Diplomarbeit:
TAOs - Tangible Active Objects for Table-top Interaction

Bielefeld, June 2, 2009

Eckard Riedenklau

Supervisors:

Dipl.-Mus. Tobias Großhauser (Ambient Intelligence Group)
Dr. Thomas Hermann (Ambient Intelligence Group)
Prof. Dr. Helge Ritter (Neuroinformatics Group)

Diplomarbeit:
TAOs - Tangible Active Objects for Table-top Interaction

Eckard Riedenklau

June 2, 2009

Personal Computers have arrived in almost every part of our live to do work faster and better. They are used for writing texts, creating music or drawings, or simply organizing and guiding everyday tasks. Nearly all these tasks are done with computers which are operated using screen, keyboard and mouse even if their use may be sometimes cumbersome or even unsuitable for some tasks. Human Computer Interaction (HCI) aims to analyze the way people use computers and suggest new methods for interaction. One area of this research field is called 'Tangible Interaction'. Tangible Interaction tries to use everyday objects as tangible representations for digital data. It is hoped that by pulling the data into the tangible real world (in contrast to the virtual world) they can be made more vivid and graspable and thereby better understandable. These real-world representations are called Tangible User Interface Objects (TUIOs) and the systems they are used in Tangible User Interfaces (TUIs).
The main goal of this work is to create active objects as a new kind of TUIO. These active objects extend the concept of TUIOs by the possibility to be not only manipulated by the user but also by the computer. Many different ways of manipulation are possible, e.g. adding LEDs or liquid crystal displays, sound output or tactile and haptic feedback with vibration, etc. One of the most challenging manipulation possibilities is computer controlled planar movement for instance on a desk surface, which will be developed in this work. The developed objects are constructed as modular as possible to be open for future extensions and modifications. A software structure for the coordination of the objects is implemented. Furthermore some applications shown to give examples for the potential of this novel technique.

Contents

1 Introduction

Before computers became pervaded in nearly every part of our live, people were used to do their daily work manually. Whatever task needs to be done, humans are very dexterous in creating, changing and exploring things by hand. Interestingly, the term 'graspable' carries the meaning of understandable. Computers are designed to process data, and since data are represented in the memory of the machine they become virtual and lose any physical graspability. Because virtual data are all abstract, researchers in the field HCI started to give virtual data physical representatives, that can be used as a handle for these data. 'Tangible Interaction' is the branch in HCI research concerned with these types of physical interfaces, and it is a steadily growing research focus. There are different approaches to make abstract virtual data graspable. Table-top TUIs are one of these approaches. Researchers use handy physical objects on table-top surfaces as representatives for digital data. These objects are tracked by a computer and the position and orientation becomes available to manipulate and interact with data. A quite prominent example is the reacTable [JKGB05]. It's data model generates sound and the tangible objects represent different sound generating and processing items (e.g. oscillators, filters, etc.). The spatial relations between the objects parameterize the configuration of the data model, which results in changing sound output. The researchers of the Tangible Media Group of the MIT introduced the actuated workbench [PMAI02], which is capable of actively moving the physical objects remotely. A special surface containing computer-controllable electro-magnetic coils makes this capability possible. Ferro magnetic objects can be moved on this surface as tangible representatives of virtual data. The capability of moving TUIOs around remotely allows novel interaction possibilities. It is possible to obtain relations between objects. This means that when one object is moved by the user, other objects can be moved by the computer simultaneously. It is also possible to represent dynamic scenarios.
In other environments, such as for instance on multi-touch desks or for tangible interfaces on a regular computer desk, it remains a problem that the state of tangible objects can not be restored on initiative of the computer. The lack of the possibility to save and restore an interaction state is a problem for many TUIs. This problem can be solved by introducing actively moving automotive TUIOs to already existing systems.
In this work, such Tangible Active Objects (TAOs) are developed, built and tested. They are designed for operation on a glass table, to be tracked from beneath by a

video camera and controlled by RF-signals. TAOs enable new tangible interfaces, mixed-initiative interaction between the human and the computer. They even enable the computer to manipulate passive objects on the surface in tabletop user interfaces. This work implements TAO prototypes in a modular fashion so that a maximal freedom remains for future extensions of the platform.

1.1 Goals

The main goal of this work is to design and build Tangible Active Objects (TAOs), that can be remotely controlled to give feedback to the user. They are cubical-shaped with an edge length of 5 cm. Three to five prototypes are build during this work. Later even more TAOs are considered to populate the Tangible Desk (tDesk) (formerly known as the Gesture Desk (gDesk) [HHR04]), which glass surface measures 70 x 70 cm. The costs for a TAO sould not exceed €100 to allow the assembling of a small swarm of TAOs. To allow the objects to move freely on the desk, a small wireless mobile robotic platform gets integrated into the TAOs. The batteries should allow at least half an hour of operation. The microcontrollers of these TAOs need to be programmed to give meaningful response to commands transmitted to them and the TAOs have to be controlled remotely by connected computer-programs, which need to be developed. Therefore the subordinate goals of the thesis are to implement the following modules to operate TAOs; namely:

Vision Module The vision module tracks the positions of the TAOs on the glass surface of the tDesk. To start working on further software modules without having a complete hardware prototype, there is an additional simulation environment that generates the same output as the vision module and reacts on messages, the path planning module creates.

Path-Planning Module This module plans the paths of the TAOs, so that they reach their target positions and orientations without colliding with each other.

Serial Relay Another module relays the commands wirelessly to the TAOs.

Application Modules Applications need to be developed to demonstrate the practical use of the system. One application allows existing TUIs to save and restore configurations of TUIOs. Another application visualizes datasets by representing cluster centers with TAOs.

1.2 Structure

The thesis gives full account on the design, construction and implementation of the hard- and software and is structured as follows:

Chapter 2 gives a brief overview of the state of the art of TUIs with self-moving objects or systems with active feedback. The described systems have similar goals but implement the solutions in a different way. In addition this chapter presents systems that are technically interesting for the implementation of this work.

Chapter 3 introduces the concepts behind TUIs and TUIOs (sections 3.1 and 3.2). Section 3.3 presents the hardware components used in this work and section 3.4 describes the used software libraries and frameworks. Finally this chapter gives a short survey of path-planning algorithms and robotic mobile platform designs.

Chapter 4 describes the different hardware designs, that arose throughout the implementation. This includes a description of the assembling of a first prototype as well as a small range of TAOs. A closer look is taken at potential problems and sources of error.

Chapter 5 discusses the implementation of the software modules and explains their role in the system.

Chapter 6 presents the final applications for this system. Two applications are described that utilize the system for data exploration and extend existing TUIs.

Chapter 7 describes the evaluation of the system regarding to accuracy and velocity of the TAOs.

Chapter 8 summarizes the complete work and highlights the main insights gained during the development.

2 State of the Art

This chapter reviews the current state of the art of TUIs with self-moving objects or systems with active feedback. A selection of systems which use active objects that are output enabled are presented. Some of these systems are, for instance, equipped with a display or with speakers. Most of these systems address tangible interaction, some do not. Finally, a short paragraph summarizes the state of the art and the implementations for TAOs.

Planar Manipulator Display (PMD)

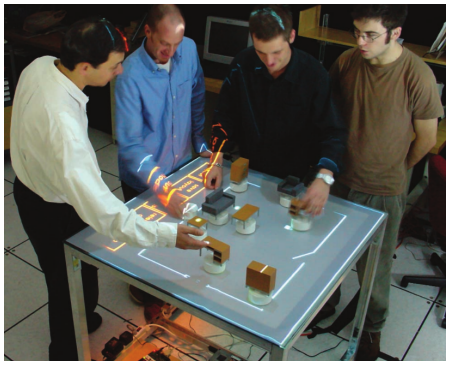
The authors of the PMD [RZSP04] (depicted in Fig. 2.1(a)) aim to use physical objects for HCI. They criticize standard computer interfaces and want to improve them by taking advantage of the human spatial awareness. The described objects in this work are remote controlled (via Infrared Data Association (IRDA)) small sized robots with a differential drive. They are used to carry models of furniture to simulate different configurations of furnishing inside a room.

The actuated workbench

The system introduced in the paper [PMAI02] utilizes active objects on table-top surfaces. It uses magnetic forces to move the objects on the surface (see Fig. 2.1(b)). This technique depends on a specially constructed surface, which allows non-vision tracking and magnetic movement of the objects in a smooth and quick manner. Only with this special surface the system works correctly.
The second paper [PI07] directs the reader's focus to physical constraints that can be applied to the system of moving objects by putting rubber belts or distance rings around them. This easily allows to obtain distances or relations between the objects. Because of the physicality of these constraints, they do not have to be modeled explicitly in software.

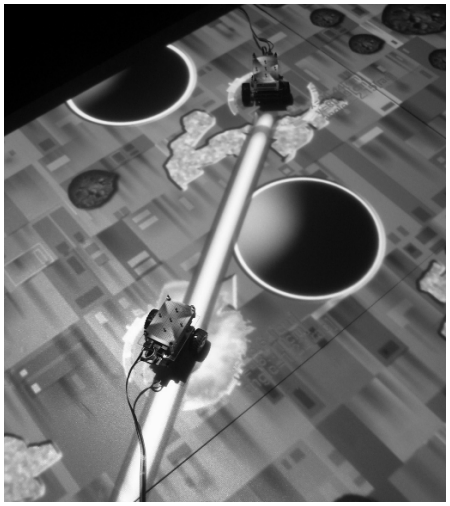
Augmented Coliseum

Figure 2.1(c) shows the Augmented Coliseum [KSN+06]. In contrast to the TAO system, Augmented Coliseum uses an active vision system for robot localization, which means that the used projection cannot disturb the recognition of passive

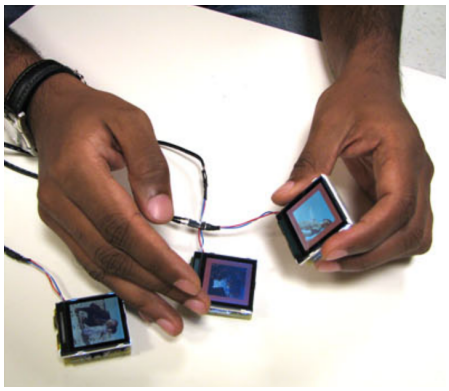


(a) PMD (graphic reproduced from [RZSP04] by courtesy the author)

(b) The actuated workbench (graphic reproduced from [PMAI02] by courtesy the author)



(c) Augmented Coliseum (graphic reproduced from [KSN+06])



(d) Prototype of Siftables (graphic reproduced from [MKM07] by courtesy the author)

Figure 2.1: Illustration of photos of systems that are the state of the art in active tangible objects and similar systems

markers. This system uses small sized robots on a projection surface to represent virtual objects in augmented reality scenarios through graspable real-world equivalents. Additionally the robots are wired, which facilitates control and power supply of the robots, which make the communication very robust, but restricts the freedom of movement. The developers intended to use the system as an augmented reality gaming environment.

Siftables

The Siftables system [MKM07] is a so-called Sensor Network User Interface (SNUI). The Siftables aim to create a novel TUI with active feedback (see Fig. 2.1(d)). The Siftables are equipped with a display, motion sensors (accelerometers) and wireless communication. They are used to give a real-world representation to digital data or digitized information. Furthermore the authors introduce gestural language primitives that can be used for interacting with Siftables.
In contrast to the TAOs, Siftabels are not modular and reconfigurable and they offer feedback exclusively via visual displays.

Curlybot

Curlybot [FSMI00] is an educational toy (see Fig. 2.2(a)), developed by researchers of the MIT Media Laboratory. The user can teach the robot trajectories by moving it around. The robot can replay this movement continuously. By attaching a pen to the robot, children can generate and explore complex graphical figures the robot draws by repeating simple trajectories again and again.

Block Jam

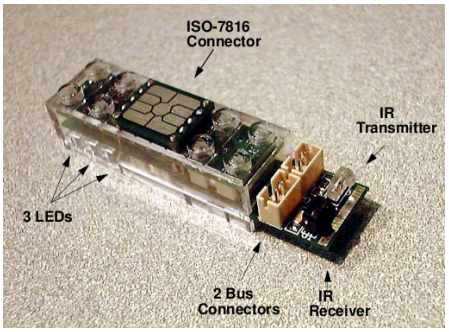
Figure 2.2(b) depicts Block Jam [NDNG03], a system consisting of blocks that represent different sound loops. By combining them to patterns, complex sound structures can be created. The blocks have a LED-Matrix display which allows them to give visual feedback and they can be called active. Additionally they are depressible (clickable) for input. Blocks can represent sound or functionality. By clicking a block, it is possible to choose its sound or functionality. They can be used for play-back routing and reflecting (like a wave that propagates through the pattern) and for control, similar to a play button.

Tangible Programming Bricks

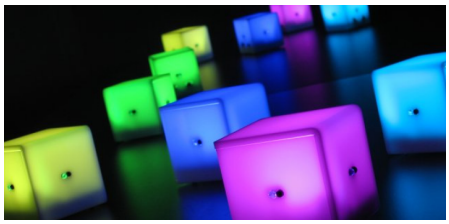
Tangible Programming Bricks [McN00] provide tangible building blocks (see Fig. 2.2(c)) as a simple programming environment. The purpose is to program electronic everyday objects such as kitchen appliances or toy cars. The aim of this

(a) Curlybot (graphic reproduced from [FSMI00] by courtesy the author)

(b) Block Jam (graphic reproduced from [NDNG03] by courtesy the author)

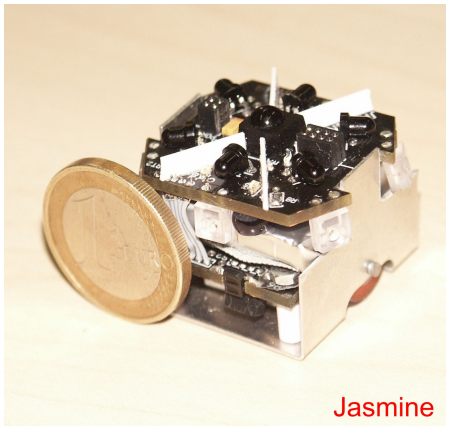


(c) Tangible Programming Bricks (graphic reproduced from [McN00] by courtesy the author)



(d) AudioCubes (graphic reproduced from [SV08] by courtesy the author)

(e) ActiveCube (graphic reproduced from [KIMK00] by courtesy the author)



(f) Swarmrobot "Jasmine" (graphic reproduced from [swa08])

Figure 2.2: Illustration of photos of systems that are the state of the art in active tangible objects and similar systems

approach is to make the interaction more user-friendly by using TUIOs, which adapt to the user's needs even more than Graphical User Interfaces (GUIs).
The objects consist of LEGO [leg08] bricks and contain a microcontroller and LEDs to provide simple visual output. They are specialized on different programming structures (program flow control, input or output, etc.). The Tangible Programming Bricks can be parameterized and extended by plugging in small smart cards that provide additional communication, sensing and motor control abilities.

