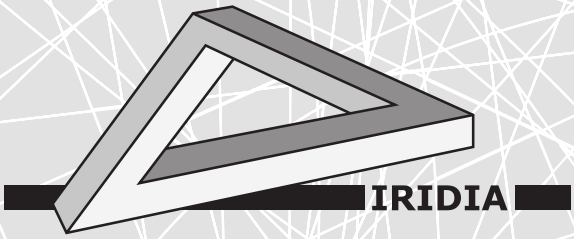

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

IRIDIA's Arena Tracking System

A. Stranieri, A.E. Turgut, G. Francesca, A. Reina,
M. Dorigo, and M. Birattari

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2013-013

November 2013

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2013-013

IRIDIA's Arena Tracking System

Alessandro Stranieri alessandro.stranieri@gmail.com
Ali Emre Turgut ali.turguti@gmail.com
Gianpiero Francesca gianpiero.francesca@ulb.ac.be
Andreagiovanni Reina areina@ulb.ac.be
Marco Dorigo mdorigo@ulb.ac.be
Mauro Birattari mbiro@ulb.ac.be

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium.

November 2013

1 Introduction

This document describes the multi-camera tracking system developed for the IRIDIA laboratory, which is targeted for multi-robot experiments in large experimental set-ups. The system is deployed in a large room exclusively dedicated to this kind of experiment, and we refer to this room as the Robotics Arena. The tracking system here presented is referred to as the Arena Tracking System, or ATS.

The purpose of this document is to provide an overview of this system, in terms of which hardware has been employed, which software has been developed and what use cases are offered to the user.

The first section is dedicated to the description of the hardware of which the whole system consists and how the experimental area has been set-up. The second section presents the software architecture of the system. The reminder of this Introduction is dedicated to the motivations that lead to the development of this system.

1.1 Motivations

One of the main focus of the research carried out at IRIDIA laboratory is on Swarm Robotics and on more general multi-robot applications. Experiments in Swarm Robotics involve large number of robots that navigate through the environment, sense it and act on it.

The initial purpose for the development of the tracking system here presented, was to provide a tool that allowed a researcher to record the state of the experiment throughout its complete execution. By state here, it is intended the position of all individuals in a robotic swarm at a particular time, with respect to a global frame of reference. As robotic swarms consist of several robots that can navigate throughout a large area, the main requirements of this system

Figure 1: Prosilica GC 1600C

were: 1) provide the possibility to follow the evolution of the experiment across a large area; 2) record the state of the experiment with a good time resolution.

During the development of this system another possible target application for such system has been conceived. Provided access to a mean of communication with the robots composing the swarm, such system could give the possibility to implement virtual sensing. This means that as a robot navigate through the environment, the system actually virtualizes fixed objects that a robot ought to avoid, or non-fixed objects that a robot could virtually move throughout the environment. This last potential objective is for the moment solely a speculation and is under consideration for development.

2 Hardware

The ATS is composed of a set of cameras whose collective field of view covers the entire Robotics Arena. The cameras feed images through an Ethernet connection to a dedicated computer, the Arena Tracking System Server, located outside the Robotics Arena. This computer hosts and runs the API that a researcher can use to record an experiment. This section is dedicated to the description of the Robotics Arena set-up and the hardware on which the ATS is based.

2.1 Cameras

The Robotics Arena is a large room around $990cm$ large and $705cm$ deep, which can host multi-robots experiment across the whole area, or even two experiments in parallel, if divided along the shorter side.

While an experiment is in progress, its State computed by applying recognition and decoding steps on a set of images acquired by an array of cameras deployed on the Robotics Arena's ceiling. This array consists of 16 cameras, arranged in a 4-by-4 matrix. The arrangement of the cameras was drawn in order to have full coverage of the experimental area, and it depended on the area to cover, as well as the cameras' specifications.

For the choice of the camera, we opted for a *Prosilica GC1600 C*, manufactured by Allied Vision Technologies and shown in Figure 1. This particular camera has been chosen as it presented a favorable compromise between resolution, focal length, ease of interfacing and speed of transmission. Some of the technical specifications of the camera are given in Table 1.

The positioning of each camera within the Robotics Arena has been determined following these criteria: 1) The maximization the area of the Robotics

Arena covered by the field of view; 2) Overlap of the field of views of two adjacent cameras; 3) Resolution useful for symbol decoding on the plane occupied by the top of the robots.

