DELIVERABLE D1.2 (v1.0)

First prototype Implementation of the CV Framework

Draft/Final Version 1.0

13 Nov 2002

Authors: *Kingsley Sage, Jonathan Howell, Wolfgang Ponweiser, Hilary Buxton, Markus Vincze, Christof Eberst, Stepan Obdrzalek*

Project acronym: **ACTIPRET**

Project full title: **Interpreting and Understanding Activities of Expert Operators for Teaching and Education**

Action Line IV.2.1: **Real Time Distributed Systems (Cognitive Vision)**

Contract Number: **IST-2001-32184**

Contents

1 Introduction

This deliverable presents the results from the first integration meeting that took place on 28-30 October 2002 at Profactor GmbH in Steyr, Austria. This demonstration was intended to evaluate and demonstrate real time framework operation and communication between the components of the Cognitive Vision (CV) Framework (see Deliverable 1.1)

Section 2 describes the set-up of the first demonstration system. Section 3 details the expected system behaviour, the test sequence used for this integration system and summarises the performance of the demonstration system. Section 4 lists a number of key elements of the further work.

2 Integration Set-up

For the integration meeting nine different components were provided from the project partners. To enable a first integration all components were available in a dummy version, where just the communication capability was implemented. The next integration step was then to use the real component versions, where the actual functionalities were realised.

The demo functionality 'a hand moving towards a CD' was selected to use a set-up (see Figure 1) in which most of the provided components are included and which corresponds to the example set-up presented in Deliverable 1.1 (see Figure 6 in that document) for a single fixed stereo camera pair.

This functionality maps some of the initial steps of the first full ActIPret scenario 'play CD in player' or 'playCD' (as referred to in the Deliverable 5.1 Conceptual Language definition file).

