences for each task and a total of 108 sequences. The images are recorded with a resolution of 320x240 pixels and with a frame-rate of 15 images per second. The object recognition results have been labeled because the evaluation experiment should concentrate on the performance of the task learning.

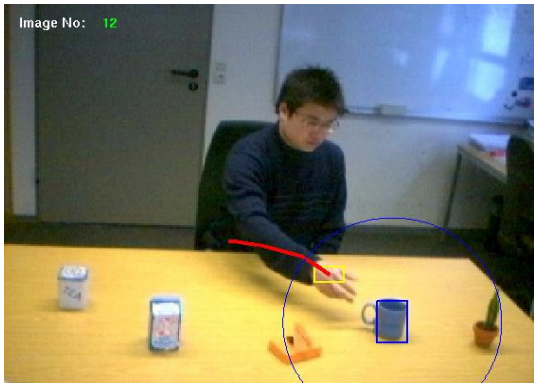

Figure 3: The office scenario used in the experiment.

5.1 Evaluation of primitive detection

To test the performance of the object-oriented manipulative primitive detection, the 108 whole task sequences are randomly divided into a training set of 60, and a test set of 48 sequences. Table 1 shows the detection results of the object-specific primitives. Because the primitives are detected from longer time observation, we use the primitive error rate (PER) defined as PER=(#Substitution + #Insertion + #Deletion)/#Truth to present the quality of the detection. Considering the random initialization of HMM,

Table 1: The detection of the manipulative primitives.

Object	Primitive	Num. of Truth	PER
tea	take	16	27.5 ± 9.4
	put	16	6.8 ± 1.9
milk	take	14	33.5 ± 11.6
	put	14	12.8 ± 5.6
sugar	take	13	53.1 ± 19.6
cup	take	48	7.7 ± 3.5
	put	42	15.6 ± 2.8
	pour	43	21.5 ± 6.9
plant	water	16	38.1 ± 24.2

the Baum-Welch algorithm ran 10 times to achieve the standard deviations. Though the results are not perfect, they provide a good chance to evaluate the task learning under noisy input.

5.2 Manipulative task learning

The second evaluation assesses the semi-supervised learning of the manipulative tasks. For every task, the whole training set contains the primitive sequences detected from 20 observations, 16 sequences are testing data. Figure 4 shows the results of recognition rate of the different tasks given different length of labeled data in the training set. In the figure the recognition error rates of the tasks, especially the error rates caused by rejection, decrease quickly when the length of labeled data increase from 1 to 4, and become stable when the length gets larger. That indicates the acceptance threshold of the task models can not be correctly set without enough labeled data. The reason for that is that too few labeled sequences of a task will lead to biased models and quite small similarity measurements for the unlabeled sequences. In this case, the system rejects them as unseen tasks. Thus, the threshold stays quite high and cause a lot of rejections during testing. In the experiment, in order to test the new task learning

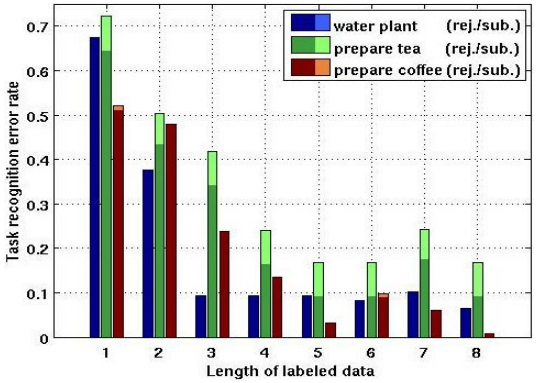

Figure 4: The recognition error rates based on different lengths of labeled data

strategy, only labeled data from two tasks are given. From

the results in Figure 4, it is found that a valid task model cannot be constructed without more than 4 labeled data. So the length of labeled observations of two tasks and the l_b are both set to 5. Figure 5 shows the task recognition error rates on the testing set when choosing different initial tasks. The results indicate that the unlabeled task was detected and correctly built up.

