Implementation of the Collaborative Cognitive Map Architecture

CoCoMaps Deliverable T8.D2

Author: Thor List, CMLabs
Created: January 2018
Last modified: February 2018

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

This is Deliverable T8.D2 *of the* CoCoMaps *project, which is funded by* ECHORD++ *through the* European Union's 7th Framework Programme for Research, Technological Development and Demonstration*, under grant agreement no.* 601116.

Contents

Executive Summary

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 system is comprised of subsystems and a middleware that allows flexible construction of robots for multimodal interaction and group collaboration. This report details v 1.0 of the CCM (and its key component CCMCatalog) which enables two robots to collaborate with two humans on completing a single task composed of several sub-tasks. A key software component for enabling these new facilities is called the CCMCatalog, whose constituents comprise the bulk of this report.

Appendix 1 describes the prior release of the CCM; Appendix 2 gives some lower-level technical details on the final CCM release system.

This Report is submitted in partial fulfillment of the requirement of a Final Implementation of the new CCM, as detailed in the *CoCoMaps Proposal*.[1]

Introduction

The main goal of the CCM is to enable the construction of robots that can collaborate with humans and other robots to achieve tasks. The CMA was not intended for multiple robots or multiple people and had no facilities to manage such control. The new Collaborative Cognitive Map (CCM) architecture has a number of features that bring collaborative capabilities to robots in a way that makes its construction more structured and manageable. The new capabilities allow the robots to share and collaboratively use observations as well as negotiate about tasks and roles. If any disagreements arise the CCM architecture allows them to attempt to reach a consensus and end up agreeing on all, some or none of the shared information.

[1] Full name of the proposal is "Collaborative Cognitive Maps", authored by Thor List of Communicative Machines Ltd. (UK) and Kris Thórisson of the Icelandic Institute for Intelligent Machines (Iceland).

The CCM is based in part of an earlier system called the Cognitive Map Architecture (CMA)[2] and inherits a lot of its foundational principles. This includes a constructionist AI methodology that derives from behaviour-based robotics (Brooks 1991), but is extended with principles that allows integration of higher-level cognitive skills such as dialogue (Thórisson et al. 2004, 2005a, 2005b). However, the CMA was not intended for multiple robots or multiple people and had no facilities to manage such control, which is the main focus of the CCM.

This report details the first full release of the CCM architecture. It contains several enhancements and updates to the original draft release and is now ready for the final demonstration of the project as well as for collaborators to download and use. This report assumes that the reader is familiar with the earlier reports.

Previous Systems & Reports

Two previous reports detailed the initial reimplementation of the original Cognitive Map Architecture, completed in July 2017[3], and the first draft release of the multi-robot Collaborative Cognitive Maps architecture[4] which was completed in November 2017 and subsequently used in the first demonstration deliverable, Demo 1. Appendix 1 describes the state of implementation at draft release (November 2017) and Appendix 2 gives a description of technical changes since that system, as of the final CCM release.

Putting Collaboration into the Cognitive Map Architecture

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.

Having a single robot interact with a single human user is considerably simpler than having multiple robots interact with multiple humans. We have demonstrated the latter capability in the CMA for simple tasks where negotiation of roles and responsibilities is in part

[2] The Cognitive Map Architecture (CMA) was developed in collaboration with Honda Research Institute in California for the purpose of adding learning and human interaction to their ASIMO robot (cf. Ng-Thow-Hing et al. 2009, 2007). It was implemented in Psyclone v.1 (cf. List et al. 2005, Thórisson et al. 2004, 2005a, 2005b) and ran a combination of C++, Java and LUA code. It had no facilities or mechanisms for supporting multiple humans or multiple robots.

[3] The CoCoMaps report *State-of-the-art implementation of the CM Architecture* detailed the upgrade of this earlier work (cf. Ng-Thow-Hing et al. 2009, 2007, List et al. 2005) to produce a current and viable implementation, in preparation for moving this into the realm of collaborative robotics.

