Analysing Engineering Tasks Using a Hybrid Machine Vision and Knowledge Based System Application*

Ioannis Kaloskampis [1,2], Yulia A. Hicks [1], and David Marshall[2]

[1]School of Engineering, Cardiff University, UK
[2]School of Computer Science & Informatics, Cardiff University, UK
{kaloskampisi, hicksya}@cardiff.ac.uk, Dave.Marshall@cs.cardiff.ac.uk

Abstract

We propose a novel application that automatically analyses video sequences arising from the conceptual stage of design engineering tasks. It is capable of handling cognitive *activities, i.e. procedures which take place in the engineer's mind. A hybrid machine vision and knowledge based system framework is employed to efficiently identify stages of the design process and detect mistakes. Activity analysis is performed by examination of the temporal relationships between the engineer's actions. Experimental results captured in a complex, real life scenario showed that our system was able to correctly characterise input sequences with high accuracy rate.*

1 Introduction

The conceptual stage of design engineering is the initial step of the design process, where the vague statement of a design task is transformed into a set of requirements. Although there are several computer-based applications that aid in latter stages of the design procedure, the conceptual stage is usually performed on pen and paper.

Activity recognition systems using pattern recognition techniques have been used in the past for monitoring kitchen activities [1], card games [2] and nursing activities [3]. Inspired by these, we develop a framework for the purpose of analysing the conceptual stage of design engineering. The contributions of this paper are: (1) a novel industrial application that aids engineers in the conceptual stage of design, (2) a method of obtaining data from *cognitive* activities, *i.e.* procedures which take place in the engineer's mind which combines activity identification techniques with a Knowledge Based System (KBS), (3) a database of correct solutions to 3 bridge design tasks, which is *learned* by our system from input sequences annotated by professional civil engineers.

This paper is structured as follows: in §2 we give an overview of our system. We then present its parts: the Machine Vision unit (§3), the KBS (§4) and the Machine Learning component (§5 and §6). We evaluate our system in §7, presenting results captured in a real life, complex task. We conclude this article in §8.

2 System Overview

Our framework automatically analyses video sequences illustrating the conceptual stage of engineering design. In these, the engineer works on a given bridge design problem at a study desk, interacting with various objects (pencil, ruler *etc*) and performing simple actions such as writing, sketching and measuring distances. During the design process, the engineer consults various knowledge sources, like regulation manuals, books or the internet. Our system provides this "expert knowledge" on-demand, in the form of a computer application which includes a KBS. The user enters simple command line queries, such as "bridge type", which brings up information regarding the specifications of various bridge models. We record the design procedure in two ways:

1. User's interactions with various scene objects are recorded using a static camera; the footage is analysed with the aid of an activity identification system, which returns a sequence of the form:

$$S = \{p_a(t_1 \rightarrow t_2), p_b(t_3 \rightarrow t_4), \dots\} \qquad (1)$$

   where $p_x(t_i \rightarrow t_j)$ an action (*e.g.* erasing, writing *etc*) that starts at time t_i and ends at t_j.

2. The KBS timestamps user's queries; from this information we attempt to deduce on which part of the design (*e.g.* foundations, piers, cost *etc*) he works at a specific time slot. We assume that the start of such activities coincides with the time point when the user inputs a relevant query to the KBS; furthermore, as we show later in this article (§4), our KBS provides the means to define the end point of each activity of this type. We therefore obtain another time sequence M, of similar form as S.

By combining S and M we obtain sequence T, which includes the complete timeline of the engineer's work. We compare T against a database of "correct" behavioural patterns in order to identify design stages and discover errors. The database is *learned* from input sequences which are annotated by experts and is represented as a Hierarchical Hidden Markov Model (HHMM) [4], whose topology implements the activities' hierarchy and structure. Thus, activity analysis is posed as a HHMM inference problem.

3 Extracting Actions from Video

Our system's machine vision unit consists of 2 components: a mechanism to track selected objects and a framework that maps from interactions of tracking windows to actions.