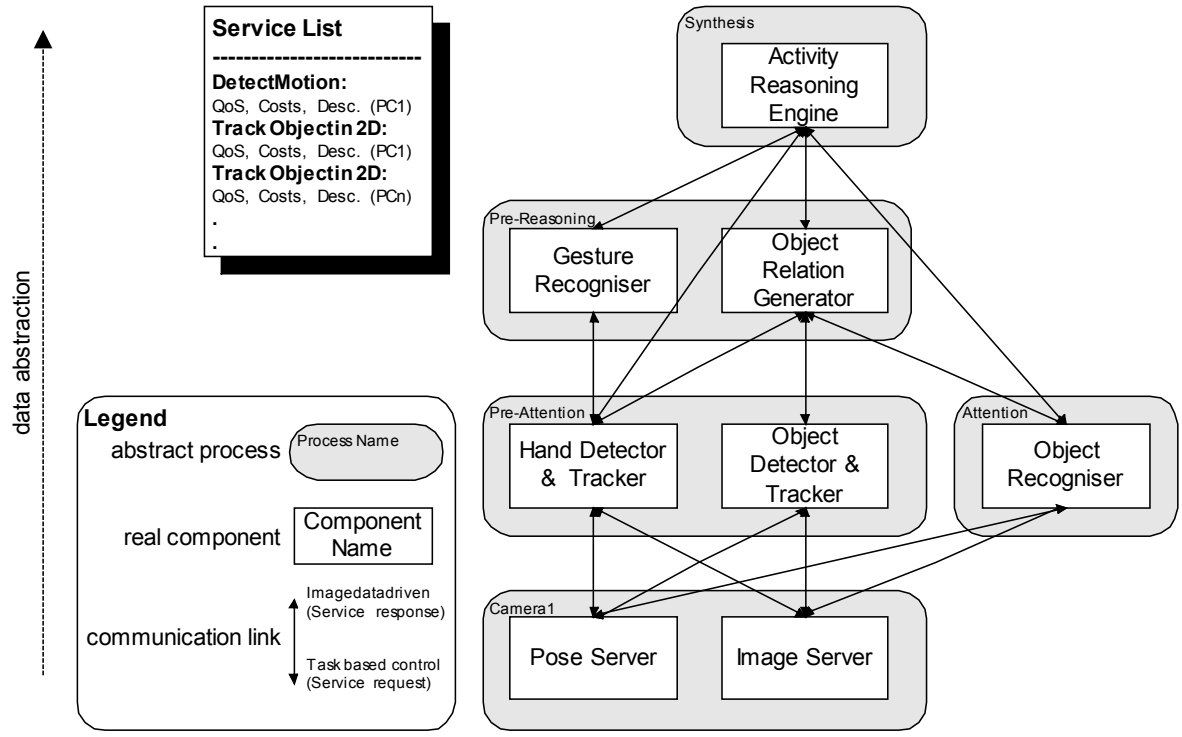


Figure 1: Demonstration set-up

Table 1 lists the involved components:

Component Name	Provided services
Service List (SL)	-
Activity Reasoning Engine (ARE)	-
Gesture Recogniser (GR)	RecogniseGesture
HandTracker (HT)	TrackObject
ImageServer (IS)	ImageServer
CameraPoseServer (PS)	PoseServer
Object Relationship Generator (ORG)	DetectProximity
Object Recogniser (OR)	Recognise3D
Object Detector & Tracker (ODT)	TrackObject, DetectObject

Table 1: Components and their provided services of the demonstration set-up.

For the first demonstration system, there were no Activity Planner and User HMI components as described in Deliverable 1.1. The ARE did not, therefore, need to provide any services to higher level components. As a result, ARE output was internal to that component and displayed to a Xterm console.

3 Demonstration system experiments

3.1 Service call sequencing and observed behaviour

The service request and according response sequencing for the nine components used for this integration meeting for the CD demo functionality are summarised in Table 2 below:

sender - receiver/ processing component	Message name	Description
ARE – GR	AutoUpdateGestures-ForHandObject	Motivated by the assumption of an initialiser 'moving hand'
GR – HT	TrackObject	Request hand trajectories continuously
HT – IS	AutoUpdateShmImage	Request stereo images continuously
HT – PS	GetCameraPose	Request camera pose

sender - receiver/ processing component	Message name	Description
*IS – HT	AutoUpdateShmImageResponse	Deliver stereo image data
PS – HT	GetCameraPoseResponse	Deliver camera pose
HT		Using colour information in the stereo images to calculate 3D hand pose
*HT – GR	TrackObjectResponse	Deliver hand pose hypotheses
GR		Using a trained TDRBF
*GR – ARE	AutoUpdateGestures-ForHandObjectResponse	Deliver gesture hypotheses
ARE		Increase hypotheses belief
ARE – ORG	AutoUpdateObjects-NearTrajectory	Request objects near the hand trajectory
ORG – HT	GetObjectTrajectory	Request hand trajectory
HT – ORG	GetObjectTrajectoryResponse	Deliver hand trajectory
ORG		Calculate 'space of interest'
ORG - OR	Recognise	Request for pre-learned objects in the 'space of interest'
OR – IS	GetShmImage	Request image
IS – OR	GetShmImageResponse	Deliver image
OR – PS	GetCameraPose	Request camera pose
PS – OR	GetCameraPoseResponse	Deliver camera pose
OR		Using learned object database to detect objects
OR - ORG	RecogniseResponse	Deliver detected CD's and their poses
ORG		Using one of the CD's to initialise the object tracker[1]
ORG – ODT	TrackObjectInitialised	Request CD poses continuously
ODT - IS	AutoUpdateShmImage	Request image continuously

[1] The ARE should bias the selection which objects should be tracked.