[4] The CoCoMaps report *Draft implementation of the new CCM Architecture* described the first release of the multi-robot Collaborative Cognitive Maps architecture, subsequently used in Demo 1 of the project.

determined by multimodal interaction using dialogue. To do so we have built several new subsystems that together enable the robots to do so.

A key software component for enabling the new multi-modal, multi-party facilities is called the CCMCatalog, whose constituents comprise the bulk of this report. First we give an overview of this component, and then we delve into details for each of its main constituents.

CCMCatalog Overview

A major new component in the CCM architecture is the CCMCatalog. It 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.[5] 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.[6]

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

[5] External programs to access it as well through its API, extending its utility well beyond the core programs that are directly part of the CCM. This can be useful, for instance, when the robots must read or write to external systems, and is envisioned as the main method for extending the task list of robots beyond their immediately sensable surroundings.

[6] In this latter case the object category comes from a pre-defined set of object classes the robots can identify, but the shared object identity has a unique identifier akin to Agre & Chapman's deictic reference system (1990).

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' tasks and subtasks are also synchronised and negotiated via the CCMCatalog. For instance, 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:

- Maps
- Objects & Identities
- Storing Observations

- Retrieving Observations
- Tasks & Task Negotiation
- Consensus-Based Negotiation
- Operator Views
- Operator Interactions

Each of these is described in order. The following definitions specify key concepts in the below description:

- **Observation:** A "snapshot" of data recorded by a robot. Robots record observations regarding all important entities in their task environment. A *confidence value* is always computed as part of any observation to be used in negotiation.

- **View:** A single observation at a single instance in time. A *raw view* is an unedited observation from an observer. In the case of negotiations, a suggested consensus is a "synthetic" view based on extrapolated versions of several observations, often from more than one robot, usually based on an average of the available raw observations. When a view is queried (via) one can choose to search in the raw observations from one observer (robot), from all observers or a negotiated views – the result is the same, a set of observations as a reply.

- **Consensus:** The CCM provides a facility for robot negotiation – any observation that two or more robots have confirmed they agree with is a consensus-based observation. These help improve reliability of robot sensing.

- **Objects, Identities:** Objects are detected objects (humans, robots, etc.), identities are the particular instances of an object, e.g. human, robot, etc. – typically identified via a name or some other unique identifier.

Maps

"Maps" are a software component that stores coordinate data, including dimensions and resolutions in metres, plus image visuals of floorplans, etc. The CCMCatalog can have a number of maps of different types, each with specifications on how they relate to each other. For instance, a map could store a building's schematics and related ones could be more detailed maps of each room. Each robot also has its own map of its proximal environment and these are coordinated with the CCMCatalog based on observations (see below). 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

"Object" is a fundamental knowledge entity in the CCMCatalog that allows an observation about any object can be entered into the CCMCatalog, even when its identity is still

unknown. However, to be created an object must have a type.[7] An object can be assigned any known type, and new objects can be created on the fly. Standard types include *human* and *robot*, but the robots can at any point decide to create a new object such as chair or table.

The "identity" of an object refers to a particular instance of an object – e.g. the name of a person. Identities for objects can be added to any already-created object by any robot, and each robot can in principle associate its own view to an observed identity that is different from someone else's; i.e. the robots don't initially have to agree. Once the identity of an object has been agreed to (by both robots) this identity will be associated with the object for past observations as well – i.e. backwards in time.

Storing Observations

An "observation" is a data entry done by a robot for the purposes of recording data gathered through its sensors. Observations of a number of types can be collected and stored in the CCMCatalog. New observation types can be added by a CCM designer, and each can be provided with its own algorithm(s) for comparison and averaging so as to be used in negotiation (see below).