AudioCubes

AudioCubes [SV08] provide a TUI for musicians to manipulate sound. The cubes can be configured to represent different sound production functionalities such as those of a microphone, a granular re-synthesizer, filters or sound output devices. The objects indicate the functionality they represent by their color (see Fig. 2.2(d)).

ActiveCube

With ActiveCubes [KIMK00] the user can construct 3D structures in virtual computer-generated environments by using physical objects (cubes). The system can detect the spatial configuration of the combined objects that are equipped with input channels such as gyros, ultrasonic sensors, luminous or temperature sensors. For future research, the cubes can possibly be equipped with output channels (e.g. displays or actuators to show simulated results). Possible applications of the system an be found in educational and/or entertainment scenarios such as games and toys that help to learn and improve skills in assembling (see Fig. 2.2(e)). Also artistic purposes are considered.

Small, Micro and Pico sized Robots and Swarms

There are plenty of small sized robotic projects. Swarmrobot [swa08] is an open-source micro-robotic project that claims to form the largest artificial swarm in the world (unfortunately without any further information). Figure 2.2(f) depicts a single robot unit, called 'Jasmine'. The Swarmrobot project aims to create swarm dynamics by combining a set of robots to create larger objects with new and emergent properties. There are other projects that focus on small sized robots, such as the One Inch Robot [Roy08] or Pico [pic08]. All these robots are not designed for the use as TUIOs, but they provide a great basis for TAO development. Problems that arise when active tangible objects are built have probably already been solved in this field. Thus experiences and research results from the related field (implementation of the mechanics and electronics) can be very valuable for the design of TAOs.

2.1 Summary

From the many approaches and applications in the field of active TUIOs only a selection of the most relevant systems was presented. PMD and Augmented Coliseum are approaches that represent changing virtual entities by using small moving robots. But neither of them mentions tangible interaction as an application field. The Actuated Workbench and the Siftabels enhance TUIOs with active feedback. A combination of these two approaches would result in objects similar to the TAOs that are developed in this work. In the presented systems Block Jam, Tangible Programming Bricks and Active Cubes are proposed as tangible programming environments. Block Jam has quite versatile in- and output channels (press-button, display and sound) compared to the Tangible Programming Bricks system, where only a few small LEDs and the corresponding program code are available. In their paper the authors of Active Cubes propose future extensions that utilize active feedback channels such as displays or actuators. Block Jam, Tangible Programming Bricks and Active Cubes follow a constructive approach, which is interesting for the field of tangible interaction. Audio Cube in contrast follows a relational approach. They offer visual feedback using color LEDs to represent the state of each cube. Finally small sizes robots such as the Swarmrobot Jasmine are technically interesting for this work. They are not intended to be used as a human-computer interface.

3 Background

This chapter explains the theoretical background of TUIs and TUIOs. Furthermore it presents the used components, tools and libraries that were used to create the proposed hard- and software.

3.1 Tangible User Interfaces

This section is based on the paper 'Tangible User Interfaces - Classification' [Loc06] which summarizes different classification approaches for TUIs. There are approacces by Ullmer and Ishii [UI00] which were extended by Hoven and Eggen [vdHE04].
Classical GUI systems can be characterized by the Model-View-Controller (MVC) design pattern [Joh87]. Transferred to TUIs, this model can be adapted to the Model-Control-Representation (physical and digital) (MCRpd) model. Figure 3.1 contrasts these two models.

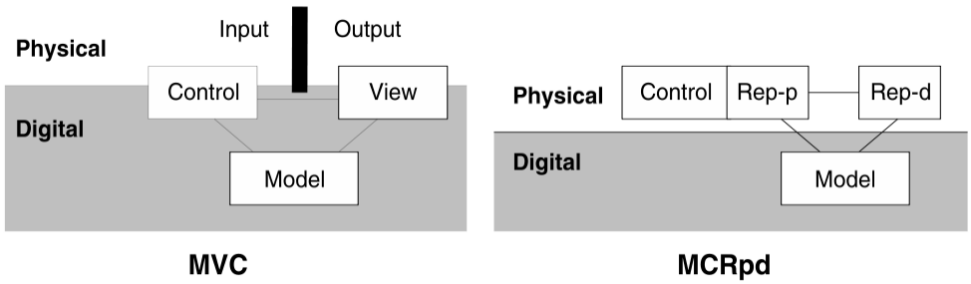


Figure 3.1: Model-View-Controller vs. Model-Control-Representation (physical / digital) (graphic reproduced from [Loc06])

While the control and the view 'live' both in the physical and the digital domain and input and output devices are separated in the MVC model, the corresponding parts, control and representation, are pulled into the physical domain as far as possible in the MCRpd model. The representation part (the former view) plays a special role, because there are two representations, one is physical and one is digital but both have effect in the physical domain, which is the reason why the digital representation is placed in the physical domain. The fact that only the

model remains completely digital and control and representation are completely physical is the main characterization for TUIs.
With this argumentation it is clear, that multi-touch systems are no TUIs because they lack the physical representation (there is only a digital one).
The MCRpd model brings up further TUI characterizations from the relations between the different parts of the model:

Physical Representation – Model: The coupling between the model and the physical representation is crucial asking the question: How do the TUIOs represent the digital data? Is it augmented digitally e.g. by projection?

Physical Representation – Control: This relation copes with control and the relation to the physical representation. Is the representation controlled by the user or by the system? What is the setting for the interaction?

Physical – Digital Representation: The relation between the physical and the digital representation focuses on whether there is a digital representation and how it augments the physical representation.

State: The last characteristic is not directly obvious from the model. It encompasses the embodiment of the digital state of the system by the physical state of the representation.

Furthermore Ullmer and Ishii [UI00] propose the following categories to classify TUIs:

Spatial: In the spatial category TUIs are gathered that take advantage of position and orientation of TUIOs.

Constructive: The constructive category contains TUIs where TUIOs get coupled mechanically together.

Relational: The relational approach establishes relations between TUIOs to put the represented data into a context (e.g. AmbiD [BHR06] and reacTable [JKGB05]).

Associative: This category includes TUIs, that simply associate data to TUIOs but without relations between them.

The above categories are not mutually exclusive. In addition, their borders are not strict, which means that systems may be classified into two or more of these groups. Depending on the purpose of creation of categories other criteria may be found and established.

3.2 Tangible User Interface Objects (TUIOs)

TUIs can use of one or more TUIOs, but nevertheless speaking about TUIs requires to define and establish characteristics for TUIOs, the most important part of a TUI from the users perspective.
The paper [FIB95] talks about Graspable User Interfaces which means the same as TUI, because the expression TUI was not established back in 1995. It introduces the concept of Bricks which is similar to the concept of TUIOs. This paper introduces a so-called design space to classify TUIOs. Table 3.1 gives an overview over this design space.

3.2.1 Taxonomy and Classification used in this work

TUIOs are physical objects which can be generally characterized by the following physical characteristics:

Shape: Physical objects can be round, rectangular, triangular, etc.

Size: Depending of the purpose of the TUI, the objects may be small or big. For example Globe4D [CvDHM06], manipulated with both hands, is with 30 cm diameter a relatively big TUIO whereas the TUIOs used in AudioDB [BEHR08] are relatively small with about 3 cm diameter. A few of them can be manipulated with a single hand.

Color and Texture: Physical objects can have different colors or textures. This also includes haptic textures such as 'smooth', 'rough' or 'bumpy'.

Degrees of Freedom: Rigid physical objects have 6 degrees of freedoms (DOFs). Depending on the system all or only a few of these are used for interaction. In addition to these 'outer' DOFs physical objects may have 'inner' DOFs, such as being bendable or squeezable, etc.

Beside these general characteristics of physical objects, there are additional TUIO specific properties:

Tracking: There are many ways of tracking the objects' position in the interaction area. For instance it is possible to track objects equipped with special markers visually. Another way would be tracking with electromagnetic fields.

Active vs. Passive: TUIOs can have manifold input and output modalities. If there are no active output modalities, that address the senses of the user, the objects are called passive otherwise they are called active.

Movement / Movability: Movement is a special case of output modality.

Constraints (virtual vs. physical): Desired constraints can be physical, but also virtual, if the used TUIOs are able to move themselves. Then it is possible to sustain certain positions or orientations. [PI07]

TUIO's internal ability	**Inert** (dump, only external shape)		Can exhibit **simple** expressions and has some internal logic (sensors, motors, indicator lights)		**Smart** (microprocessor, sensors, programmable)	
Input / Output	**Input-Properties sensed** Position, Orientation, Audio, Temperature, Tactile/Pressure, Light, Visual			**Output-Properties displayed** Position, Orientation, Audio, Tactile, Light, Visual		
Spacially aware	**Unaware**, works in isolation		**Mutual aware** (aware of each other)		**Aware of surroundings** (sensing of environment plus other TUIOs)	
Communication (inter-TUIO and to host)	**Wireless** (Infra-Red (IR), Bluetooth, etc.)		**Tethered** (cable)		**Grid board**	
Interaction time span	**Quick** gestures, fraction of seconds		**Intermediate cache**		**Long term** (days, months, years between interactions; archives)	
TUIOs in use at same time	1	2 - 5	5- 10	10 - 50	50 - 100	even more
Function assignment	**Permanent** (each TUIO is assigned to one function)		**Programmable** (functional roles can be reassigned)		**Transient** (rapid assignment; time multiplexed or space multiplexed)	
Interaction representations	All **physical** artefacts	Mix, but **physical** dominates	Balanced **mix** (Equal. Complementary or Combinatoric rep.)		Mix, but **virtual** dominates	All **virtual** artifacts
Physical and Virtual layers	**Direct** (layers superimposed)			**Indirect** (layers separated)		
Bond between Physical and Virtual layers	**Tightly coupled** (objects tracked continuously in real-time)			**Lousely coupled** (objects tracked and sensed in batch mode)		
Operating granularity	**Desktop** (fraction inch accuracy)		**Room** (inch accuracy)		**Building** (room accuracy)	
Operating surface type	**Static** (printed material, graphics, text does not change)			**Dynamic** (computer display)		
Operating surface texture	**Discrete** (plug-in positions on grid)			**Continuous** and smooth		

Table 3.1: Design space of TUIOs. [FIB95]

Container Object vs. Fixed Functionality: Some systems allow to assign different functionalities or data to the used TUIOs. The objects that act as containers and can be filled or emptied and reassigned with meaning by the user. [UIG98], [DHKR08].

Meaning and Functionality: [Dou04] points out that the user should be able to determine the meaning of the functionality of TUIOs. This means that the system should be designed as open as possible, to make the functionalities flexible to use. Perhaps the user might want to use this functionality in a way, the designer has not thought about. This should not be forbidden explicitly by the system design.

Iconic vs. Symbolic: 'The difference between iconic and symbolic signs is the fact that iconic signs share some representational properties with the objects they refer to. This is not the case for symbolic signs.' [Loc06] For instance the physical objects used in Urp [UI99], building models to represent buildings for city planning, are iconic where as AudioPad [PRI02] uses objects with a rather symbolic character, by only augmenting them through projection they become more iconic.

Reference Frame: The reference frame defines the physical interaction space. [Loc06] In the AcouMotion/Blindminton [HHR06] this is the whole room, many other systems such as the reacTable [JKGB05] use a table-top surface. Tangible artifacts such as audio shaker [HJBdC06] do not have an articulated or bordered reference frame.

Generic vs. Personal: The TUIOs can have different characteristics for the user. They can be general objects or personal with a special meaning for the user. [vdHE04], [Loc06].

Coupling: The coupling of objects describes the way of how the user utilizes the TUIOs [Dou04].

Single Object vs. Multi-Object System: Some tasks require only one object to be coped with, while others may require more. In [FIB95] the authors propose gestures for single or multi-object tasks.

The presented taxonomy allows to classify many systems. Some systems may not be easy to describe by these terms or require additional features to be described. So this list is not exhaustive.

3.3 Components and Tools used for Hardware Design

This section briefly introduces the used tools and hardware components that were used to build the hardware of the TAO system.

3.3.1 TUImod

TUImod [BKHR08] are modular building-blocks that can be combined in various ways to create TUIOs. The elements of TUImod were created by using a rapid-prototyping technique. Figure 3.2 shows an exploded assembly drawing.

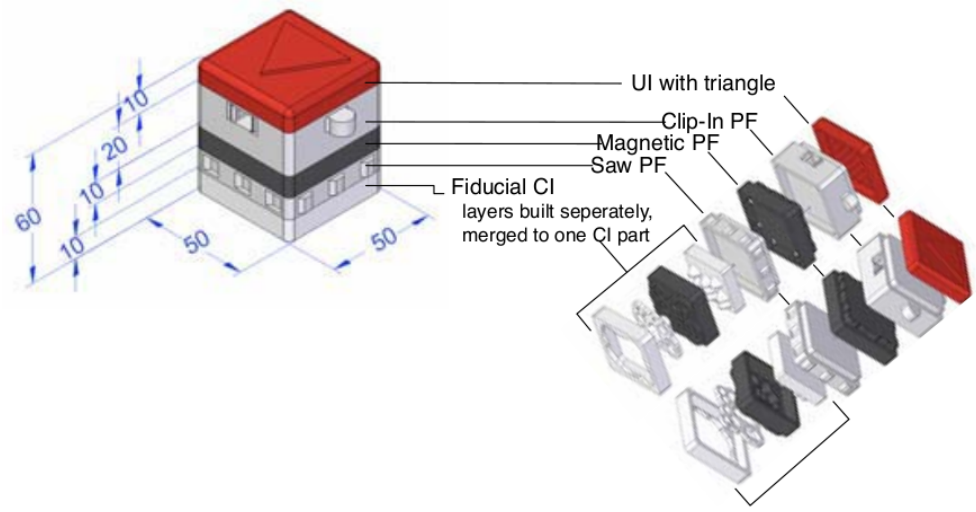


Figure 3.2: TUImod (graphic reproduced from [BKHR08] by courtesy the author)

TUImod provides three different layers of elements, User Interface (UI), Physical Functionality (PF) and Computer Interface (CI). The UI elements are attractive to the users perception. They differ in color, shape and haptics. The second layer is characterized by the PF. It offers magnetic attraction or repulsion, clip-in functionality or saw-tooth details fitting into each other. The last layer is the CI. It contains markers such as the fiducial markers [BKJ05].
The housings of the TAOs were based on the drawings of TUImod. This makes it possible to use the passive TUImod elements in combination with TAOs.