* These responses are calculated and sent continuously.

sender - receiver/ processing component	Message name	Description
ODT - PS	GetCameraPose	Request camera pose
IS - ODT	AutoUpdateShmImageResponse	Deliver image
PS - ODT	GetCameraPoseResponse	Deliver camera pose
ODT		Using object model knowledge to reliably track object

Table 2: Demonstration sequence

3.2 Off-line training

COGS made use of an existing hand gesture database to pre-train the GR Time Delay Radial Basis Function (TDRBF) network to recognise functional gesture phases appropriate to the task of 'picking up'. The gesture data used for the experiments in this paper was the Terminal Hand Orientation and Effort Reach Study Database created by Human Motion Simulation (HUMOSIM) at the Center for Ergonomics, University of Michigan, USA. Further information about this database can be found at [1] and [2]. The HUMOSIM database uses a torso centred co-ordinate system and a measurement system in centimeters. For the demonstration, we trained the TDRBF on 18 complete HUMOSIM hand trajectory sequences encompassing functional gesture phases for both hand moving towards an object (away from a torso) and away from an object (towards the torso).

The OR component used a database of objects to be recognised) that was built prior to system startup. Each database object is represented as a collection of images of these objects. For better results, the images should be visually as similar as possible to these encountered during runtime (i.e. similar image quality, resolution, illumination condition, etc.). The learning of the database is performed in real time during the component startup.

3.3 Summary of performance of the demonstration system

The demonstration system was built using a single static stereo camera setup. The system consisted of nine independent processing components built using the common ActIPret framework built around OSCAR and CORBA middleware. Message passing was achieved in a timely manner and the system was able to produce a high level symbolic interpretation from low level vision derived signals. More specifically, the system was able to:

- detect and track a hand (HT – see Figure 3),
- recognise functional gestures phases for that hand (GR – see Figure 5),
- request object relationships of the tracked hand (ORG),
- recognise CDs in the environment of the hand (OR – see Figure 4),

- initialise the object tracker with the CD pose information (ODT),and
- generate hypotheses about which objects the hand was interacting with (ARE).

The system proved capable of performing these operation on both the pre-stored image sequence and live images.

3.4 Example output from demonstration system

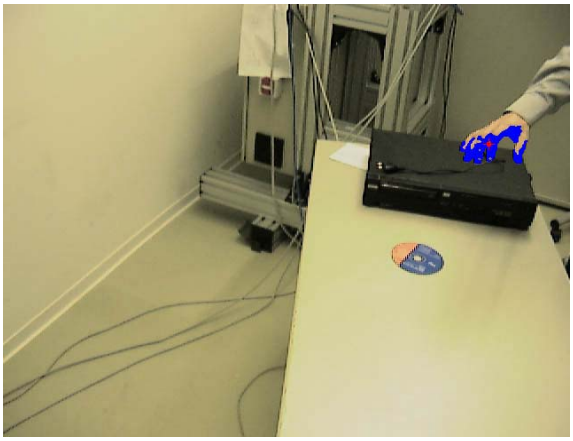
At this first demonstration stage, the system output is the interpretation in the ARE. At this stage the internal function of the ARE is very simple and consists of a linear Finite State Machine (FSM) that uses a single state variable to represent the state of the task control. The ARE tries to build and maintain a simple list structure of action, activity and event based hypotheses (such as the 'pickup(hand1,cd1,location)'). The task based control rules that were embedded in the ARE FSM relevant to the demo functionality 'hand moving toward the CD' can be summarised as shown in Table 3 overleaf:

ARE FSM State	**Description**	**Task control strategy**
0	INITIAL	Change to state 1.
1	LOOKING_FOR_1st_PURPOSEFUL_ HAND_MOVEMENT	Start the GR RecogniseGesture service to look for purposeful gestures (assume there is only 1 hand that the GR and HT will find). When the returned data indicates a moving hand, change to state 2.
2	LOOKING_FOR_1st_OBJECTS_ON_ HAND_TRAJECTORY	Start the ORG DetectProximity service to look for objects on the trajectory of the moving hand. Use the returned data to build and maintain a set of hypotheses about which objects the hand is interacting with. When the returned data indicates the hand gesture GET has completed, attempt to verify which objects the hand has interacted with, change to state 3.
3	LOOKING_FOR_2nd_PURPOSEFUL_ HAND_MOVEMENT	And so on …

