Draft Implementation of the Collaborative Cognitive Map Architecture

CoCoMaps Deliverable T8.D1

Author:	Thor List, CMLabs
Created:	November 2017
Last modified:	December 2017

References:	State-of-the-art implementation of the CM Architecture Draft implementation of the new CCM Architecture

CMLABS
Powering the AI Revolution™

ICELANDIC INSTITUTE FOR INTELLIGENT MACHINES
VITVÉLASTOFNUN ÍSLANDS

Contents

Introduction

The Collaborative Cognitive Map (CCM) architecture is a new AI agent / robot middleware built to enable coherent integration of time-based control systems for robots and other sensor-actuator systems such as embedded systems and distributed systems, with specific focus on multimodal multi-party dialog and task execution.

The original Cognitive Map Architecture was created in collaboration with Honda Research Institute in California for the purpose of adding learning and human interaction to their ASIMO robot. It was implemented in Psyclone version 1 and ran a combination of C++, Java and LUA code.

An earlier report (*State-of-the-art implementation of the CM Architecture*) from the CoCoMaps project detailed the upgrade of this earlier work to produce a current and viable implementation, in preparation for moving this into the realm of collaborative robotics.

This report details the first draft release of the Collaborative Cognitive Map architecture. It allows two or more robots to share information in real-time and to negotiate about the validity of observations when the data does not quite agree.

CoCoMaps is funded by the ECHORD++ project through the European Union's 7th Framework Programme for Research, Technological Development and Demonstration, under grant agreement no 601116.

Making the Cognitive Map Architecture Collaborative

The original Cognitive Map Architecture (CMA) allowed a single robot to store observations and tasks for its own use. It allowed the robot itself to search observations based on temporal and spatial constraints, such as the location of an observed object at a specific point in time, or which object was within a location boundary.

A major component in the new architecture is the CCMCatalog. It can be viewed as a shared memory space which exists outside any one robot, but is accessed in real-time by all robots. Each robot can enter observations into the shared memory and these observations will be tagged by the robot system ID and the ID of the object which the observation describes. Other robots can now see and search for this observation and relate this to its own observations of the object.

To keep track of objects the concept of object identity is introduced. This way, two robots can add observations about an object they observe and at some point they can agree that

these observations are in fact relating to the same object by creating a shared object identity. This can either be from a list of predefined identities (such as employees in a company) or dynamically created identities (objects present in the scene).

Each object is tracked in the scene map. Each robot has its own view of the map and their own location within it and these views can be synchronised in the CCMCatalog to enable the robot to agree on the location of an object. If they do not agree they can initiate a negotiation which may result in a recalibration of their own map view in relation to the other robot's map, or it may result in either an agreement or disagreement with the other robot about the object's location. Whether they agree or not can be recorded in the CCMCatalog as an observation's consensus value and each robot can put in their own confidence value, which will be used by the robots during the negotiation.

CCMCatalog – Overview

The CCMCatalog implements a shared memory space that exists outside any single robot, and can be accessed in real-time by any robots that are part of the CCM. Each robot can enter data - called "observations" - into the CCM, which will automatically be tagged by the robot's unique identifier (UID) and the UID of the object which the observation describes. When an entity or event is logged in the CCMCatalog by one robot the other robots can see it, search for the object described, and relate it to their own observations. As a result, the CCMCatalog provides a way for the robots to corroborate their sensory classifications (e.g. whether a particular person is present in the room), to coordinate their movements (so as not to bump into each other or other objects), and to negotiate a course of actions (operations on objects in the environment).

To keep track of objects we introduce the concept of a *negotiable object identity* in the CCMCatalog. It works by allowing two (or more) robots to add individual observations about any object in their surroundings yet at a later point agree that these observations are in fact relating to the same object, resulting in a *shared object identity* entity being created. This can either be from a list of predefined identities, such as employees in a company, or dynamically created identities, e.g. objects present in the scene.

One component of the CCMCatalog keeps track of the 2D maps of the surroundings as reported by each robot. The location of each object the robots enter into the CCMCatalog is tracked in the robots view of the scene map and the maps of the robots are correlated in the CCMCatalog to allow translations between them. The locations of all objects and the robots themselves are accessible to all robots with or without translation into their own map reference frame. Each robot has its own location and view within it that it can use to predict where objects are, and multiple views can be synchronised via the CCMCatalog to enable the robots to agree on the location of objects in the scene. Negotiations about entities are initiated if either robot disagrees about the identify or location of an object, e.g. a robot's

position in the map, but also if an object listed in one observation is predicted by either robot to be the same object as referenced in another one. A negotiation may result in (i) an agreement of the observations referring to the same object, in which case a shared object identity is created, (ii) a disagreement that leads to a recalibration of e.g. a robot's local map view in relation to the other robot's map, subsequently resulting in agreement, or (ii) it may result in unresolved disagreement. Whether the robots agree or not is recorded in the CCMCatalog as a joint observation's consensus value, and each robot computes its own confidence value in that agreement, which is used by the robots during the negotiation.

The robots can engage in a joint search of a given area or office, and they will dynamically agree on how to execute the search pattern, using the CCMCatalog's negotiation mechanism. The negotiation algorithm takes into account the areas of the space which has most recently been observed and which areas compliments the most recently searched areas best. Because a robot's vision will be degraded while it moves the robots will attempt to agree that only one robot will move at any given time, but any robot can choose to ignore this and move independently if agreement is not reached.

CCMCatalog Implementation

The CCMCatalog is implemented in C++ within the same architectural framework on which Psyclone is built. This makes them instantly compatible and allows a system designer using CCM to place the CCMCatalog component either inside one of the robots or outside on a separate computer. Either way, logically the CCMCatalog does not belong to any single robot and no robot has more access or rights to the component than any other robot.