*This project was funded by the Human Factors Technology Centre (HFTC), Wales, UK (www.hftc.cardiff.ac.uk).

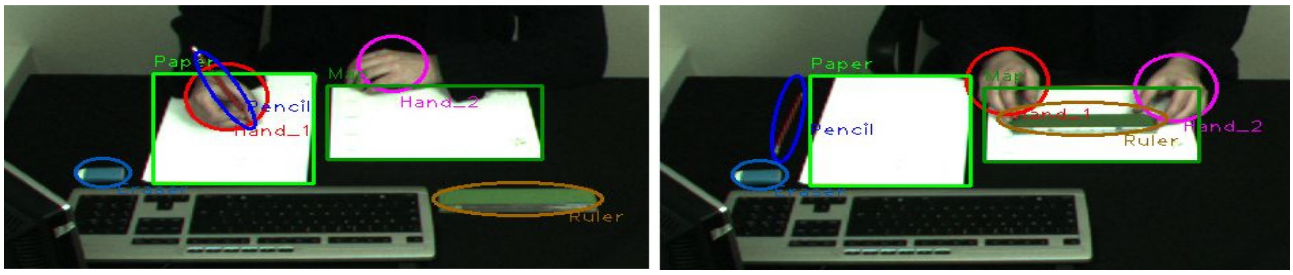

Figure 1. QSR framework application. *Left*: Action writing or sketching: {Hand (S)urrounds Pencil} and {Paper (S) Hand}. *Right*: Measuring on map: {Hand (T)ouches Ruler} and {Map (S) Ruler}.

3.1 Object tracking

The objects participating in the scene are at least partially visible at all times; we monitor their movements by placing one video tracker on each object at the first frame of each sequence. We employ a tracking algorithm which uses a colour histogram-based observation model and a second order autoregressive dynamical model [5].

3.2 Action identification

Extraction of actions from footage is achieved by identifying patterns of *qualitative spatial relations* (QSR) [1] between the tracking windows of moving objects. Two tracking windows surrounding key objects can be either spatially Disconnected (D), or connected through the surrounds (S) or Touches (T) relationships. This framework is capable of detecting simple actions such as measuring (Figure 1). We specify the set of possible object interactions *a priori*; *e.g.*, we define that relationship {Hand(T)Ruler} and {Map(S)Ruler} is interpreted as the action *measuring*.

We disambiguate between spatially similar actions, such as writing and sketching, by statistically analysing the motion trajectories of the objects involved in these. This analysis is performed with the aid of a continuous Hidden Markov Model (HMM). The model is trained with $T = 500$ sequences, 250 representing the class *sketching* and 250 the class *writing*.

4 The Role of KBS

We would like to recognise activities relevant to a bridge design procedure, such as *"choose bridge type"* and *"estimate soil condition"*. We cannot detect these *cognitive* activities by the interaction of the engineer with standard objects; we achieve this by monitoring his interactions with the KBS (Figure 2): We have omitted important details from the design scenario given to the participant, without which the task cannot be completed. The engineer can access all missing information at any time by consulting the KBS. Each piece of provided information is linked to a specific part of the design task. Therefore, when the user queries the system, we can deduce on which stage of the design he is working. When he completes a cognitive activity, he inputs the result in a cell corresponding to this activity. The software records input time, which signals the end of the relevant activity. Thus, apart from providing expert knowledge, the KBS informs us about the starting and ending point of a cognitive activity.

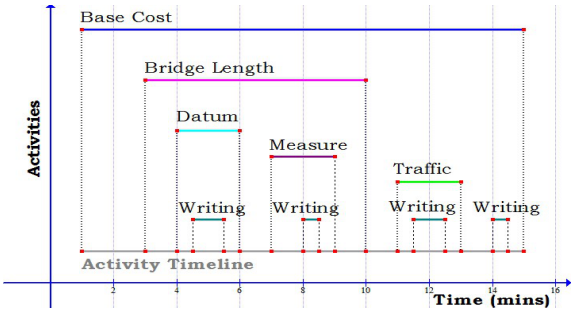

Figure 3. Forming an activity's timeline from temporal occurrence of its constituent actions. Each action's start and end point is inputted as an event in the timeline.