The final layout of the cameras with the distances between each camera is presented in Figure 2. Each camera has been placed on a wooden structure specifically manufactured and at a height of 243cm from the ground. With this configuration, each camera covers an area of about 350x260cm, with a spatial resolution on the ground of about 4.6px/cm

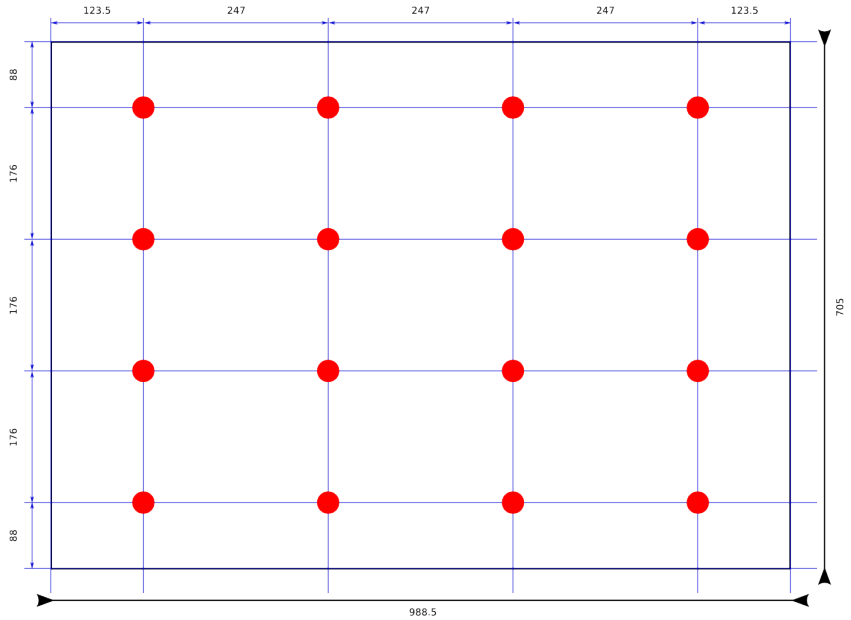

Figure 2: Layout of cameras (distances in cm)

2.2 Network configuration and Computer Host

The cameras employed in the Arena Tracking System transmit the acquired images through an Ethernet interface linked through a 1Gbit connection to a dedicated switch installed in the Robotics Arena, which mounts a module for 10Gbit/s connection to the Arena Tracking System Server. The Arena Tracking

Prosilica GC 1600	
Interface	IEEE 802.3 1000baseT
Resoultion	1620 x 1220
Max frame rate	15 fps
Color modes	Gray Scale, RGB
Dimensions ((L x W x H in mm))	59x46x33
Mass	97 g

Table 1: Camera Specifications

System Server hosts 16 Intel® Xeon(R) CPU E5-2670 at 2.60GHz. Each core features the Intel® Hyper-Threading Technology, which enables it to run two threads per core. The Operating System is GNU/Linux Ubuntu 12.04.1 LTS for 64 bits architectures.

3 Arena Tracking System API

The use of the ATS is based on an API whose main current purpose is to allow a user to easily record and extract the state of a robotics experiment while this is in progress. This API offers the possibility to configure different aspects of a tracking session, such as the cameras to be used, the objects to detect and the global frame of reference to in which the robots' positions have to be computed. This section provides a brief overview of the API.

3.1 Architecture

The Arena Tracking System's API is built on top of the QT Framework and the Halcon library [2]. Halcon is a library by MVTec Software GmbH, used in for scientific and industrial computer vision applications. It provides an extensive set of optimized operators and it comes along with a full featured IDE that allows for fast prototyping of computer vision programs, camera calibration and configuration utilities. The library also provides drivers to easily interface with a broad range of cameras. The current architecture of the API is presented in Figure 3 and consists of two modules: Core and Tracking. The Core module provides the data types that are related to the representation of a tracking session state, whereas the Tracking module provides the tools to configure a tracking session and to extract the experiment state instances.

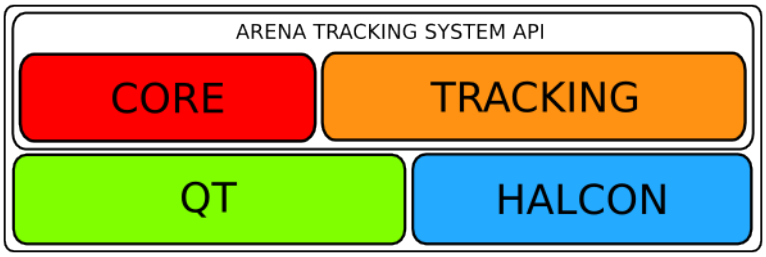

Figure 3: Arena Tracking System API

3.2 The ArenaState Class

The use of the Arena Tracking System API revolves around two main classes: ArenState and ArenaStateTracker.

Given a running experiment that involves several robots navigating across the Robotics Arena, the ArenaState type represents the state of the experiment at a given time step. Figure 4 provides a simplified view of this type, and its components. An object of the ArenaState class contains all the elements detected at the timestep of creation. Each element is represented by its own ID,

the position in the image where it was acquired and the position in the world frame of reference. There are two ways of accessing the elements detected. The method GerArenaElements gives a simple list of all the robots detected across the whole Experimental Arena. The other possibility, is to access the detected robots under a specific camera, by requesting access to the camera's specific tracker with the GetCameraState method.