3.3.2 SolidEdge and Eagle

SolidEdge [sol08] and Eagle [eag08] are both Computer Aided Design (CAD) programs. SolidEdge allows digital product development. 3D models can be easily created and produced with a 3D rapid-prototyping printer. The housings of the objects based on the TUImod models were created using this CAD system.
Eagle is a specialized CAD program for designing and layouting of Printed Circuit Boards (PCBs). It assists the user in designing schematics and generating PCB layouts from these schematics. To route the wires between the parts, a powerful auto-router is included.

3.3.3 Arduino pro mini

The Arduino pro mini [ard08] is an open source rapid-prototyping board (see Fig. 3.3) with an ATmega168 microcontroller. It provides 14 digital input/output ports (of which 6 can be used as Pulse-Width Modulation (PWM)[1] outputs) and 6 analog inputs. The development environment contains the easy to use Arduino Programming Language, a variant of C with additional features, such as program flow control and convenience functions. These features are also known from Processing [pro08], a Java based rapid-prototyping language for quick and easy design of programs with graphic output. Additionally some very useful libraries are included which make this platform very versatile. The Arduino platform is heavily used in physical computing development [Igo07], because it allows to build prototypes in a short time. Compared to other microcontroller development systems, the Arduino system has a quite low learning curve.

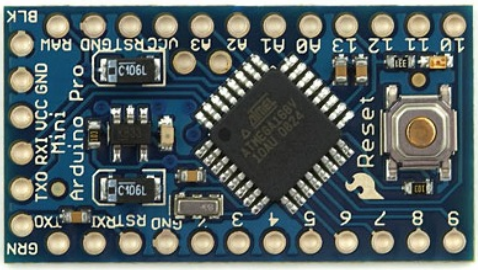


Figure 3.3: Arduino pro mini (graphic reproduced from [ard08])

3.3.4 XBee

It was very difficult to chose the right components for wireless communication between the PC and the TAOs microcontrollers. After experimenting with infrared communication (see Section 4.1.1), XBee modules [xbe08] became the components of choice.
Basically there are two versions of XBee modules (both in different flavors, e.g. with chip antennas or wire antennas or different ranges and pro versions). Both series implement the 802.15.4 stack, which is the basis for Zigbee [zig08]. XBee Series 1 only implements the basic functionalities and series 2.5 improves these by adding a full-featured mesh networking algorithm which also requires a different chipset to run. Table 3.2 sums up the differences between both versions.

[1] Pulse-Width Modulation (PWM) describes the modulation of the duty cycle of a power source, to control the amount of power sent to a load. The low-pass filtered and thereby integrated signal results in a higher amount of power at a higher modulation frequency and in a lower amount of power at a lower modulation frequency.

	Series 1	*Series 2.5*
Chipset	Freescale	Ember
Firmware	802.15.4	ZigBee
Network Topologies	Point-to-point, peer-to-peer, point-to-multipoint (star)	mesh
Indoor Range	30 m (90 m pro-version)	40 m
Outdoor Range	90 m (1.6 km pro-version)	120 m

Table 3.2: Comparison between XBee Series 1 and Series 2.5 [xbe08]

During experiments with both series it turned out that the mesh capabilities of series 2.5 slow down the communication. Sometimes a latency of about 10 seconds occurred, which makes this version insufficient for the operation of TAOs.

As table 3.2 shows, series 1 only allows point-to-point, peer-to-peer and point-to-multipoint network architectures. Series 2.5 allows a wide range communication where participants of the network act as relays to transmit over longer distances. So series 2.5 is great for wireless sensor networks where the nodes transmit data only every few minutes and real-time capabilities are not important. In contrast, series 1 is much faster but limited due to the range of the network, which is not important in this work. Because of the better real-time capabilities, finally XBee pro Series 1 has been chosen for this work.

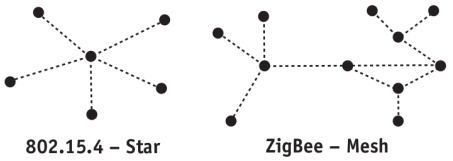


Figure 3.4: XBee network architectures (graphic reproduced from [xbe08])

3.4 Used Software Frameworks and Libraries

This section describes briefly the frameworks and libraries used in this work.

3.4.1 Image Component Library and Projects

The Image Component Library (ICL) [icl08] is an image processing library which is mainly based on the Intel Performance Primitives (IPP) [ipp08] library. The ICL adds some more libraries to be more versatile and provides the software designer with a very convenient API with easy access to image manipulation functionalities, GUI creation using Qt (see Section 3.4.2) and camera integration. So it is quite easy to rapidly design computer vision programs with this library. Additionally the ICL project ICLFiducialFinder wraps the Fiducial tracking library [rea08]. This is widely used in TUIs so it suggests itself to be used for the visual tracking of the TAOs.

3.4.2 The Qt Framework

Qt[2] is a very versatile C++ programming framework. It is intended to be used for the creation of GUI applications but beside the core (QtCore) and GUI components (QtGui modules and the QtOpenGL module) it offers way more capabilities encapsulated in other modules. The most important modules are

QtNetwork This module contains classes for a wide range of common network communication. Starting with classes for handling low-level socket connections (Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)), there are classes for File Transfer Protocol (FTP) and Hypertext Transfer Protocol (HTTP) transfers (including Secure Sockets Layer (SSL) support and cookies) which enable the programmer to easily implement servers and clients using these protocols.

QtSql The QtSql module offers easy access to databases that use Structured Query Language (SQL) as a query method. The open source version of Qt contains plug-ins/drivers for the following database access standards: MySql, Oracle Call Interface (OCI), Open Database Connectivity (ODBC), PostgreSQL, Sybase Adaptive Server, IBM DB2, SQLite and Borland Interbase.

QtScript QtScript offers an ECMAScript implementation [ecm08], the standard for JavaScript. Using this module enables the programmer to offer scripting possibilities for the user of the program. Small scripts easily extend programs without the need to recompile it.

QtXml and QtSvg The QtXml module offers support for the Simple API for XML (SAX) and the Document Object Model (DOM). Both methods allow the parsing and processing of Extensible Markup Language (XML) documents. As Scalable Vector Graphics (SVG) is and XML-based graphics format, the QtSvg module bases upon the QtSql module and offers the processing of SVG.

QtWebKit The QtWebKit module provides an interface for the very versatile browser engine WebKit [web08], used in Apple's browser Safari [kde08], Konqueror [kon08], the browser of the K Desktop Environment (KDE) project and many more. Using this powerful browser engine in Qt programs allows to easily create rich applications bridging between desktop and web content.

[2] Qt is available under two different licensing models. There is a commercial version and an open source version which is released under the GNU Lesser General Public License (LGPL) / GNU General Public License (GPL). Because the latter is used in this work, all descriptions relate to the open source version.

Phonon Qt encapsulates the Phonon framework, introduced by the KDE project [kde08]. It allows to include multimedia content, such as sound files and videos into Qt programs.

All these major features make the Qt framework a quite powerful toolkit for software creation, which make it the first choice for this work.

3.4.3 LibSerial

The communication between the microcontroller network and the controlling PC is done via a serial connection. Although on Unix systems serial ports are files that can be easily handled using pipes, handling the serial port is quite inelegant and confusing in C++. That is the reason why LibSerial [lib08] is introduced here. This library encapsulates the access to the serial port into convenient classes that allow simplified access to the serial port. Only few lines of code are needed to open the serial port with the right parameters and to send and receive messages.

3.4.4 Extended Communication Framework and XML Template I/O

The Extended Communication Framework (XCF) [xcf08] is an Internet Communication Engine (ICE) based communication framework that uses XML documents for encapsulating and describing all kinds of data. ICE is a slim implementation of the Common Object Request Broker Architecture (CORBA) base concepts aiming at easy usability. In XCF XML documents are used as a very flexible data structure. It implements a wide variety of communication paradigms:

- Client / server Remote Method Invocation (RMI) (many-to-one)
  - synchronous: client waits for reply of the server
  - asynchronous: client does not wait for a reply; server contacts the client when the result of the call is available
- Publisher-Subscriber paradigm (one-to-many)
- Naming service (dispatcher); cares for network transparency
- content-driven event-notification
  - white-board (ActiveMemory) that can be read and written by all participating processes
  - processes register on content using XPath
  - published content is automatically send to processes that subscribed on that content semantically
- XML Template I/O (XMLTIO)

Because SAX and DOM parsing of XML documents is relatively complex in terms of convenience, XMLTIO is crucial for software designers that use XCF in C++ programs. It encapsulates the XML parser Xerces-C++ [xer08] to allow easy and direct access to all elements of a XML document using XPath. Additionally it provides direct conversion of element data into native primitive data types.

3.4.5 The Cluster Library

The Cluster library as a part of the Cluster project [clu08] was originally developed as a computational environment for analyzing data from Deoxyribonucleic Acid (DNA) microarray experiments. Since it implements standard clustering algorithms such as K-Means and the EM-Algorithm or Self-organizing Maps (SOMs) and distance metrics, it can be used for all kinds of data. Along with Cluster there comes a C library that holds all these algorithms. It is utilized in this work to process datasets in the DataProcessor application (see Chapter 6).
In the DataProcessor the K-Means algorithms gets applied:

Result: partitioning of the given data into k clusters
Initialize

- *Cluster number* k
- *Initialize cluster centroids randomly.*

foreach *data point* **do**
  Assign data point to the most next cluster (center).
  Calculate new cluster centers depending on the assigned data points.
end

Algorithm 1: K-Means Algorithm

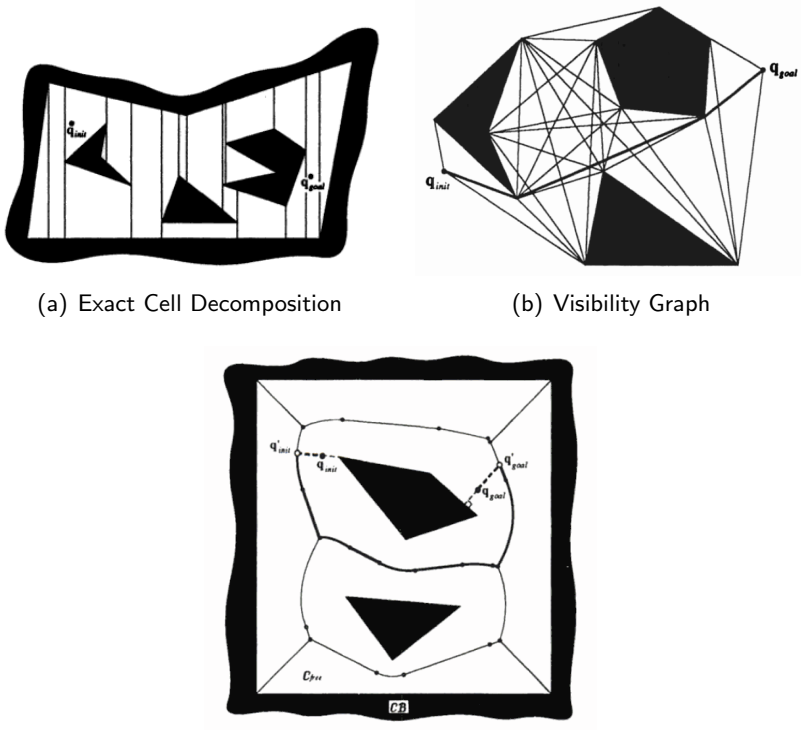
3.5 Path Planning Algorithms

In contrast to animals or human beings, navigating mobile robots through an environment equipped with obstacles is quite difficult. It gets even more difficult, if the obstacles move, too. Continuously the robot has to sense it's environment and plan the next step based on the sensed data. Thereby an as accurate as possible model of the environment needs to be estimated and updated every planning step in real-time. The problem gets much more complex if the robot has the ability to move in all six DOFs, which is the case with robot arms. This increases the computation cost immensely.
Since the movement of the TAOs is only planar this short survey of path planning algorithms only focuses on two-dimensional approaches. The free configuration space is called C_{free}. This space is allowed to be navigated. On the contrary the configuration space that is occupied by obstacles is called $C_{forbidden}$. It must be

avoided by the path planning algorithm to avoid collisions with obstacles. The junction of both spaces C represents a complete model of the environment.
There are different methods of determining these spaces. On the one hand there are discrete approaches, that try to sample the environment to get an approximately accurate model of C_{free} and $C_{forbidden}$. From this discrete grid of sampled points, graphs can be generated to navigate the robot through it's environment. On the other hand there are continuous approaches. One is the potential field method, that is used in this work. It does not sample the C_{free} directly, but the borders of $C_{forbidden}$ need to be known. Based on this borders and the desired targets the potential field approach finds the shortest way to the goal. One disadvantage is the fact that without further effort this approach can get stuck in local minima, but in the scenario of this work this case is improbable.

3.5.1 Roadmap Approaches

There are different types of roadmaps, that serve the problem of path planning. [Lat93] introduces four of them, but the number of ways of finding valid roadmaps is only limited by the creativity of the developer. Roadmaps can be regular grids that lie in the free configuration space C_{free} or non regular graphs.



(a) Exact Cell Decomposition

(b) Visibility Graph

(c) Voronoi Diagram

Figure 3.5: Different roadmap generation approaches. (graphic reproduced from [Lat93])

Cell Decomposition The Cell Decomposition methods generate non-overlapping patches, that cover C_{free} almost completely (see Fig. 3.5(a)). These patches can be used to create a roadmap. The decomposition can be exact or approximately. Approximately means that the patches are simpler, which reduces the computation cost. For instance creating trapezoidal patches along a curved obstacle needs quite a lot points to represent the curved shape. Approximating this curve by a rectangular shape is much more efficient, but less accurate. Both methods only work properly in two dimensions. In higher dimensions they only serve as a prove of concept while both get too complex to be calculated in real-time. After decomposing the C_{free} a graph can be created by connecting the centers of all adjacent patches with a line. This network of lines and nodes can be used to plan trajectories.