As such, multiple independent Psyclone systems will have equal access to the single CCMCatalog, allowing them to create new objects, enter observations, query existing data and negotiate about data and tasks using Collectors and Negotiators running locally inside each robot's Psyclone system. Each Collector and each Negotiator has a direct line of communication with the CCMCatalog across the network and all communication is synchronised to the nearest microsecond. No robot has the ability to communicate directly with any other robot, all data and negotiation goes through and is logged by the CCMCatalog. Currently the CCMCatalog is programmed in C, to make it very efficient for searching and querying observations efficiently. Any other technology would be slower when going to thousands of observations.

The CCMCatalog contains a number of distinct handlers of information and operations, which will now be described.

Maps

The CCMCatalog can have a number of maps, each with specifications on how they relate to each other. One map could be a building schematic and related ones could be more detailed

maps of each room, for example. Each robot will also have its own map of its own environment and these will be matched with the CCMCatalog based on observations. Each observation will relate to one map by ID and can be queried from another map's point of view. Visual maps can also be requested for operator monitoring viewing.

Objects and Identities

An observation about an object can be entered into the CCMCatalog without yet knowing the identity of the object. The identity can be associated later on and each robot can in principle associate its own view, i.e. the robots don't initially have to agree. Once the identity of an object has been agreed this identity will be associated with the object for past observations as well. An object can have any type and new types can be created on the fly. Standard types include humans and robots, but the robots can at any point decide to create a new type such as chair or table.

Storing Observations

Observations of a range of types can be collected and stored in the CCMCatalog. A ***Location*** observation contains the X and Y coordinates of the object's centre point in a specific map and also contains an optional size (width, height) estimation. Other types of observations could be of type **Dimension**, **String**, **Integer**, **Time**, etc. The observations would traditionally be collected from data flowing in the individual robots Psyclone system by a CCMCollector which will also add the system's own ID and other metadata such as timestamp, confidence and additional data which may later on be of interest. Observations are usually collected several times per second and the CCMCatalog will manage the memory usage by keeping all observations for a period and after this summarise them into fewer data points which are kept and stored on disk.

Retrieving Observations

Any system connected to the CCMCatalog can retrieve and query observations using a rich query specification. This includes querying observations by type, time period and system ID. Each observation type will handle querying of values from, to and between which for location observations would be an X,Y square, but for other types of observations means something different. The system is set up so custom and more complex observation types can be added later on while still being entered, stored and queried using the same and existing mechanisms.

Negotiation

When two or more robots disagree on observations they enter into a managed negotiation phase which may result in a settled agreement or may result in a decision to disagree. What the robots individually do afterwards is up to them. The negotiation goes through a number of steps and robots can choose to skip any step or exit the negotiation at any time. First of all, each robot needs to provide a confidence level which tells the other robots how confident this robot is that what they say or ask for is best. Once all the confidence votes are

in the robots can choose to select the winner or carry on. If they carry on the negotiation will be marked as in disagreement and a CCMCatalog arbitrator is created. It will investigate the data available to see how closely the robots agree and if they are close enough it will make a choice of a winner and tell everybody that they are roughly in agreement using a consensus estimate. If the robots now agree they can end the negotiation and accept the winner. If they continue to disagree or the data is not close enough the arbitrator will look for the maximum consensus and remove outliers from the negotiation and repeat the consensus-based negotiation. The robots will be informed and again they can chose to agree or not. If this does not work a non-consensus-based winner will be chosen and the negotiation will finish. Robots can then choose what to do with this information.

Operator Views

The CCMCatalog offers a rich API for an external operator to query the information of a system. It can provide a Web browser view of mapping and object information including identities and object locations within the map, but also supports the same custom query structure as above, available as Web APIs to any third party software which supports HTTP queries.

Operator Interaction

Both the Web interface and the Web API supports the operator pushing information and events into either the CCMCatalog and the individual Psyclone systems. Such information could be to offer corrections of erroneous data such as location of objects or the fact that a human has entered the scene which the robots didn't discover yet. Such information can either be used directly by the robots or be recorded as passive data events for subsequent analysis of performance and accuracy.

APPENDIX 1: Details of Current State of Implementation

The state of the CCMCatalog and the associated components at the time of the first draft release is as follows.

CCMCatalog and CCMProxy
The initial version contains all the features described above and works as a standalone component with which the individual systems connects to via the CCMProxy catalog. The network communication is a little bit cumbersome and we plan to revisit this before the final release. The CCMCatalog currently supports all the observation types needed for the project such as Location, Dimension, String, Integer, Float and Time and all can be stored and queried fully with each observation object able to deal with the query in its own custom way. Currently the modules communicating with the local CCMProxy catalog all need to provide the system ID which we plan to centralise for ease of use. Short-term observation storage is complete, but the long term summarised storage to disk still needs some work.

CCMCollector
The CCMCollector catalog is able to pick up data flowing in the local system and convert these to observations to be sent to the CCMCatalog via the local CCMProxy. It supports dynamic conversion of data entry names and types such as mapping from a data entry called ‘xpos’ of type string to the observation data entry ‘x’ of type integer. This mapping information can be specified at runtime. At the moment this component needs to know the system ID of the robot – we hope to move this to the CCMProxy shortly.

Negotiators
Currently, data and task negotiation is done using a custom module and the plan is to allow the robot modules themselves to negotiate using a more versatile API. Negotiation right now mainly covers observations, tasks work reasonably well, but need more work and we need this in place for Demo 1.

Next steps
The project is moving rapidly towards Demo 1 which involves all areas of the CCMCatalog, CCMCollector and Negotiators. Most of the outstanding tasks described in the previous section will need to be completed before the demo and we will undoubtedly find more areas needing work as we start to exercise the system properly.