5 Sequence Analysis

Analysis of input video sequence with the aid of QSR, object trajectory classification and KBS query time stamping offers a basic understanding of the studied scene. The resulting data is in form:

$$T = \{p_a(t_{a,s}), p_b(t_{b,s}), p_b(t_{b,e}), p_a(t_{a,e}), \dots\} \qquad (2)$$

with $p_x(t_{x,s}), p_x(t_{x,e})$ start and end of a sub-activity or action p_x. Formulation of sequence T is shown in Figure 3. Note that this representation can handle *concurrent* activities.

Due to the design procedure's variable nature, the order of a sequence's constituent elements can be changed and elements can be added to it or omitted from it without significant alteration of the overall process. To model this variability, we represent design activities using statistical graphical models (§6) and separate actions (sequence elements) into 2 classes: (1) *critical*, which are vital for the completion of certain activities, *e.g.*, activity *examining soil condition* is almost impossible to solve without using a graphical representation of the problem data; thus, *sketching* is a *critical* action for this activity, (2) *common*, which do not constructively affect the design process; *e.g.*, in T, let p_a be *erasing*: while designing, the engineer may or may not need to erase, without *erasing* affecting the process, therefore *erasing* is a *common* action. *Common* actions can be thought of as noise; eliminating them leads to a more compact problem representation.

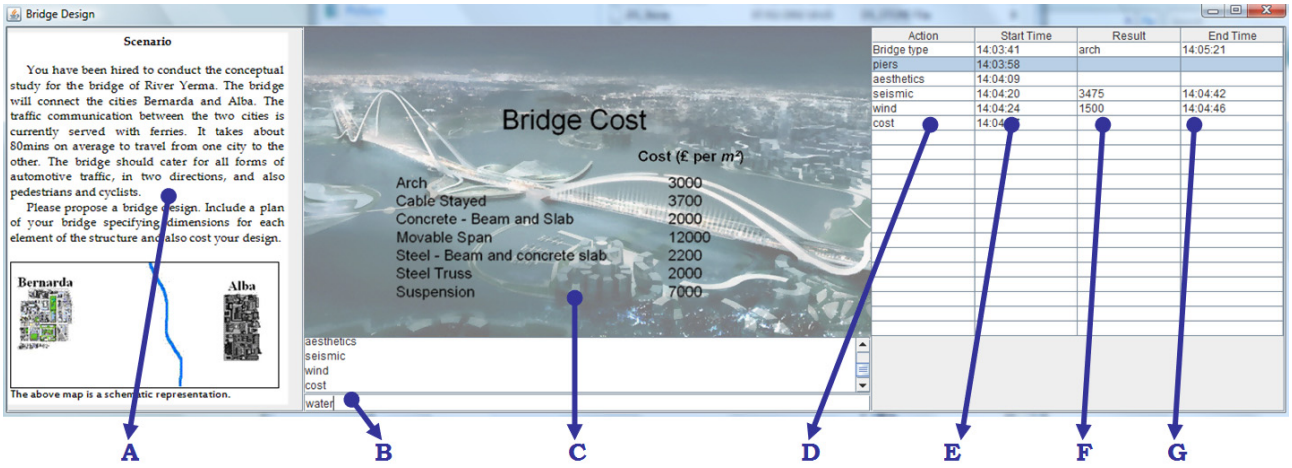

Figure 2. Overview of the KBS interface: (A) Task scenario, (B) input console, (C) returned expert knowledge, (D) entered queries and (E) their timestamps, (F) slot for user to input result and (G) result's timestamp.