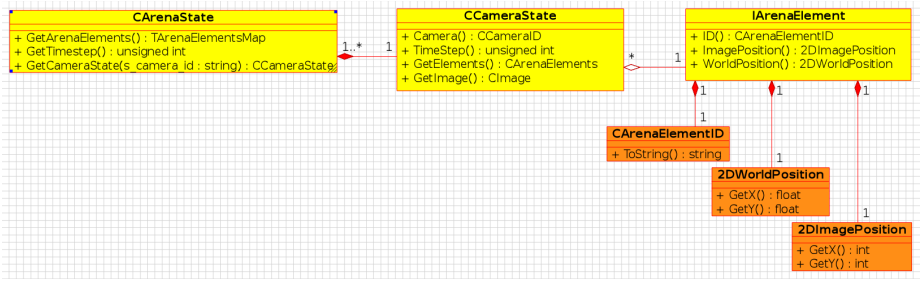

Figure 4: ArenaState class diagram

3.3 The ArenaStateTracker Class

A new instance of the ArenaState class can be created from the GetArenaState method of the ArenaStateTracker class. This class represents the configured set of resources that make up for the particular instance of a tracking session. Figure 5 contains the class diagram of the ArenaStateTracker class and its components.

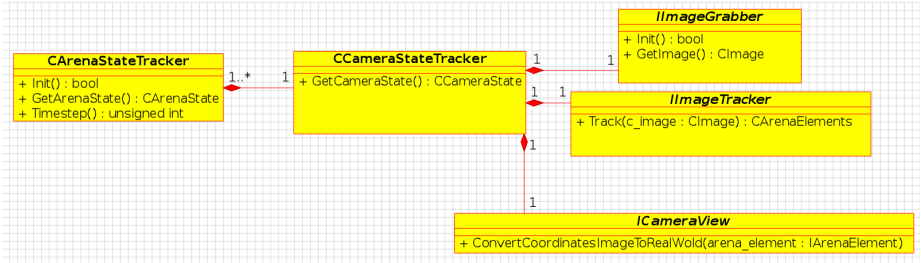

Figure 5: ArenaStateTracker Class Diagram

The ArenaStateTracker's job is to configure and orchestrate a set of CameraStateTracker instances. Each CameraStateTracker is associated to an image source, such as a camera, and the operators to detect the imaged robots. It is basically a virtual device that performs all the steps(see Figure 6) to output a sequence of elements within a specific field of view. A CameraStateTracker is composed of an ImageGrabber, which encapsulates the logic to configure an image source and acquire an image; an ImageTracker, which is responsible for detecting the robots in the input image, decode their ID and output a list of objects that represent them; a CameraView, whose job is to convert the image positions of a list of detected objects into the coordinates of a world reference system, given the external and internal parameters of a particular camera.

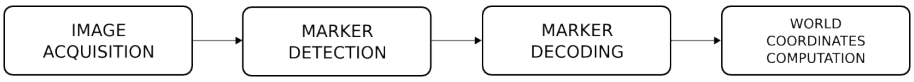

Figure 6: CameraStateTracker flow

3.4 Detection

As mentioned in the previous section, an instance of the ArenaStateTracker class applies a detection and decoding operator on the images acquired at a given timestep. In order to detect the robots in action during an experiment while they are navigating, we designed a type of marker that allows for easy detection and easy decoding even when the image is relatively small. An example of the marker is depicted in Figure 7. These markers are applied on top of each robot involved in an experiment(see Figure 8 for an example with e-puck robots [1]) and carry their ID encoded in binary as matrix of black and white cells. Each time the detection step is performed, each frame acquired by the ArenaStateTracker is scanned for occurrences of the marker. Each time a marker is detected, the inner matrix is decoded and converted to an ID.

Figure 7: Marker Example

Figure 8: Markers on e-puck robots

3.5 Configuration

Other two important classes of the API are ResourceManager and ExperimentManager. These two classes are fundamentally responsible for declaring the set of resources that a user wishes to have available for image acquisition and marker detection, and for configuring a tracking session. Instances of these classes can load a tracking session's configuration from XML files.

The ResourceManager class can be used to read a XML file containing the definition and the parameters of cameras, detectors and image transformations available for the session. Each of these elements is mapped to a unique ID, which can be used to acquire the object which controls it.

The ExperimentManager is instead used to read an XML file that describes the configuration of the ArenaStateTracker, in terms of its components. The user who wishes to configure an ArenaStateTracker instance has to simply specify in the XML file how many CameraStateTracker object compose the ArenaStateTracker and, for each of them, specify the IDs of the resources they should use.

4 Concluding remarks

This document is not the final version and it will be updated along with the technical development of the Arena Tracking System.

References

[1] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stéphane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.

[2] MVTech Software GmbH. Halcon library website. URL `http://www.mvtec.com/halcon/`. Last checked on November 2013.