Visibility Graphs Visibility graphs (see Fig. 3.5(b)) can be created by connecting corners of all obstacles, visible from the current position and the target position. Additionally these corners are interconnected, but these lines may not cross $C_{forbidden}$. Again this method generates a roadmap that is suitable for path planning.

Voronoi Diagrams Voronoi Diagrams (see Fig. 3.5(c)) are created by drawing equidistant lines between the obstacles. This method is also called Delaughney-Tessellation.

Regular Grids Simple grids can be used to tessellate the free configuration space. Nodes that lie in the forbidden configuration space $C_{forbidden}$ need to be removed from the grid. Such grids are valid roadmaps, too.

3.5.2 Graph-based Path Planning Algorithms

Roadmaps are graphs. There are algorithms, that find the shortest connection between two nodes in a graph. Dijkstra and A* (both described by [Lat93]) are two of them, while Dynamic Wave Expansion Neural Network (DWENN) represents a combined approach using a regular grid.

Dijkstra The Dijkstra algorithms tries to find the shortest path through a given graph. The graph is weighted e.g. with an estimated distance. Like the pseudo code in Algorithm 2 describes, all nodes in the graph have the property 'distance' and 'parent'. All nodes but the initial node get the distance ∞, the initial point has the distance 0. For all unvisited adjacent nodes the one with the minimal distance is selected and saved as visited. For that new node the sum of distances (and arc weights) occurred on that path so far are saved in that node. The node with the shortest distance gets a pointer to it's parent node and so on. When arrived at the goal node, the pointers to the parent nodes represent the shortest trajectory through C_{free}. An optimal path has then been found.

Data:

- Nodes V
- Arcs $E \subseteq V \times V$
- Weights $W : E \to \mathbb{R}^*$

Result: Optimal path through the graph
Initialize

- $\forall v \in V g(v) = \infty$;
- $g(v_{start}) = 0$;
- $Q = \{v_{start}\}$;

while $Q \neq \emptyset$ *or* $v \neq$ *goal* **do**
  Choose $v \in Q$ with minimal distance $g(v)$:
    $v = \arg\min_{v \in Q} g(v)$;
  Remove v from Q : $Q' = Q \backslash \{v\}$;
  foreach *neighbors* u *of* v **do**
    if $g(v) + w(v, u) < g(u)$ **then**
      $g(u) \leftarrow g(v) + w(v, u)$;
      $bp(u) \leftarrow v$ (backpointer);
      Insert u or update $g(u)$ in Q and resort;
    end
  end
end

Algorithm 2: Dijkstra Algorithm

A* The A* algorithm searches the shortest path same as Dijkstra does. Using a heuristic, A* performs a deep search in contrast to Dijkstra, which does a broad search. To accelerate the search towards the goal, the heuristic additionally estimates the distance to the goal. Thereby A* finds the shortest path in optimal time. The pseudo code in Algorithm 3 describes the difference to Dijkstra.

DWENN DWENN [LSR03] does path planning on a regular grid, which is a special type of graph. Every node is interpreted as a neuron:

$$a_i(t+1) = \sum_j w_{ij}(t) \cdot a_j(t) + \underbrace{\theta_i}_{(=2)} \tag{3.1}$$

$a_i(t)$ is the neuron's activity, while w_{ij} represents the connection strength between neuron j $\to$ i. The idea of DWENN is to propagate a wave from targets a_* to the current position in C_{free}. Targets have the lowest activity (1). The gradient towards the lowest activity draws the path to the target.

Data:

- Nodes V
- Arcs $E \subseteq V \times V$
- Weights $W : E \to \mathbb{R}^*$

Result: Optimal path through the graph
Initialize

- $\forall v \in V g(v) = \infty$;
- *estimate* $h(v)$;
- $g(v_{start}) = 0$;
- $Q = \{v_{start}\}$;

while $Q \neq \emptyset$ *or* $v \neq$ *goal* **do**
  Choose $v \in Q$ with minimal distance $g(v)$:
    $v = \arg\min_{v \in Q} g(v) + h(v)$;
    `/* now like Dijkstra */`
  Remove v from Q : $Q' = Q \backslash \{v\}$;
  foreach *neighbors* u *of* v **do**
    if $g(v) + w(v, u) < g(u)$ **then**
      $g(u) \leftarrow g(v) + w(v, u)$;
      $bp(u) \leftarrow v$ (backpointer);
      Insert u or update $g(u)$ in Q and resort;
    end
  end
end

Algorithm 3: A* Algorithm

DWENN uses the following update rules.

Target

$$a_i(t+1) = 1 \tag{3.2}$$

Target-Neighbor

$$a_i(t+1) = \begin{cases} 2 & \text{Target has move since last step} \\ a_i(t)+1 & \text{Target has not moved} \end{cases} \tag{3.3}$$

All other nodes If $a_i(t-1) \neq 0 \wedge a_i(t) = 0 \Rightarrow a_i(t+1) = 0$
→ if node got deactivated, it stays deactivated another time step.
Else

$$\begin{aligned} a_i(t+1) &= \sum_{j \in N_i} w_{ij}(t) \cdot (a_j(t)+2) \\ &= \begin{cases} a_k(t)+2, & \text{if k first best neighbor} \\ 0, & \text{no good neighbor} \end{cases} \end{aligned} \tag{3.4}$$

Maximal one weight $w_{ij}(t)$ is set to 1. This weight describes the gradient in the grid ($\hat{=}$ backpointer).

Weights

$$w_{ij}(t) = \delta_{jk} \tag{3.5}$$

where k is the first "best" neighbor of node i.

3.5.3 Continuous Approaches

As already mentioned, there are other approaches besides the graph-based path-planning algorithms.

Potential Field Methods

In [Lat93] a complete potential field approach is described. Originally in [Kha86] it has been proposed for online navigation where the robot does not have a complete global model of the environment. Instead the robot is interpreted as a particle in the influence of different force fields. The goal attracts the robots while the (sensed) obstacles generate a force field, that repulses the robot. If the robot does not get stuck in a local minimum of the potential field, this approach leads it to it's goal. [Lat93] proposes the following definitions for the different potentials and their derived forces, which give the direction of the trajectory.
The attractive potential is defined as a conic well, centered at the goal position.

$$E_{att}(\vec{x}) = \xi \|\vec{x} - \vec{x}_t\| \tag{3.6}$$

Here ξ is a positive scaling factor. As the repulsive potential the following definition is proposed. For better readability $C_{forbidden}$ is now denoted as C_b in the following equations.

$$E_{rep}(\vec{x}) = \begin{cases} \frac{1}{2}\eta \left(\frac{1}{\|\vec{x}-\vec{C}_b\|} - \frac{1}{\rho_0}\right)^2 & \text{if } \|\vec{x} - \vec{C}_b\| \leq \rho_0 \\ 0 & \text{if } \|\vec{x} - \vec{C}_b\| > \rho_0 \end{cases} \tag{3.7}$$

The combination of both potentials result in the final potential field.

$$E = E_{att} + E_{rep} \tag{3.8}$$

To get an appropriate direction, the potential field needs to be derived. The derived scalar potential at a certain position in the field results in a vector pointing towards the direction that is a linear combination of all forces at that point. So the robot at that position is pulled in the direction of this vector.

$$\nabla \vec{E}_{att}(\vec{x}) = -\xi \left((\vec{x} - \vec{x}_t) \cdot \frac{1}{(\vec{x} \cdot \vec{x}_t)} \right) \tag{3.9}$$

According to the attractive forces, the repulsive forces are defined by deriving the repulsive potential.

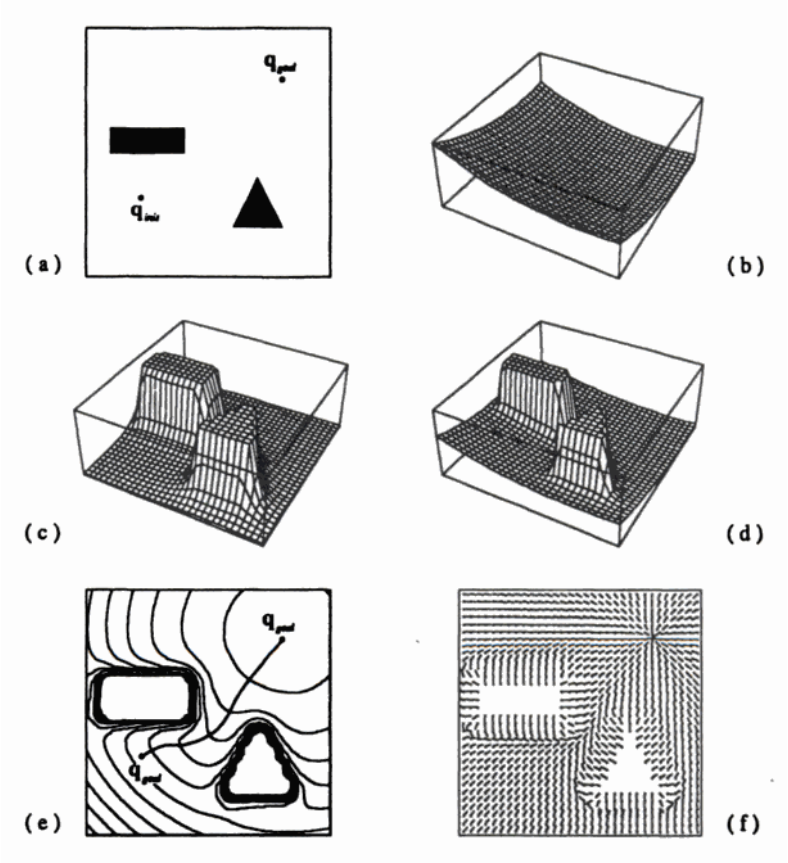


Figure 3.6: 'This shows an attractive potential field (Figure b), a repulsive potential field (Figure c) and the sum of the two (Figure d) in a two-dimensional configuration space containing two C-obstacles (Figure a). Figure e displays both several equipotential contours of the total potential and a path generated by following the negated gradient of this function. Figure f shows a matrix of the negated gradient vector orientations over free space.' (graphic reproduced from [Lat93])

$$\nabla \vec{E}_{rep}(\vec{x}) = \begin{cases} \eta \left(\frac{1}{\|\vec{x}-\vec{C}_b\|} - \frac{1}{\rho_0} \right) \cdot \frac{1}{\|\vec{x}-\vec{C}_b\|^2} \cdot (\vec{x} - \vec{C}_b) & \text{if } \|\vec{x} - \vec{C}_b\| \leq \rho_0 \\ 0 & \text{if } \|\vec{x} - \vec{C}_b\| > \rho_0 \end{cases} \tag{3.10}$$

Alternative A different way (not proposed by [Lat93]) of calculating appropriate potential fields is the use of *harmonic potentials*, which is used in this work. Here the attractive potential is defined as following.

$$E_{att}(\vec{x}_j) = \lambda \|\vec{x}_j - \vec{x}_t\|^2 \tag{3.11}$$

The repulsive forces are defined using a Gaussian, centered at the obstacles' positions:

$$E_{rep}(\vec{x}_j) = \mu \sum_{i \neq j} \exp\left(-\frac{1}{2\sigma^2}(\vec{x}_i - \vec{x}_j)^2\right) \tag{3.12}$$

The resulting forces of the combination of both potential fields are easily computed by computing the gradient:

$$\begin{aligned} \Delta\vec{x}_j &= \nabla E(\vec{x}_j) \\ &= 2\lambda(\vec{x}_j - \vec{x}_t) - \mu \sum_{i \neq j} \frac{(\vec{x}_j - \vec{x}_i)}{\sigma^2} \exp\left(-\frac{1}{2\sigma^2}(\vec{x}_i - \vec{x}_j)^2\right) \end{aligned} \tag{3.13}$$

E*

While DWENN produces rectangular wavefronts, E* [Phi04] is more inspired by real waves, which are circular, so are the wavefronts generated by E*. This results in very smooth trajectories. Additionally the library implementing the E* algorithm, provides a very versatile framework for creating path planning pieces of software including motor driving classes, obstacle detection using laser scanner data and so on. It is very promising but a bit too much for the needs of this work at this stage.

3.5.4 Summary

In the scenario proposed in this work the potential field approach is exactly what is required. Configurations containing local minimums are relatively rare, but can become probable with an increasing number of TAOs. It also inherently assures a minimal spacing between the moving robots. Roadmaps need to be validated. Even if the found roads are valid, the ways may be quite near to the forbidden configuration space. There is the danger that the robot gets too near to obstacles and collides with them. Such awkward roadmaps are quite common, as first experiments with approximate cell decomposition showed (see Section 5.2.6 for details). So the potential field approach using harmonic potentials is the method of choice for this work.

3.6 Robotic Mobile Platform Designs

For locomotion of Wheeled Mobile Robots (WMRs) in planar terrain, there are mainly four different types of mobile platforms that vary in mechanical design.

Differential Drive The differential drive mainly consists of two opposing driving wheels (and one or two free wheels, so called Castor wheels). With only these two wheels it is possible, to move the robot forward and backward

(both wheels drive in the same direction) or rotate in place (wheels drive in opposing directions). It is even possible to drive curves, but it is impossible to rotate during linear translation.

Tricycle Drive The tricycle drive uses three wheels, two rear wheels and one front wheel. The front wheel is used for steering and driving. The control patterns are more difficult than those of the differential drive. It is not possible to rotate in place, due to the non-holonomic constraints of the wheels. A couple of maneuvers (and space) are necessary to rotate the robot.

Synchro Drive A synchro drive allows each wheel to be steered and driven. The wheels need to be aligned into the same direction and powered with the same speed (synchronously). This fact gives the drive it's name. It allows the robot to rotate in place and drive curves, but the alignment of the wheels consumes time. Rotation during driving is impossible.