A *Location* observation, for instance, contains the *x* and *y* coordinates of the object's centre point in a specific map, and also an (optional) width-height estimation. Other types of observations are *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 subsystem connected to the CCMCatalog can retrieve and query observations using a rich query specification through the CCMCatalog query API. This includes querying observations by type, time period and system ID. This would be needed by a robot, for example, when the location and identity of a human is necessary information for its task planning. In that case they would query the current position of the human to see if the other robots agree with its assessment of the human's position, or whether the position may be disputed (in disagreement - see "Consensus-Based Negotiation" below). If so, the robot can choose to do some more observations, to increase its own confidence, before using the value, or use consensus observations that other robots have agreed on. Querying for consensus status on observations is generally be more valuable when a robot's own

[7] This limitation will be removed in the future as the system is extended, as it is not inherent to the algorithms but rather a an unimplemented feature.

observations of critical data has low confidence, so thresholds for deciding when to look at consensus data tend to be set to guarantee this for all low-confidence observations.

Each observation type will handle querying of values *from*, *to*, and *between,* which for location observations could mean an *x,y* square, but for other types of observations could mean something different. The system is set up so custom and more complex observation types can be changed, added, and modified at any time by CCM users, through a consistent uniform method to enter, store and query it, through currently available mechanisms.

Tasks & Task Negotiation

A "task" in the CCM is a tree whose sub-nodes are finer-grained sub-tasks and whose complete structure contains the full definition of a task, its potential states, and resolution methods.[8] Tasks are created at runtime based on stored templates constructed by a human task designer; a task's executor is negotiated based on roles and/or the system IDs. Task trees are created by programmers and stored as JSON structures. Tasks can be activated as a configuration at system startup or dynamically during the runtime of the robot, based on external events such as a user asking a robot to perform a particular task. Tasks can also be added by one of the participating systems.

A task's sub-tasks each need to be completed in turn in order for the task as a whole to be marked as "done". Each subtask can be "grabbed" (select for execution) by any of the participating robots and marked as "in progress", at which point no other robot can grab that particular task.

If two robots try to grab a task before the allocation has been confirmed a negotiation will start where each robot needs to enter a *confidence* level with which they believe that they can complete the task. The outcome of a negotiation is that one of the robots is assigned the task; if no such conclusion is reached (within pre-determined time limits) the task will be marked as "in disagreement" and an arbitration sub-process of the CCMCatalog itself will kick in to help the robots assign the task to a single robot.

Any robot can at any point choose to disregard the negotiation process and mark the task as "in dispute"; other robots will then decide for themselves whether they continue with the task or not. In any case this would be a design decision that must depend in large part on the nature of each particular task and sub-task.

Consensus-Based Negotiation

When two or more robots disagree on either observations or task allocation they enter into a *managed negotiation phase* which may result in a settled agreement or may result in a

[8] Currently serial sub-nodes are supported; future versions will allow parallel sub-tasks to be represented and executed by the robots, as well as alternatives where particular algorithms are used to select which one based on values of specified parameters at the time of selection, in the environment, robot, and state of the task.

decision to disagree. (What the robots do outside of this phase is decided independently by each robot.) The negotiation is based on a number of steps and robots can choose to skip any step or exit the negotiation at any time.

Observations are related to each other by the topic (object type or entity ID) and the time that the observation was made. All observations added by any observer in the CCM is a candidate for negotiation – nothing is excluded.

Observations are entered by type, so for example observations about the location of objects can be compared and averaged in one way, whereas numerical or string observations are compared (and "averaged") in a different way.

To negotiate about tasks, first each robot needs to provide a confidence level which tells the other robots how confident this robot is that whatever they are stating or requesting is best. In the case of a human, if the human's face is visible in e.g. both the USB and the depth camera images the location confidence is very high (>90%). If the location is based on finding the torso, the confidence is a bit lower and if it is based on finding the legs it is lower again. Other observations can have other confidence algorithms, but they are always determined at the time of observations and computed by the observers themselves.