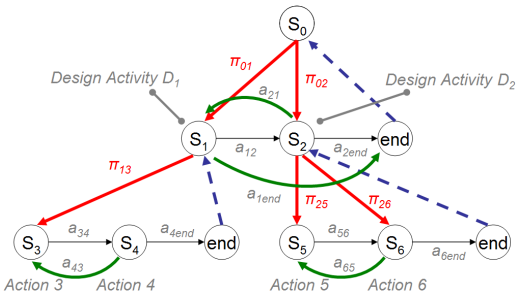

Figure 4. A 2-level HHMM representing 2 sample design activities, D_1 and D_2. We denote transition probability between two nodes *(i, j)* of the same level with a_{ij} and with π_{mn} the initial probability of child n of parent state m. Actions below a state represent the emitted symbol.

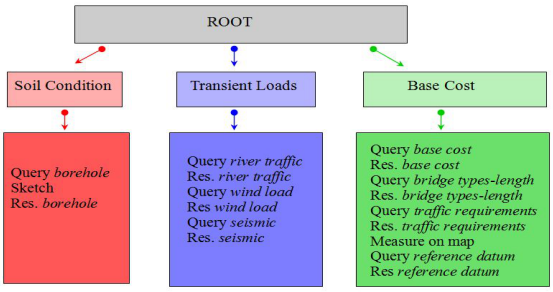

Figure 5. State hierarchy for the activities in the bridge task. Only *critical* actions are shown.

Classification of actions in *common* and *critical* with respect to each activity is performed by experts.

6 Activity Representation

We represent actions constituting an activity as states of an *action chain*. Transitions from one state to another are allowed or not subject to the set of rules that govern the activity. Such representations have been employed frequently in the past, usually taking the form of the HMM and some of its variations. However, these flat representations cannot sufficiently represent complex activities, as they fail to model their hierarchic structure [6]. More recent work has adopted extensions of the HMM in a hierarchical manner, such as the HHMM [6], which we use here (Figure 4).

In order to produce more compact behavioural models and reduce computational complexity, we simplify sequence T by ignoring *common* actions.

6.1 The Hierarchical Hidden Markov Model

Each state of the HHMM can either emit observations ("production states") or strings of observations ("abstract states"). Each abstract state is a sub-HHMM that can be called recursively and integrates *end states*, which signal when the control is returned to the parent HHMM. We work with discrete HHMMs, which are defined by a 3-tuple $< \zeta, Y, \theta >$: the topological structure, ζ, defines the number of levels, the state space at each level and the parent-children relationship between levels. The observation alphabet, Y, is the set of the symbols emitted by the model's states; the set of parameters, θ, includes the matrix of transition probabilities between nodes, the initial probability distribution between the children of each node and the observation probability distribution.

The topology of the HHMM implements the rules that govern the activities taking place within the design task. State hierarchy for the model is shown in Figure 5. The model is learned from annotated input sequences as described in [6]. Discovery of a potential mistake is the problem of determining if a sequence resulting in one of the experiments is a possible output of the HHMM. Activity identification is, likewise, the problem of determining which activity has been performed, given an input sequence, $y_{1:T}$. These problems can be solved by computing probability $P(S_k \mid y_{1:T})$ for all sets of nodes $S_k = \{k\} \cup parents(k)$ in the HHMM. This is achieved by converting the model to its corresponding Dynamic Bayesian Network and applying the Junction Tree algorithm [4]. Pseudo-code for sequence analysis is given in Algorithm 6.1.

Table 1. System performance evaluation.

Activities	ID	ERR	Total (%)
Soil Condition	9/10	9/10	90.0
Transient Loads	9/10	9/10	90.0
Base Cost	9/10	8/10	85.0
Overall	27/30	26/30	88.3