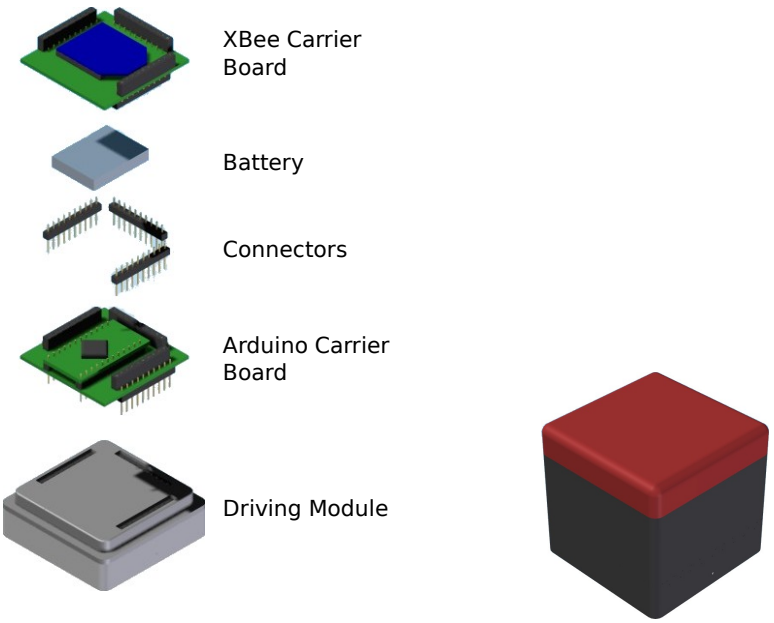
Holonomic or Omni Drive The omni drive is the most versatile drive. It uses wheels with inner DOFs (so-called Swedish Wheels). Those three wheels are arranged in a triangle around the robot. Simply driving two of the three wheels makes the robot move forward or backward perpendicularly on the not driving wheel. It also allows to rotate in place and to rotate while driving. The only disadvantage is that relatively much energy is lost at the wheels and the wheels are mechanically quite complex, which makes the holonimic drive hard to be implemented in the size of a TAO.

Ackerman Drive The Ackerman Drive is well known from conventional cars. It utilizes four wheels from which the two front wheels are used for steering and driving. Like the tricycle drive the Ackerman drive needs relatively complex maneuvers and space. It is not possible to rotate in place. But it is very practical for driving curves.

The best design would be the holonomic drive. It is able to use all available DOFs at the same time. But it is quite expensive and complex to develop such a drive in this small size. For these reasons the cheapest and easiest drive, the differential drive is chosen as the mechanical base design in this work.

4 Hardware Design

This chapter decribes the development of the TAO hardware in detail. In the first stage, the wireless serial transmitter is presented, followed by the manufacturing of a first TAO prototype (see Fig. 4.1). After proving the prototype to work properly, a small range of 'series' TAO models are created. The chapter ends with the description of two serial protocols, implemented in the Arduinos' firmware.



(a) Exploded Assembly Drawing without upper housing

(b) Drawing of a completed TAO

Figure 4.1: Assembly Drafts of a TAO

The housing of the prototype has been created using SolidEdge [sol08] and the TUImod [BKHR08] as framework for TUIO building blocks. It was then manufactured using a 3D rapid-prototyping printer. The object's electronics will be build using the Arduino mini microcontroller board [ard08] and Eagle [eag08]. This provides a very flexible programmable an reconfigurable platform for all forms of active objects.

4.1 Wireless Serial Transmitter

The first attempt to transmit wirelessly commands used pulsed infrared light because of the cheapness of the used components. Unfortunately it turned out that sometimes the serial connection was faulty due to the absence of debugging possibilities we regarded it as insufficient for this project. Nevertheless it is described here for completeness.

4.1.1 IR Transmitter board

The transmitter board relays the serial commands to the active objects via infrared. In [Igo07] different ways of wireless communication are introduced. Infrared communication seemed to be the cheapest and easiest way to make the TAOs wireless. So far only a unidirectional connection is needed. In future developments bi-directional communication can be desirable, for instance to allow the implementation of input channels in the TAOs.

Used Parts

1. *Universal Serial Bus (USB) Universal Asynchronous Receiver Transmitter (UART) Adapter*
   The FT232R is a popular USB to serial Transistor Transistor Logic (TTL) converter Integrated Circuits (ICs) (UART). Sparkfun Electronics [spa08] offers breakout boards for convenience usage. These boards allow to easily access microcontoller ICs over USB connections.

2. *TX-IRHS IC*
   The website of Reynolds Electronics [tx-08], manufacturers of the TX-IRHS says:

   > 'The TX-IRHS is an 8-pin PIC microcontroller custom programmed as a High-Speed infrared carrier modulator/transmitter IC. The TX-IRHS is designed for use in High-Speed [up to 19200 baud] infrared serial data links, and remote control applications.'

3. *20 MHz ceramic resonator*
   The TX-IRHS needs a clock generator. It needs a 20 MHz ceramic resonator, also offered by Reynolds Electronics.

4. *IR diodes*
   To emit the modulated signal over infrared, IR diodes are used (TSHF5410 by Vishay Semiconductors).

5. 220Ω *Resistor*
   The IR diodes need a series resistor. A simple 220Ω resistor is used here.

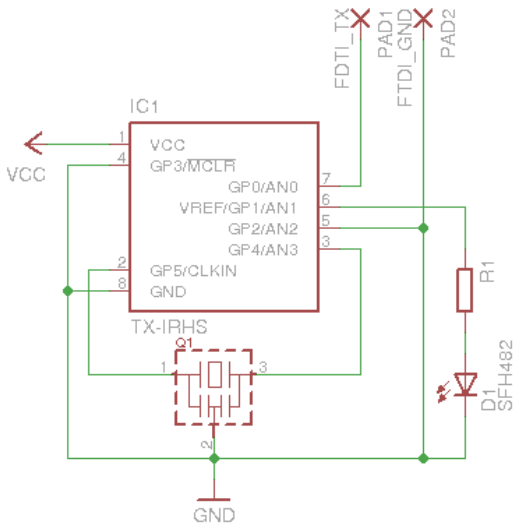


Figure 4.2: Scheme of the IR-Board

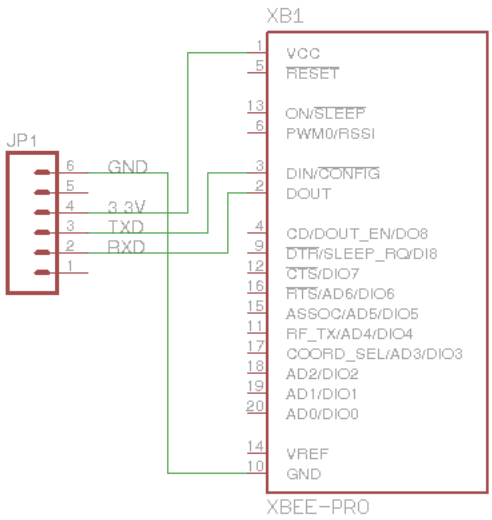
Assembly Description

The transceiver circuit, build up on a breadboard, uses the TxD (Transmitted Data) signal and the common ground pin of the FTDI USB interface. This serial TTL signal gets modulated with a carrier frequency of 455 kHz by a TX-IRHS (IC 1), a custom programmed Programmable Integrated Circuit (Microcontroller) (PIC). This modulated signal can be send using IR emitting diodes and received and demodulated by the TSOP7000, build into a prototype TAO.
The TAO then receives the serial data stream as if it would be wired directly to the computer (without back-channel). Every object has its unique ID and every command transmitted starts with the ID of the desired object. So every object can be controlled individually.

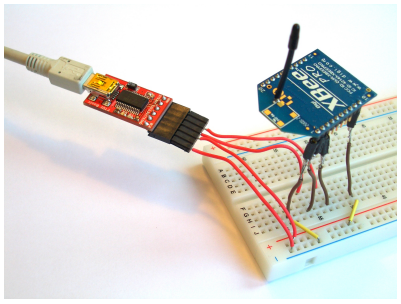
4.1.2 XBee Transmitter Board

Because the IR connection was not stable, experiments with XBee modules were made. It turned out, that XBee modules create a quite robust connection, provide a back channel for feedback and build a one-to-many wireless network. This is an almost optimal connection, that suits the needs of this work. The schematic is quite easy (see Fig. 4.3(a)). Only power and data pins are connected and the modules are configured with the following profile, generated by the X-CTU software for XBee configuration:

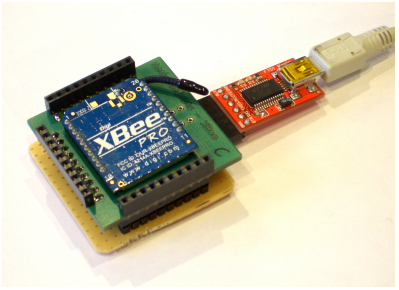
```
1 XBP24_15_4_1084.mxi          4 241
2 FE                           5 1084
3 0                            6 0
```



(a) Schematic



(b) Prototype



(c) Handmade board with manufactured XBee Carrier Board

Figure 4.3: XBee Transmitter

```
7  [A]CH=C
8  [A]ID=3332
9  [A]DH=0
10 [A]DL=0
11 [A]MY=0
12 [A]RN=0
13 [A]MM=0
14 [A]CE=0
15 [A]SC=1FFE
16 [A]SD=4
17 [A]A1=0
18 [A]A2=0
19 [A]PL=4
20 [A]CA=2C
21 [A]SM=0
22 [A]ST=1388
23 [A]SP=0
24 [A]DP=3E8
25 [A]BD=3
26 [A]RO=3
27 [A]D7=1
28 [A]D6=0
29 [A]D5=1
30 [A]P0=1
31 [A]AP=0
32 [A]PR=FF
33 [A]RP=28
34 [A]CT=64
35 [A]GT=3E8
36 [A]CC=2B
```

This allows peer-to-peer communication between all XBee modules, that joined the network. When two or more modules transmit at the same time, only the data of one of the modules is received. This fact makes the evaluation of multiple participant modules tricky or even impossible. Because the applications developed in this work do not need a back-channel, this problem is ignored. It can be coped with an advanced protocol that assures that only one module transmits at the same time.
In the prototyping phase, the XBee module was connected using a breadboard. This was quite tricky, because the XBee modules are manufactured in 2 mm hole spacing. The breadboard is designed to carry parts in 2.54 mm hole spacing. Figure 4.3(b) shows how the XBee was mounted. 4.3(c) depicts the transmitter using an adapter board for the manufactured XBee carrier boards. The following section lists the used parts for the XBee transmitter using the manufactured XBee carrier board.

Used Parts for the Series Transmitter

1. *USB UART Adapter*
   FT232R ICs are popular USB to serial TTL converter ICs (UARTs). Sparkfun Electronics [spa08] offers breakout boards for convenience usage. These boards allow to easily access microcontoller ICs over USB connections.

2. *XBee module*
   For wireless communication XBee pro Series 1 modules are used, as described in 3.3.4.

3. *XBee Carrier PCB*
   As described in Section 4.2.2

4. *Prototyping PCB*
   The 40 x 40 mm prototyping board carries all connectors to act as an adapter between the XBee carrier board and the FTDI adapter.

5. *angled six-pin male Connector*
   An angled six-pin connector is used to connect the transmitter board to the FTDI adapter.

6. *three 10-pin female Connectors*
   The three 10-pin female connectors are used to mount the XBee carrier board on the prototyping PCB.

4.2 Prototype

After determining the parts needed to build the first prototype, the design of the housing was the first step. Using TUImod (see section 3.3.1), a well defined frame

was set. So the housing has a base area of 50 x 50 mm. To make the housing as stable as possible, because it will be manually used as everyday objects, the side thickness was set to 5 mm. So all electronics were chosen to fit on PCBs of 40 x 40 mm size. The prototype PCBs (see section 4.2.2) were soldered using drilled prototyping boards silver wire and isolated wire and stacked in a sandwich architecture, to stay inside the border of 40 x 40 mm. When the PCB were finished, the housing height could be determined. They are designed as two halves (top and bottom), that clip together when pressed into each other. The module boards fit into these halves and only expose their connection pins, that still fit together, which results in a modular layer structure, which is highly reconfigurable. This structure is comparable to the shield concept, known from the more common, bigger sized Arduino boards, such as the Diecimila or Duemilanove.

4.2.1 Used Parts

For the prototype the following parts and components were used, which are depicted in Figure 4.4(a) and Figure 4.4(b):

1. *Housings and Wheels*
   Housings, based on the TUImod framework (see section 3.3.1), and wheels were designed using SoligEdge (see section 3.3.2). The parts were created using a rapid prototyping 3D printing system. For better debugging, only the housing of the motor layer was used.

2. *Arduino pro mini*
   As introduced in section 3.3.3, this microcontroller platform is used as the 'brain' of the active objects.

3. *Motors*
   The motors are 6 mm inline gear motors by Precision Microdrives [pre08]. They are distributed by ROBOTmaker [rob08].

4. *Batteries: Fullriver 130*
   Distributed by WES-Technik [wes08], the batteries are Fullriver 130 mAh lithium polymer accumulators.

5. *H-bridge L293D*
   As H-bridge, a L293D IC, distributed by Reichelt Elektronik [rei08], is used.

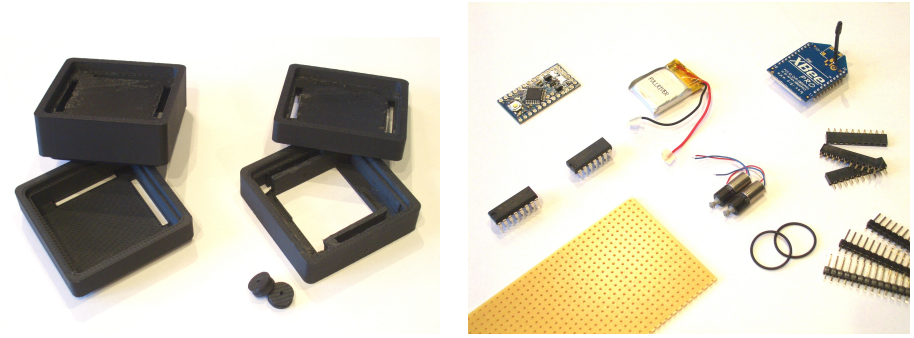
6. *Inverter: 74AC04*
   The 74AC04 IC is distributed by [rei08]. It is used for inversion of the direction pins, which allows to save two digital output ports of the Arduino.

7. *XBee pro Series 1*
   A Xbee module as introduced in section 3.3.4 is used for wireless communication.

8. *Connectors*
   A set of different connectors (male and female) are used to create the sandwich architecture of the PCBs. Like the L293D and the 74AC04, they are distributed by [rei08].

9. *Drilled Prototyping Board*
   The drilled prototyping boards are also distributed by [rei08] and by [far08].

10. *Rubber Belts*
    For driving the wheels, two 15 mm x 1 mm, shore 70 rubber belts[1], distributed by [rob08] were used.