Second, once all the confidence estimates are in, the robots choose to either accept the negotiated observations or carry on the negotiations because they choose not to agree. If they carry on the negotiation will be marked as "in disagreement" and a CCMCatalog arbitrator object is created, as mentioned above. This is a static object to be used by the negotiators in the further negotiation process. Using available data, they will decide how closely the observations agree and if they are close enough (using pre-determined threshold for various data types) the arbitrator will compute a suggested consensus and inform everybody that they are in agreement if they use that consensus estimate. The suggested consensus is usually a time-observant averaging across the available data. Each observation type will still have its own algorithms for averaging values, for e.g. location-based observations this relates to the *x*, *y* and *z* coordinates provided, as well as the 3D distance to other observations.

If the robots now accept the suggested consensus the negotiation ends. If they continue to disagree, e.g. because the data still doesn't match their own thresholds, the arbitrator will enter into the next phase where it looks for the maximum consensus and remove outliers from the negotiation and repeat the consensus-based negotiation. The robots will be informed and again and they can chose to accept or reject the new recomputed consensus. At this point the negotiation will finish and the robots can choose individually what to do with this information.

Operator Views

The CCMCatalog offers a rich API for an external operator to query the information of views in the system. It supports 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 CCM Web interface and the CCM 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: Previous State of Implementation

(Draft Release of CCM)

At the time of the first draft release of the CCM (Draft Implementation of the CCM, Deliverable T8.D1) it contained an early version of the *CCMCatalog*, *CCMProxy*, *CCMCollector* and *Negotiator*. What follows is a short description of each at the time of the draft release.

CCMCatalog and CCMProxy

CCMCatalog: The main storage and querying device in CCM. See further details in this document.
CCMProxy: Catalog which Psyclone systems use who do not themselves have the CCMCatalog. Implements a CCMCatalog API for systems that run more than one CCM. All modules in a Psyclone system will talk to the Proxy and think they talk to the CCMCatalog.

The initial version of the CCMCatalog and CCMProxy contained 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 was a little bit cumbersome and we planned to revisit this before the final release. The CCMCatalog supported all the observation types needed for the project such as Location, Dimension, String, Integer, Float and Time and all could be stored and queried fully with each observation object able to deal with the query in its own custom way. Then the modules communicating with the local CCMProxy catalog all need to provide the system ID which we planned to centralise for ease of use. Short-term observation storage was complete, but the long term summarised storage to disk still needed some work.

CCMCollector

A Catalog which translates between system-specific data formats (e.g. x-pos = 10) and CCM-specific format (e.g. x = 10), so the CCM architecture can be embedded into any existing robot architecture or system without having to change the existing modules/software definitions and names of variables. Any data it sees will be posted to the CCMCatalog automatically as CCM observations, using the translation mechanisms provided.

The CCMCollector is a catalog that collects data flowing in a local CCM system and converts it to observations for the CCMCatalog via the local CCMProxy. It supported 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 could

be specified at runtime. Back then this component needed to know the system ID of the robot – we hoped to move this to the CCMProxy.

Negotiators

Also back then, data and task negotiation was done using a custom module and the plan was to allow the robot modules themselves to negotiate using a more versatile API. Negotiation mainly covered observations and tasks worked reasonably well, but needed more work before Demo 1.

APPENDIX 2: Technical Details of Full Implementation

(February 2018)

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

CCMCatalog and CCMProxy

CCMCatalog: The main storage and querying device in CCM. See further details in this document.
CCMProxy: Catalog which Psyclone systems use who do not themselves have the CCMCatalog. Implements a CCMCatalog API for systems that run more than one CCM. All modules in a Psyclone system will talk to the Proxy and think they talk to the CCMCatalog.

The initial version contained all the features described in this document, but several issues and additional requirements were identified and added for the final version. More observation types have been added for the purpose of storing information about human facial expressions and communication, as well as the state of the current and past dialogues.

Network communication has now been streamlined and works much faster, including serialisation and reconstruction of observations. The CCMProxy now keeps track of all the system specific information such as the system ID, so that no other component has to know this.

Long-term summarised storage to disk is now complete.

CCMCollector

CCMCollector: a Catalog which translates between system-specific data formats (e.g. x-pos = 10) and CCM-specific format (e.g. x = 10), so the CCM architecture can be embedded into any existing robot architecture or system without having to change the existing modules/software definitions and names of variables. Any data it sees will be posted to the CCMCatalog automatically as CCM observations, using the translation mechanisms provided.