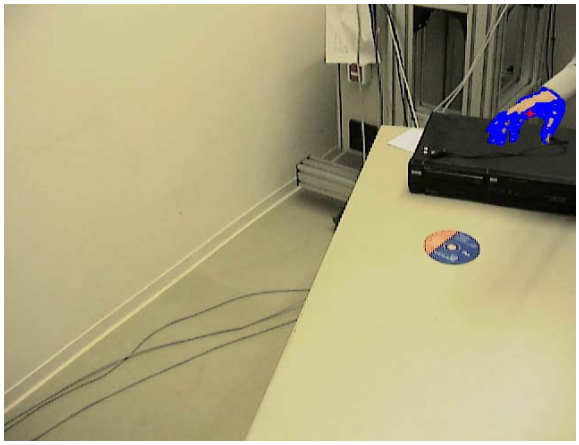
Table 3: Demonstration sequence

We can define success, therefore, for the demonstration system as a whole for the demo functionality 'hand moving toward the CD' if the ARE is able to make progress through the FSM from state 0 to state 3. On arrival at state 3, the ARE should contain hypotheses to the effect that the hand may be picking up any of the objects defined along the hand trajectory path as determined by the ORG.

Figures 3 to 5 show the output of intermediate steps, in particular of the components HandTracker, and ObjectRecogniser.



LEFT IMAGE



RIGHT IMAGE

Figure 3: Example stereo pair of images from the HandTracker (HT). Detected hand shown in blue. (black on a grey scale printout).

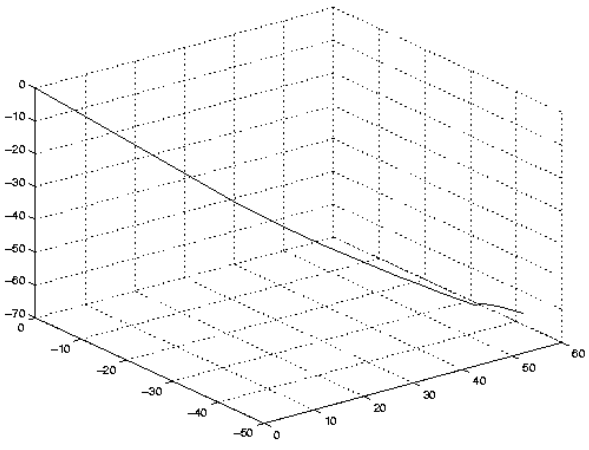


Figure 4: Example of a reconstructed hand trajectory based on smoothed output from the HandTracker (HT). Here the data has been rescaled into a torso centered co-ordinate system using the initial hand position as the origin. This plot assumes a uniform temporal interval as no timestamp data was available.