(a) Printed housing parts and wheels

(b) Used electronic parts. Arduino mini pro, Batteries, XBee module, L293D, 74AC04, drilled prototyping board, motors, rubber belts and connectors

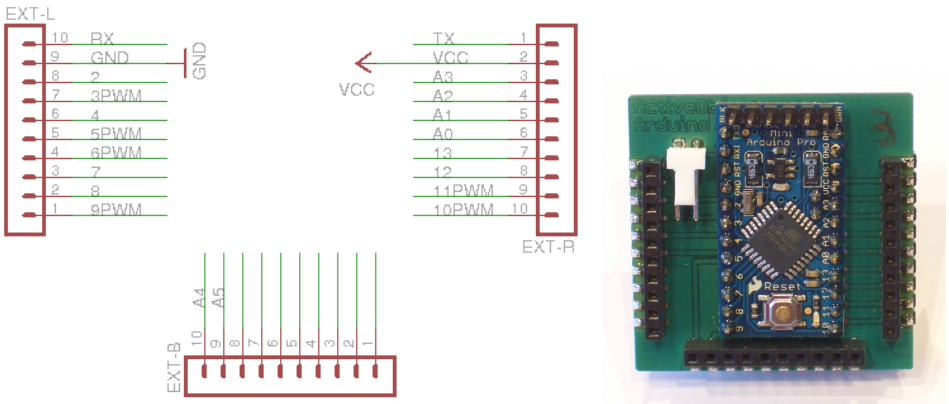
Figure 4.4: Parts, used for the prototype

4.2.2 Schematics and Board Design

The sandwich architecture uses a vertical bus with 30 pins. These pins are subparted into three 10-pin buses, that lie on three of the four sides of the PCBs (see Fig. 4.5). This arrangement avoids connecting modules in wrong ways. EXT_L is the bus on the left side, EXT_R resides on the right side and EXT_B is found at the bottom side. EXT_B has still unused pins (pins 1 to 8), which are reserved for future extensions.

Arduino Carrier Board On the Arduino carrier board (see Fig. 4.6) the Arduino mini pro is soldered. It connects all pins of the Arduino to the bus system. Pins D2 to D9, GND and RX on the left of the Arduino board are connected to the left bus EXT_L and pins D10 to D13, A0 to A3, VCC and TX are connected to the right bus EXT_R. To the bottom bus EXT_B only A4 and

[1] shore is a durometer scale, describing the hardness of a certain material



(a) Pin Assignment Schematic

(b) Manufactured PCB for Comparison

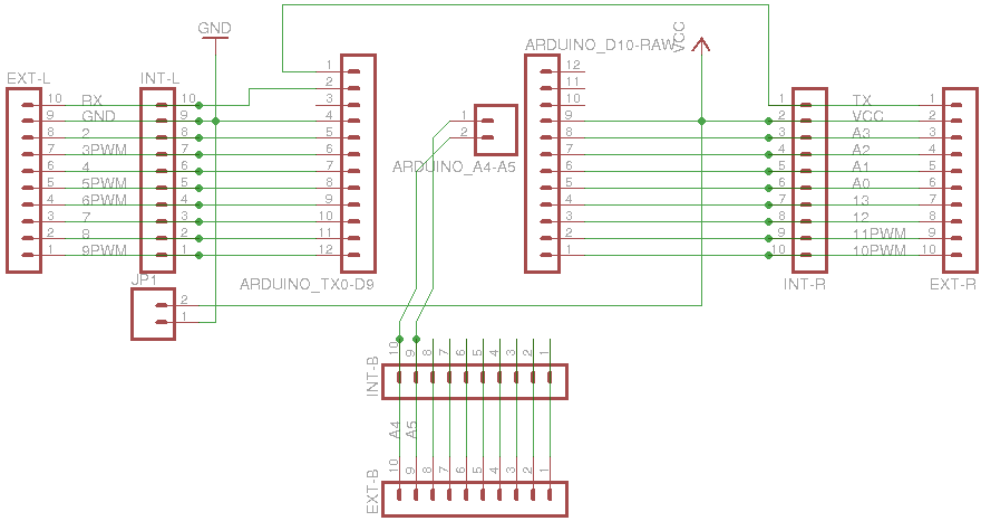
Figure 4.5: Pin Assignment Overview. Pins 1 to 8 of the EXT_B connector are reserved for future extensions.

A5 are supposed to be connected, which was not done, because they are so far not needed. The remaining pis of EXT_B are so far unassigned and reserved for future extensions. All connections to the buses are made using silver wire. Male and female connectors are added to the prototype PCB as Figure 4.6 shows. Finally a six-pin connector is soldered to the Arduino mini pro to make it connectible to the USB UART board.

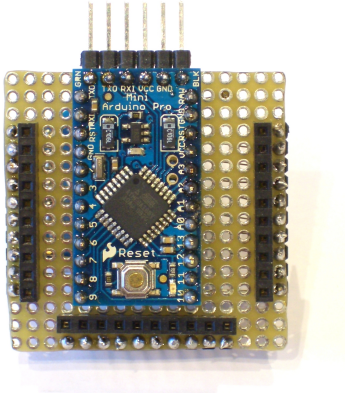
XBee Carrier Board Figure 4.7 shows the schematic and the prototyping PCB carrying the XBee module. As already mentioned in section 4.1.2, the XBee does not directly fit on prototyping PCBs because of its 2 mm pin spacing. So the prototype board only provides sockets to mount the XBee module like it is mounted on a breadboard. The status Light Emitting Diodes (LEDs) mentioned in the schematic are ignored to keep the design simple.

Motor Driver Board The motor driver board, depicted in Figure 4.8, is crucial for this work. It implements the movability capabilities and is responsible for the fist active output modality. The motor driver consists of two ICs, A L293D H-bridge and a 74AC04 inverter (for schematics see Fig. 4.9).

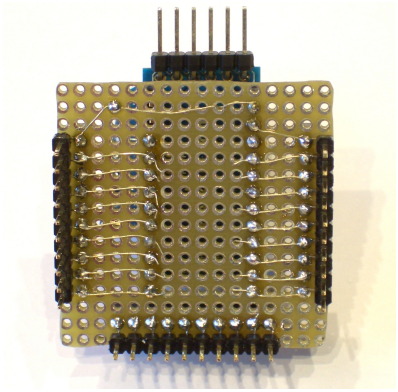
The H-bridge allows to control a motor by using three ports of a microcontroller. One (PWM) port is used for the velocity and the other two ports are digital ports to determine the direction (one is set to high and the other to low or vice versa). If both digital ports are set high, the H-bridge lets the motor block so this can be used as a brake. To save one of the digital ports an inverter IC such as the 74AC04 is used. By giving up the ability to use the motor for breaking by connecting one port directly to the first direction pin and connecting the second pin using an inverter such as the 74AC07,



(a) Schematic



(b) Prototype (top)



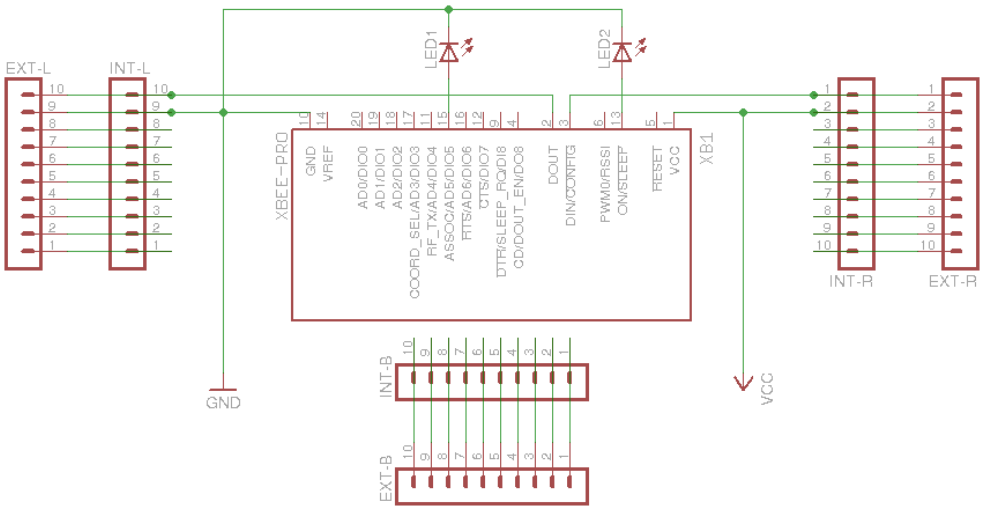
(c) Prototype (bottom)

Figure 4.6: Arduino Carrier Board

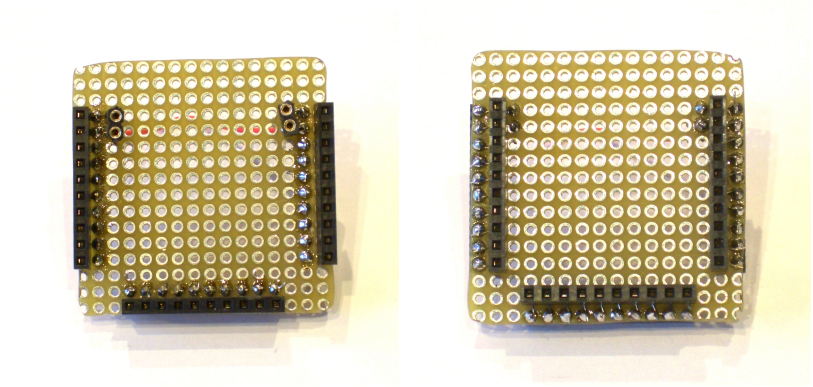
a second digital port can be saved. The direction can be determined using only one port, by setting the digital port high or low. The motor driver board uses pins D2 to D5 and GND and VCC. D2 is the direction port and D3 determines the speed for the first motor, D4 and D5 do the same for the second motor. Figure 4.10 depicts the ready build prototype.

4.2.3 Fiducial Marker

In the Ambient Intelligence Group [ami08] fiducial markers [BKJ05] are common for visually tracking objects on table-top surfaces. Due to the complexity of the popular 'amoeba' set [BKJ05], a new much simpler set was created for the



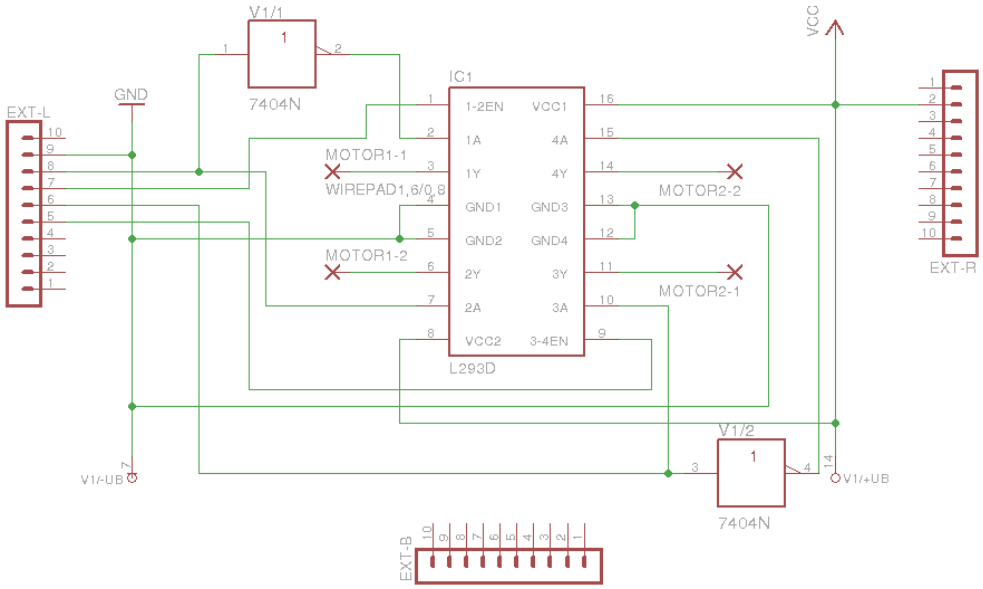
(a) Schematic



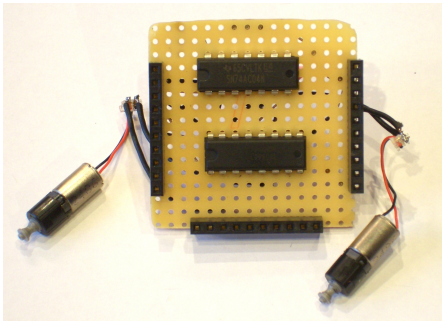
(b) Prototype (top) (c) Prototype (bottom)

Figure 4.7: XBee Carrier Board

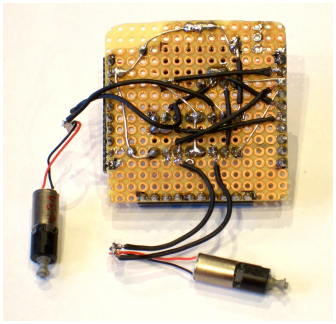
TAOs, because the space underneath them is limited and thereby too small for the markers of the amoeba set.
The fiducial tracking library is designed to recognize the graph-like structure of black-/white-changes. Figure 4.11 explains the design. The graphs are so-called left heavy depth sequences. After building up the structure (from root to leafs) the sequence of hierarchical elements can be read starting from root to the first left leaf to the next right leaf and so on. After the last leaf of the first node reading continues with the next upper right node and its leafs. The sequence is complete when the last (most right) leaf has been read.
As already said, the amoeba marker set is too complex to be printed and robustly recognized under the TAOs. The space under the TAOs is limited and the black-/white-changes of the amoeba markers would be in sub-pixel dimensions, which



(a) Schematic



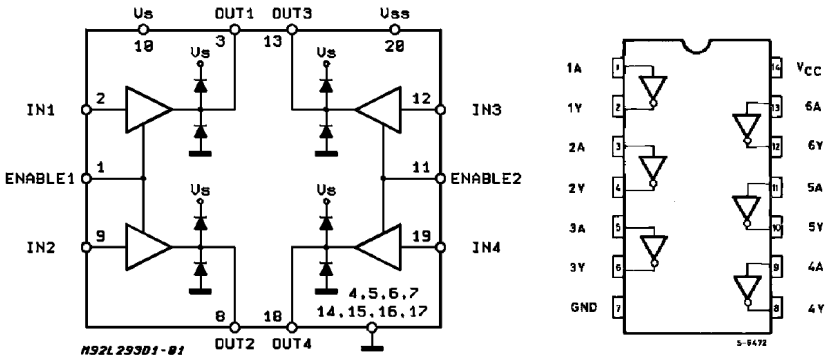
(b) Prototype (top)



(c) Prototype (bottom)

Figure 4.8: Motor Module Board

makes them unable to be tracked. Therefore a new marker set was created in this work. The first four markers of the new simpler marker set, called 'alien faces', are depicted in Figure 4.12(a).
The elliptic shape of the first version of the marker set was changed to a rectangular one (see Fig. 4.12(b)), because the markers are supposed to be included into the housing. The new marker set makes better use of the space underneath the TAOs, which improves the tracking stability, too.