The CCMCollector is now simpler and faster as the Proxy has taken over a lot of the communication tasks. Multiple collectors can be created now, each focusing on different types of data, coexisting nicely with each other and the Proxy.

Negotiators

Data and task negotiation has been decentralised so this can be done from the modules that need to do this directly. This makes the architecture easier to follow and less data needs to flow needlessly around the system just for the purpose of negotiation.

Negotiation now covers observations, tasks and dialogue progress.

Roles and Tasks

A set of roles can now be specified which the robots can assume at different times during the system execution. Some can be exclusive so only one robot can be in that role at any given time, some can be share so multiple robots can have this role at the same time and some again can be complementary so if one robot takes on one role the other robot(s) will be assigned to this other role for the duration of the task. The roles and assignments are managed by the *RoleNegotiator* modules in each robot.

Tasks are created dynamically and can either be left for other robots to sign up to or be automatically assigned to a robot by UID or by current role. When a task is created, assigned, accepted, updated and complete (either successfully, by timeout or by failure) all robots are notified and the *TaskNegotiators* and *TaskExecutors* in each robot will deal with these events and keep track of all roles. The TaskExecutors will, when assigned a task, accept it and send out messages for other modules to perform the actual task. Two or more robots can accept the same task if it was created as a shared task, but if the task is an exclusive task only one robot will be allowed to accept it, based on simple temporal negotiation. In the future a more advanced confidence based task acceptance negotiation is planned.

REFERENCES

Agre, P. & D. Chapman (1990). What are Plans For? *Robotics and Autonomous Systems*, **6**:17-34.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, **47**:139-159.

List, T., J. Bins, R. B. Fisher, D. Tweed & K. R. Thórisson (2005). Two Approaches to a Plug-and-Play Vision Architecture – CAVIAR and Psyclone. In K. R. Thórisson, H. Vilhjalmsson, S. Marsella (eds.), AAAI-05 Workshop on Modular Construction of Human-Like Intelligence, Pittsburgh, Pennsylvania, July 10, 16-23. Menlo Park, CA: American Association for Artificial Intelligence

Ng-Thow-Hing, V., K. R. Thórisson, R. K. Sarvadevabhatla, J. Wormer & T. List (2009). Cognitive Map Architecture: Facilitation of Human-Robot Interaction in Humanoid Robots. *IEEE Robotics & Automation Magazine*, March, **16**(1):55-66.

Ng-Thow-Hing, V., T. List, K. R. Thórisson, J. Lim & J. Wormer (2007). Design and Evaluation of Communication Middleware in a Humanoid Robot Architecture. IROS 2007 Workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware, Oct. 29, San Diego, CA.

Thórisson, K. R. O. Gislason, G. R. Jonsdottir & H. Th. Thorisson (2010). A Multiparty Multimodal Architecture for Realtime Turntaking. Proc. Intelligent Virtual Agents (IVA '10).

Thórisson, K.R., T. List, C. Pennock & J. DiPirro (2005a). Whiteboards: Scheduling Blackboards for Semantic Routing of Messages & Streams. K. R. Thórisson, H. Vilhjalmsson, S. Marsella (eds.), Proc. of AAAI-05 Workshop on Modular Construction of Human-Like Intelligence, Pittsburgh, Pennsylvania, July 10, 8-15. Menlo Park, CA: American Association for Artificial Intelligence.

Thórisson, K. R., T. List, C. Pennock, J. DiPirro & F. Magnusson (2005b). Scheduling Blackboards for Interactive Robots. Reykjavik University School of Computer Science Technical Report, RUTR-CS05002.

Thórisson, K. R., C. Pennock, T. List & J. DiPirro (2004). Artificial Intelligence in Computer Graphics: A Constructionist Approach. *Computer Graphics Quarterly*, **38**(1):26-30. New York: ACM SIGGRAPH.