Algorithm 6.1: ACTIVITY ANALYSIS(T_i, D_j)

comment: Identify activity in sequence T_i

$D_j, j \in \{1, \ldots N\}$ design task

$U(D_j)$ common actions for D_j

$G(D_j)$ graphical sub-model for D_j

for $j \leftarrow 1$ **to** N

do $\begin{cases} \text{Define } U(D_j) \\ T_i^{(j)} = T_i \backslash U(D_j) \\ P_i^{(j)} = P(T_i^{(j)} | G(D_j) \end{cases}$

if $\max P_i^{(j)} = 0$

then T_i erroneous

else $T_i : \Leftrightarrow \arg\max_{D_j} P_i^{(j)}$

7 Experimental Results

We tested our system in a real-life bridge design scenario, designed with the help of experts. 4 professionals and 12 students participated in our study, resulting in a total of 20 hours of video footage. 40 sequences were extracted from this video to serve as our training set. In these, professional civil engineers execute one of three complex behaviours: *evaluate soil condition*, *estimate transient loads* and *evaluate bridge cost*. State hierarchy for these activities is shown in Figure 5. The test data is a different set of 60 sequences obtained in a similar way, but, this time, the task was performed by students.

Evaluation of our framework's efficiency in behaviour analysis is presented in Table 1; system performance exceeds 88%. Analysis is based on system's ability to identify correctly performed activities (column "ID") and detect mistakes in the test sequences (column "ERR"). Ground truth was provided by experts, who annotated the extracted sequences and evaluated each participant's study output.

To justify the use of the HHMM, we compare the model's performance with the flat HMM (Figure 6). An HMM is created for each complex behaviour; its parameters are learned using training data. We consider the number of each task's constituent actions as a measure of task complexity and observe that the flat HMM performs equally well with HHMM in the task *evaluate soil condition*, which includes a small number (3) of actions. As complexity increases, performance slightly deteriorates for both models, but it is clear that HHMM exhibits a milder decrease rate. These findings are consistent with results reported in [6].

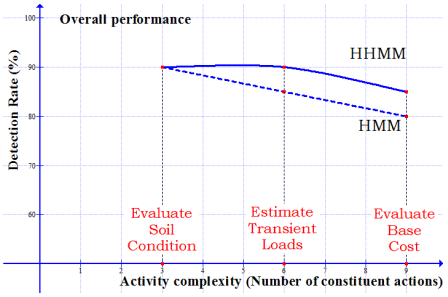

Figure 6. Comparing HHMM and HHM in behaviour analysis. HHMM performs better than the flat HMM.

8 Conclusion

We have presented a system that uses a combination of machine vision with KBS, capable of efficiently analysing video sequences arising from design engineering tasks. Experimental results captured in a real life, complex task that can be carried out in a large number of ways demonstrate the ability of our framework to model complicated human behaviour. The high rate of correct behaviour detections (88%) showcases the usefulness of our system as a tool that can aid in the improvement of engineering design process.

In future work, we will apply our system to various other design engineering tasks (*e.g.* dam construction) and explore its applicability to other areas such as medicine (*e.g.* surgery monitoring).

Acknowledgment

We would like to thank Christos Katsaras, Barry Mawson, Professor John Miles, Professor John Patrick and Victoria Smy for their expert advice on the development of the bridge design task.

References

[1] M. Sridhar, A. G. Cohn, and D. C. Hogg: "Learning Functional Object-Categories from a Relational Spatio-Temporal Representation," *Frontiers in AI and Applications*, vol.178, pp.606-610, 2008.

[2] D. Moore and I. Essa: "Recognizing Multitasked Activities from Video Using Stochastic Context Free Grammar," *AAAI*, pp.770-776, 2002.

[3] T. Inomata, F. Naya, N. Kuwahara, F. Hattori, and K. Kogure: "Activity recognition from interactions with objects using dynamic Bayesian network," pp. 39-42, 2009. *3rd ACM Int. Workshop on Context-Awareness for Self-Managing Systems*, pp.39-42, 2009.

[4] K. P. Murphy and M. A. Paskin: "Linear Time Inference in Hierarchical HMMs," *NIPS*, 2001.

[5] P. Perez, C. Hue, J. Vermaak, and M. Gangnet: "Color-Based Probabilistic Tracking," *ECCV*, vol.1, pp.661-675, 2002.

[6] N. T. Nguyen, D.Q. Phung, S. Venkatesh, and H. H. Bui: "Learning and detecting activities from movement trajectories using the Hierarchical HMM," *CVPR*, vol.2, pp.955-960, 2005.