(a) Prototype (top) (b) Prototype (bottom)

Figure 4.9: Schematics of the driver ICs used in the motor driver board (courtesy of SGS-Thomson Microelectronics).

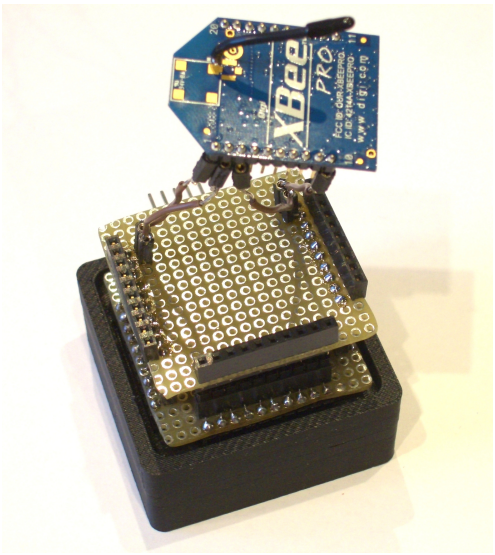


Figure 4.10: Photo of the functional Prototype, without upper housing.

4.3 Series Production

This section describes the creation of a set of active objects. Some changes had to be made to the prototype to enable a faster creation and more reliable results. These changes are described here.

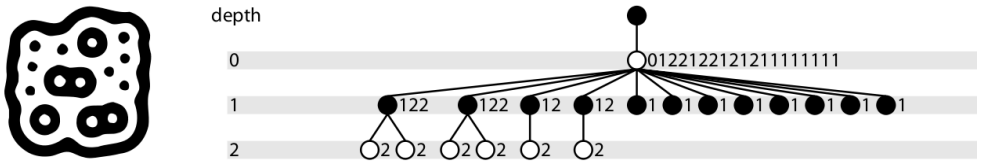
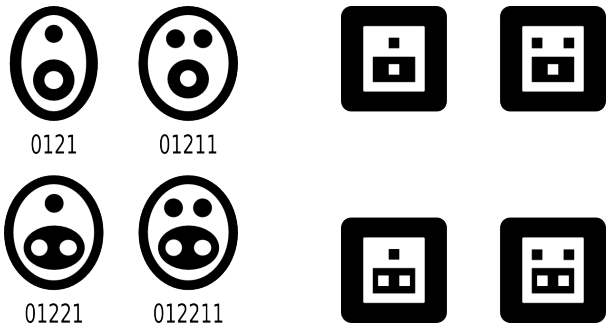


Figure 4.11: An amoeba fiducial marker and its left heavy depth sequence of black-/white-changes. (graphic reproduced from [BKJ05]). The sub-sequence of the nodes are printed right to each node. The unique structure of the 'amoeba' and 'alien faces' marker sets allow to organize the tree structure in a sequence (beside the zero-depth node).



(a) First elliptic version (b) rectangular version

Figure 4.12: Alien faces fiducial markers for visual tracking of the active objects. The number sequence below the markers are the left heavy depth sequences of their corresponding tree graphs, see Fig. 4.12(a).

Because of the mistake of designing the housings of the prototype mixing male and female housing connections, the housing connections between the prototype's modules of the TAOs and the passive TUImod parts are partly incompatible. This required a redesign of all housing parts. Figure 4.3 shows the new housings.

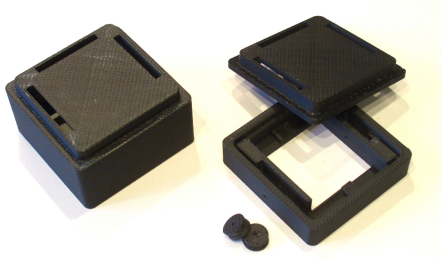
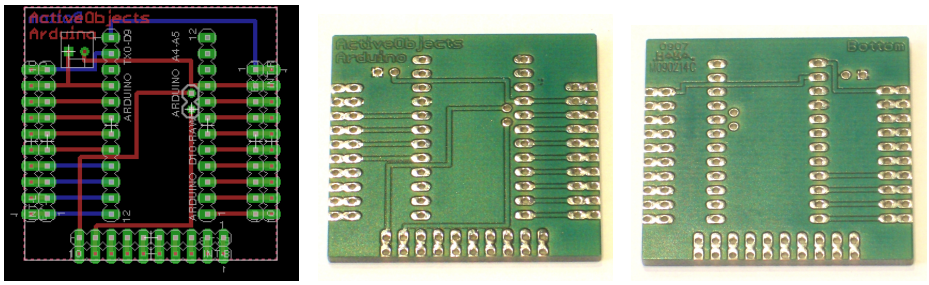


Figure 4.13: New version of the housing parts

While the female connectors round up with the lower surface of the connection, the male connectors of the connecting module need to stick out to make contact with the lower module. These pins are not considered in the passive TUImod parts. Other connection strategies need to be found for future redesign, like the smartcard mechanism, which is also used in [McN00].



(a) Routed Board Design of the Arduino Carrier Board

(b) Manufactured Arduino Carrier Board (top)

(c) Manufactured Arduino Carrier Board (bottom)

(d) Routed Board Design of the XBee Carrier Board

(e) Manufactured XBee Carrier Board (top)

(f) Manufactured XBee Carrier Board (bottom)

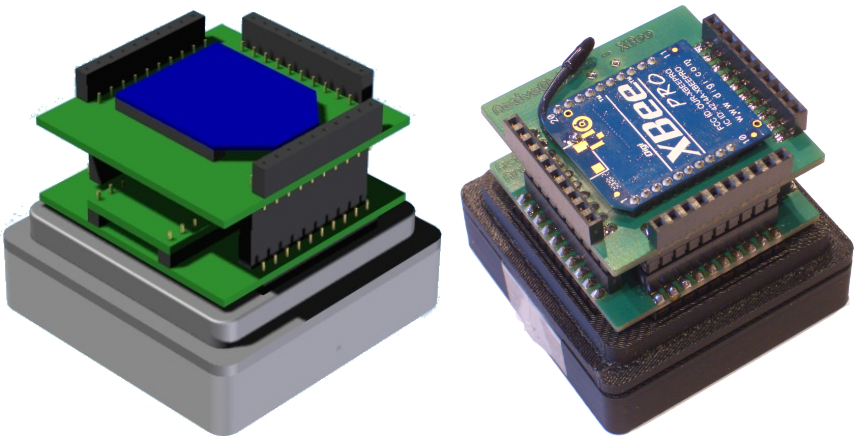
(g) Routed Board Design of the Motor Driver Board

(h) Manufactured Motor Driver Board (top)

(i) Manufactured Motor Driver Board (bottom)

Figure 4.14: The manufactured PCBs

The most important change from a prototype to series production is to change the production to manufactured PCBs. Figure 4.14 shows the results. The schematics described in section 4.2.2 were routed using Eagle (see Section 3.3.2) and the resulting PCB layouts were professionally manufactured by [hak08]. Figure 4.15 shows a ready build series model and Figure 4.16 depicts the complete tDesk setup, including camera and lighting, with three TAOs on the table-top surface.



(a) Design Drawing (b) Assembled TAO

Figure 4.15: Finished TAO series Model without upper housing

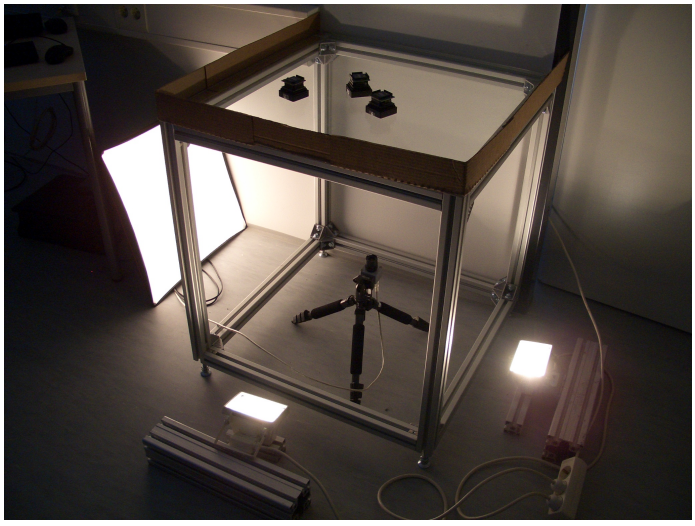


Figure 4.16: The tDesk hardware setup.

4.4 Arduino Firmware

There are existing libraries that allow to remotely control an Arduino microcontroller over a serial connection. For instance there is the Firmata firmware [fir08] and the Messenger library, formerly known as Simple Message System [mes08]. Both are versatile protocols but they consider the serial connection as a point-to-point connection. So addressing multiple microcontrollers is not considered or even implemented. Due to this fact a new simple protocol had to be developed.

4.4.1 SerialControl Protocol

The SerialControl protocol has been implemented for the prototype. It describes very simple human readable commands (ASCII[2] coded strings) that are transmitted to the Arduino over a serial connection. The commands wrap the I/O commands of the Arduino language and extend the wrapped functions for convenience use. The syntax is quite simple: The first statement is the numeric ID. It names the active object that should react on the command. For broadcast purposes this can also be a single '*' as a wildcard. The ID is followed by the command name, which is a unique string. Commands may have up to two arguments. All parts, the ID, the command name and the arguments are separated by spaces (ASCII code 32) and complete command strings are terminated by a colon (ASCII code 59). Six commands have been implemented so far:

ID pinMode Pin Mode; The *pinMode* command makes it possible to change the port property from OUTPUT to INPUT and vice versa. The first argument is the desired port and the second one is the desired mode (OUTPUT or INPUT).
Response: "{ID}: Set mode of pin {Pin} to {Mode}"

ID digitalWrite Pin Value; *digitalWrite* allows to change the state of an output pin. The first argument is the pin number of the output port and the value argument can be HIGH or LOW.
Response: "{ID}: Set pin {Pin} to {Value}"

ID digitalRead Pin; The *digitalRead* command reads out the value of an input pin. It receives only one argument, the input pin. The returning value can be HIGH or LOW.
Response: "{ID}: Pin {Pin} is {Value of the pin}"

ID analogWrite Pin Value; The *analogWrite* command writes analog (PWM) values to an output pin. Like digitalWrite it gets two arguments, the output pin and the new value.
Response: "{ID}: Set pin {Pin} to {Value}"

ID analogRead Pin; *analogRead* reads the value of input pins. In contrast to digitalRead these values are analog values. It receives only one argument, the input pin and returns the value of the pin.
Response: "{ID}: Pin {Pin} is {Value of the pin}"

ID all Mode; *all* is a convenience function. It allows to set all pins of the Arduino to HIGH or LOW, so it gets only one argument, the new state of all pins.
Response: "{ID}: All pins are {Mode}"

[2] American Standard Code for Information Interchange (ASCII)

Additionally special convenience commands for driving have been implemented to reduce the communication overhead:

ID configureWheels; *configureWheels* is used to configure the first four pins used by the motor driver board as output pins.
Response: "{ID}: Wheel output configured"

ID wheelSpeed Left Right; The *wheelSpeed* command sets values of the first four output pins to values, determined by the two arguments, the speed of the left and the right wheel as signed PWM values. This means that the sign determines if the direction port is set HIGH or LOW and the PWM port is set to the actual value.
Response: "{ID} Set wheel speed to {Left}, {Right}"

4.4.2 SerialControl2 Protocol

The SerialControl protocol turned out to be inoperative when used together with multiple XBee modules in peer-to-peer mode. All TAOs reacted on the response of others by resetting. Therefore there is much potential for optimization and simplification in terms of serial communication.
The command set has been reduced and the command syntax has been changed substantially to ease the string parsing on the microcontoller. The semantic remained untouched.
SerialControl2 uses a new command structure: *IDCMARG1ARG2*; The first two characters are used as the ID, so 100 TAOs can be addressed. The next two characters encode the command followed by four characters for each of the two arguments plus the final colon. Thereby the command strings of SerialControl2 have a maximum length of 13 bytes. Because of the fixed length of the commands' parts, no delimiters are needed and parsing the commands has become much easier. The following list opposes the commands used in SerialConstrol and their the corresponding command characters used in SerialControl2:

pm pinMode

dw digitalWrite

dr digitalRead

aw analogWrite

ar analogRead

al all

ws wheelSpeed

db debug

Obviously the configureWheels command is gone. Furthermore a new command *debug* is introduced to allow to turn on normally inactive debugging messages. In summary, the SerialControl2 protocol reduces the communication overhead. Because the command *digitalRead* and *analogRead* are not used in this work, the are only reserved, but not implemented, yet.

5 Software Implementation

This chapter describes the development of the software modules to operate the TAO. The software modules are organized as shown in Figure 5.1. The vision module searches the camera image of a camera, placed underneath the glass surface of the tDesk, and broadcasts the found marker positions to the XCF architecture. Other modules such as path planning and application modules receive these marker positions and navigate the TAOs to new positions by transmitting commands to the transmitter module, which wirelessly transmits the commands to the TAOs. The TAOs react on the commands remotely, which results in new marker positions. A control loop is closed, as depicted in Figure 5.1.