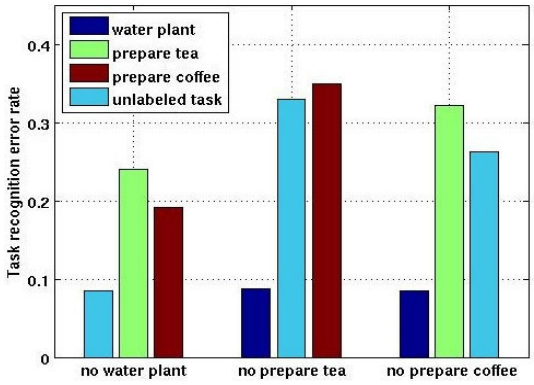

Figure 5: The task recognition error rate based on labeled data only from two tasks

6 Summary

Social robots that act in domestic environments need to be able to extend their knowledge during the interaction with their environment. Here we focus on the recognition of manipulative tasks. We put forward a semi-supervised incremental task learning strategy. The paper assumes that the robot can start the learning by a few labeled training samples and optimize the models afterwards without the need of a human pre-labeling. The task models are based on a Markov process that is coupled to a HMM-based recognition of primitive actions. A random model is taken into consideration to adjust the score of a primitive sequence. Thereby, the system is able to reject unseen tasks. Then, a solution is presented that collects and filters the rejected tasks in a buffer which is used to learn new models in an unsupervised way. This scheme provides the robot the ability to pre-structure its observation. In future work this could be the basis for an active learning strategy of the robot that could ask questions about previously unknown task sequences observed.

The experiments show the applicability of this approach. Pre-learned tasks were stably recognized from an initially labeled set of only four samples. Additional tasks that were previously unknown were newly instantiated and successfully recognized on a test set.

References

[1] A. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. In *Royal Society Workshop on Knowledge-based Vision in Man and Machine*, 1998.

[2] M.T. Chan, A. Hoogs, J. Schmiederer, and M. Petersen. Detecting rare events in video using semantic primitives with hmm. In *ICPR04*, pages IV:150–154.

[3] J. Fritsch. *Vision-based Recognition of Gestures with Context*. Dissertation, Bielefeld University, 2003.

[4] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. In *Int. J. Computer Vision*, pages 5–28, 1998.

[5] Z. Li, J. Fritsch, S. Wachsmuth, and G. Sagerer. An object-oriented approach using a top-down and bottom-up process for manipulative action recognition. In *DAGM06*, pages 212–221, Berlin, Germany, 2006.

[6] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 20:91–110, 2003.

[7] D.J. Moore, I.A. Essa, and M.H. Hayes, III. Exploiting human actions and object context for recognition tasks. In *Proc. ICCV*, pages 20–27, 1999.

[8] C. P. Nehaniv. Classifying types of gesture and inferring intent. In *Proceedings of the Symposium on Robot Companions: Hard problems and Open Challenges in Robot-Human Interaction AISB'05*, pages 74–81, Hatfield, UK, 2005.

[9] M. Pardowitz, R. Zöllner, and R. Dillmann. Unsupervised and incremental acquisition of and reasoning on holistic task knowledge for household robot companions. In *IROS06*, Beijing, China, 2006.

[10] Sira Panduranga Rao and Diane J. Cook. Predicting inhabitant action using action and task models with application to smart homes. *International Journal on Artificial Intelligence Tools*, 13(1):81–99, 2004.

[11] Ying Wu and Thomas S. Huang. Vision-based gesture recognition: A review. *Lecture Notes in Computer Science*, 1739:103–114, 1999.

[12] M. Yamamoto, H. Mitomi, F. Fujiwara, and T. Sato. Bayesian classification of task-oriented actions based on stochastic context-free grammar. In *Proceedings of FGR06*, pages 317–323, 2006.