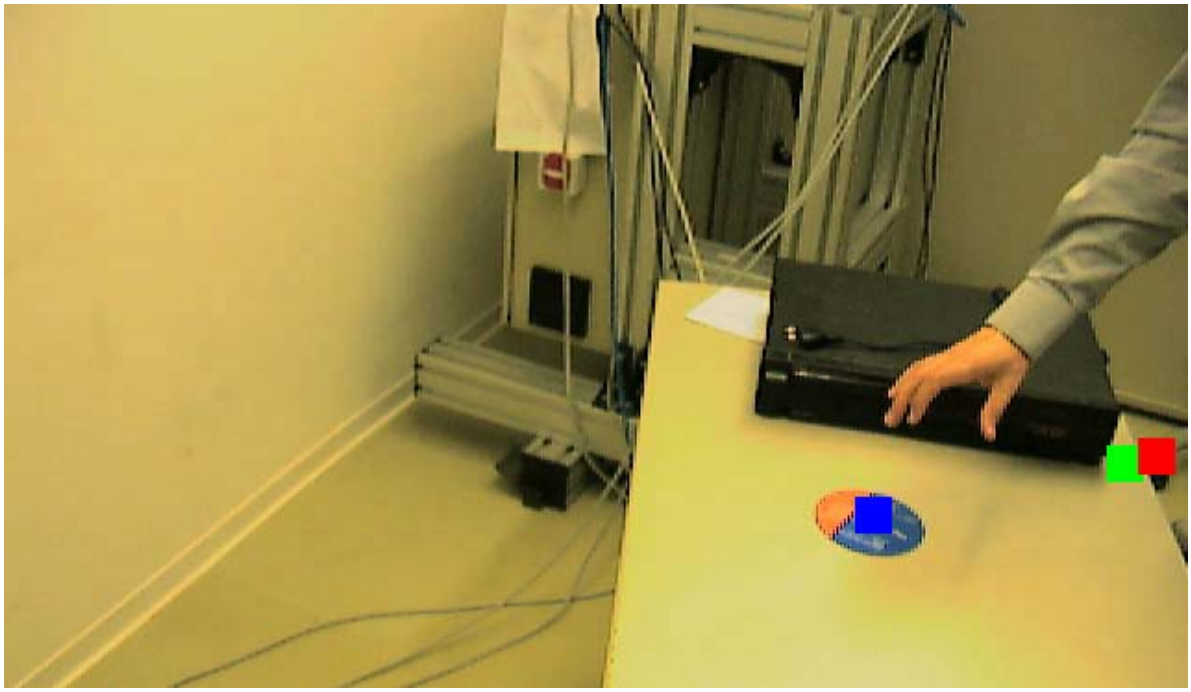


Figure 5: Example of object localisation provided by the ObjectRecogniser (OR). In this case, the OR has detected three candidate CD objects (marked by the blue, green and red squares). The two at the right are false responses, as the particular scene background is visually similar to a CD. The OR results would be improved by a) better image quality, b) removal of the cluttered background, and c) restricting the space of interest to just the table

4 Further work

It should be noted that any qualitative evaluation of the components functionality was outside the scope of the demo; only the interrelation between the components was tested.

The current demonstration works with nine components. There are still some evaluation tests and extensions that could be usefully performed on this configuration before work begins on project task 1.3 (full implementation of the CV framework) including the:

- evaluation of the timing behaviour of the entire system, including issues such as the overhead attributable to the dynamic and distributed processing structure, and
- introduction of a unique logging instance to enable system tracing.

In the light of the integration meeting, the following issues also need to be addressed:

- enabling of blocking messages to enable a greater range of component complexities and simplifying programming,
- redefinition of the IDL types used in the interfaces to unify service definitions revised in the light of the first integration,
- the erratic timing nature of the 'soft real time' messaging and the varying service response times of individual components means that a new message may not be available at every system cycle. This means that timestamps are required with all data (this issue is being addressed in the revised IDL definitions),
- the GR needs to be trained on actual hand trajectory data from the HT (already agreed in principle between FORTH and COGS), and
- FORTH will experiment with variable smoothing of the hand trajectory data with the HT as a function of top down functional gesture phase expectation determined by the GR.

Future work will also focus on introducing active vision to the ActIPret Demonstrator:

- The experiments performed thus far have been with fixed cameras only For the next version of the Framework (1.3), PROFACTOR's ViewController (VC) and ViewContractManager (VCM) will be integrated and tested with multiple active cameras mounted on robots.
- A providing component that includes active-vision properties, i.e. one that commands a sensor, is started by the requesting component on the camera/robot instance that has been selected by the VCM according to estimated cost and quality properties (see Deliverable 6.1). The providing component will send view-requests to its associated VC in order to improve its own performance. The VC will merge all view-requests and command the robots/cameras. The VCM and VC will cooperatively evaluate the cost for performing a service for a given robot/camera entity.

5 References

[1] Buxton, H., Howell, A.J. and Sage, K., "The Role of Task Control and Context in Learning to Recognise Gesture'" Cognitive Vision Workshop, Zürich, Switzerland, September 2002.

[2] Howell, A.J., Sage, K., and Buxton, H., "Developing Task-Specific RBF Hand Gesture Recognition", submitted to 3rd International Conference on Computer Vision Systems - ICVS 2003, Graz, Austria, April 2003.