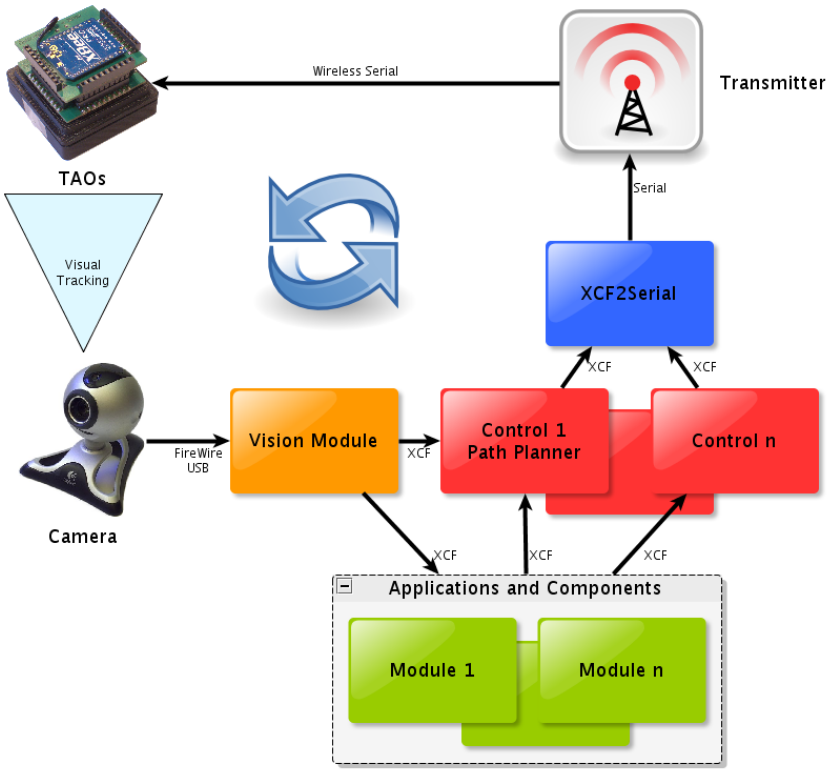


Figure 5.1: Overview of the Software Architecture

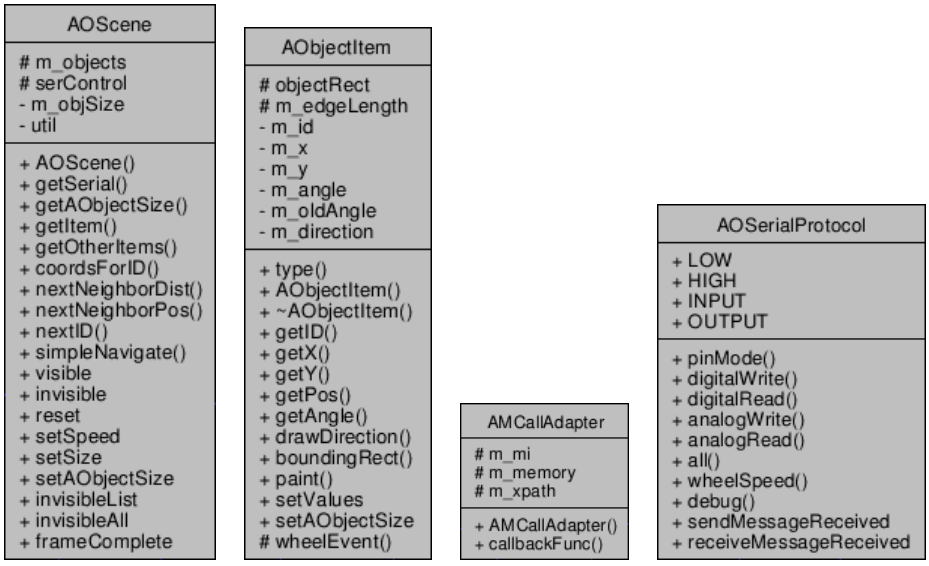
The software is considered to be implemented in modules using C++ programming language and the XCF middleware framework [xcf08] for communication. This makes the TAO system able to be integrated into larger environments like the AmI-Lab [ami08]. The first module implemented is the vision module, that tracks the objects' markers and publishes the tracked positions using XCF. The second module will be the serial relay module, that transmits command strings received over XCF to the TAOs using the XBee Transmitter, introduced in section 4.1.2. Additionally a debugging and monitoring module is implemented to monitor the communication between the TAOs for debugging purposes. A simulation module is developed, too, because software creation and prototype development needed to be done in parallel tasks to save time. This allows to work on the software while waiting for delivery of needed parts. The simulation module simulates TAOs by creating XCF output, such as generated by the vision module. It also reacts on XCF input, that contains command strings and updates the virtual TAOs. For a demonstration in the laboratory of the Ambient Intelligence Group [ami08] a simple demonstration module was created that works together with the hardware prototype to demonstrate simple navigation. Two path planning algorithms were considered for implementation. During the implementation of a cell decomposition approach, known disadvantages of this method could be confirmed in this setup (see Sections 3.5.1 and 5.2.6 for details). That was the reason why the development of the graph-based planning module was canceled and the efford was concentrated on the potential field approach. The Application Programming Interface (API) structures and modules supporting the development of applications makes the base system a handy development environment for future developments.

5.1 Base Classes and their derivatives

This section describes the base classes, introduces in this work. It gives Unified Modeling Language (UML) diagrams and explains the important attributes of the base classes. The base classes' names start with the letters 'AO', which refers to the working title of the TAOs, 'ActiveObjects'.

5.1.1 AOScene

Figure 5.2(a) shows the UML description for the *AOScene* class with it's attributes and properties. *AOScene* is the central widget in terms of Qt. All modules with a GUI visualizing active objects use an instance of this class or it's derivatives. This class itself is derived from *QGraphicsScene*, which provides a versatile frame for graphical visualizations. As Figure 5.2(a) shows, *AOScene* has the protected property *m_objects*, a container to hold representations of tracked TAOs. This container is maintained by Qt slots. These are *visible*, *invisible*, *invisibleList* and *invisibleAll*. The *visible* slot adds or updates a representation and the slots starting with 'invisible' remove objects lost by



(a) AOScene (b) AObjectItem (c) AMCallAdapter (d) AOSerialProtocol

Figure 5.2: UML definitions

the tracking system (see API documentation for details). Beside the information about the tracked objects, the tracking module transmits information about the camera image, because the positions of the objects are given in pixel coordinates. To be notified about the camera image size, the *setSize* slot is used. There is also the slot *frameComplete* that is called whenever the processing of a camera image is completed. Additional there are slots for GUI connections. The *reset* slot transmits '*wheelSpeed* 0 0;' commands to all TAOs. Thereby it acts like a panic button. *setSpeed* and *setAObjectSize* can be connected to widgets to set the desired speed of the active objects in PWM values or to set the size (edge length) of the object's visualization. *setAObjectSize* has a getter function called *getAObjectSize*. There are also a few getter functions that return different properties or can be used for convenience. The *getSerial* function returns the *SerialAdapter* instance. *getItem* returns the instance of a *AObjectItem* with a given ID and *getOtherItems* returns a list of all items except the given ID. To get the coordinates of an *AObjectItem* instance, the convenience function *coordsForID* can be used. The functions *nextNeighborDist*, *nextNeighborID* and *nextID* are used for convenience to get the distance, position and ID of a TAO next to the given position excluding the given ID. Finally the *simpleNavigate* function implements a naive navigation of a given ID to the given position.
For specialization some derivations have been created. See the API documentation on the enclosed CD for details. The derived classes are:

- *AOGraphScene* is used in the GraphPlanner module. This scene is highly specialized and depends on additional *QGraphicsItem* derivatives to allow the computation of the cell decomposition in a graphical manner.
- *AOPotentialScene* is the scene implementation for the PotentialPlanner module. It is specialized for the potential field navigation approach.
- *AOSimpleScene* is used in the SimpleNavigator demonstration module.
- *AOSimScene* is part of the ObjectSimulator module. It is able to generate XCF output for vision simulation and adapts simulated objects according to XCF serial commands. It also allows to manipulate the simulated active obejcts with the mouse.

5.1.2 AObjectItem

Figure 5.2(b) shows the UML inheritance graph for the *AObjectItem* class. Beside *AOScene*, *AObjetItem* is the most important class, because it is used to visualize the TAOs on the *AOScene*. Because *AObjectItem* is the representation for tracked TAOs, it contains properties such as *m_id*, *m_x*, *m_y* and *m_angle* and corresponding getters and the *setValues* setter. As being a derivative of *QGraphicsItem*, this class implements the *type*, *getBondingRect* and *paint* functions. *paint* is very important for classes derived from the *QGraphicsItem* class. This method is responsible for the visualization of the items. Here a *QPainter* instance is used, which implements a very versatile 2D drawing state machine.
For further information have a look at the API documentation on the enclosed CD. Derivatives:

- *AOGraphItem* is used in the GraphPlanner module. It adds several additional *QGraphicsItem* derivatives to the scene to compute the cell decomposition in a graphical manner.
- *AOSimpleItem* is used in the ObjectSimulator module. Here the *wheelEvent* is used to enable rotation of simulated objects using the mouse wheel.

5.1.3 AMCallAdapter

AMCallAdapter is not a specialized for the TAO system, but it is introduced to ease the connecting to ActiveMemories (that is the reason, why it starts with 'AM' and not 'AO'). It provides a base class to encapsulate the registration of a virtual callback function for convenience use. This class is not productively usable. It needs to be derived, to implement functionality in the callback function. In the derivatives Qt slots can be emitted in the function when parsing the received information. Derivatives:

- *AOMarkerListAdapter* is implemented to react on XCF 'MarkerList' content. It is enabled to emit *visible*, *invisible*, etc. signals that can be connected to the slots of *AOScene*. Once these connections are established, visualization of tracked objects is already working.

- *AONavigateAdapter* implements parsing of XCF 'Navigate' messages. The signals emitted fit to the *navigateID* slots of the *AOPotentialScene*.

- *AONavigationStateAdapter* reacts on XCF 'Navigation' messages, which represent states of running navigation tasks. So it emits *navigation*(*id*, *state*) and *navigation*(*state*) signals. Here *id* holds the ID of the navigating TAO and *state* describes the different states ('reached', 'complete'). The latter signal is used for the navigation state of multiple TAOs. That is the reason for the missing ID argument.

- *AOSerialAccessAdapter* is a low-level implementation of an adapter, reacting on *SerialAccess* XCF messaged and is used by the XCF2Serial module.

- *SerialAdapter* is a higher-level implementation (in contrast to *AOSerialAccessAdapter*). It is equipped with Qt signals and slots that allow easy gating to Qt GUI programs. Before the development of the *AOSerialProtocol* class, this was the central class for transmitting serial messages to the TAOs over the XCF architecture.

5.1.4 AOSerialProtocol

The upcoming of an improved serial protocol (see section 4.4) makes an abstraction layer beneficial to easily change from one protocol to another. *AOSerialProtocol* is an almost virtual class that to defines the frame for the intersection of both implemented serial protocols. It defines function bodies that wrap the commands introduced in the serial protocols. Additional Qt signals are defined to encapsulate responses coming from the active objects. The two implementations *AOSerialControl* and *AOSerialControl2* derive from this class to specialize on the two protocols: Derivatives:

- *AOSerialControl* implements the first protocol of the first prototype. It utilizes a *SerialAdapter* instance to communicate to the TAOs.

- *AOSerialControl2* implements command handling for the second protocol SerialControl2. It introduces constants containing offsets to easily access parts of the command string.

Since *AOSerialProtocol* only defines the frame for serial protocol implementations, both implementations add helper functions for convenience use, such as *setSpeed* and *drive*, which abstract the *wheelSpeed* function. Classes derived

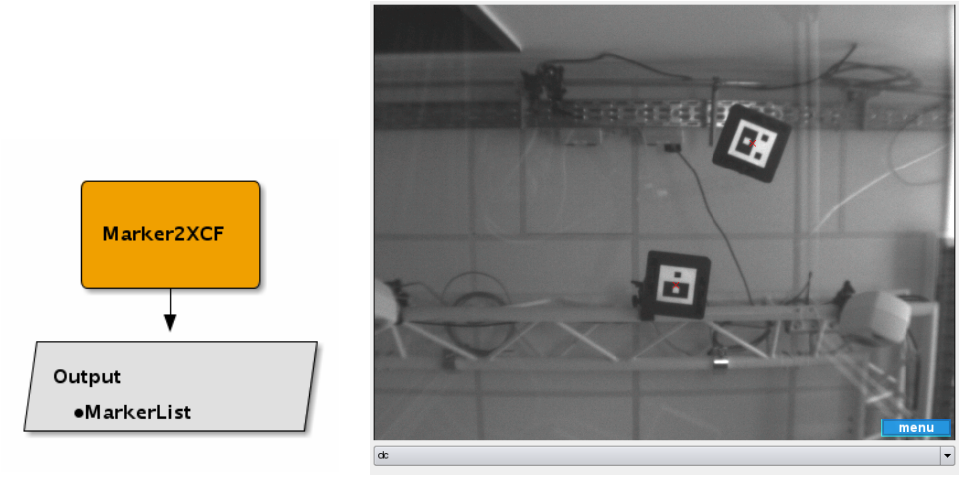
from *AOSerialCode* are meant to make the direct use of *SerialAdapter* unnecessary.

5.2 Software Modules

This section illustrates every base software module, developed in the course of this thesis. To ease the understanding of the XCF data streams, diagrams, describing the in- and output streams of each module, are included in the description of the module.

5.2.1 Vision Module: Marker2XCF

The Marker2XCF module, which is depicted in Figure 5.3(b), is written using the ICL (see section 3.4.1). It reads the image of an attached camera via Firewire (DC), USB (Philips Web Cam (PWC)), Unicap driver or previously recorded image files. Every taken picture of the video stream gets searched for fiducial markers (see section 4.2.3).



(a) Marker2XCF XCF In- and Output diagram

(b) Screenshot

Figure 5.3: Marker2XCF module

Every occurrence of a marker gets stored in a XML document like printed below. The document contains a tag called *MarkerList* which takes every occurrence of a marker in the current frame. Each occurrence is documented by it's *angle* followed by *x* and *y* value. Due to compatibility with other object describing XML documents already used in the Neuroinformatics Group, the attributes *kind*="*relative*" and *ref*="*image*" are included. Additionally the timestamp is attached to the list.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<MarkerList imageHeight="480" imageWidth="640">
  <TIMESTAMP>
    <INSERTED value="1227121349" />
  </TIMESTAMP>
  <Marker id="25" type="fid">
    <coord angle="2.997053146362305" kind="relative"
      ref="image" x="239.4224548339844" y="235.7020416259766"
        />
  </Marker>
</MarkerList>
```

To be able to track different types of fiducial markers, the module expects a file named *all.trees*. Like introduced in section 4.2.3, the configuration of the heavy left sided trees of the markers are stored in this file. For the first four markers depicted in Fig. 4.12 such a file has the following content. The IDs are assigned line wise starting with 0:

```
0121
01221
01211
012211
```

To enable the module to track markers of other sets, only the configuration of these sets need to added to the *all.trees* (or completely replaced). The quite popular amoeba set, introduced by the reacTIVision software [rea08] is also included on the enclosed CD in the module's source directory.

5.2.2 XCF to Serial Module: XCF2Serial

The XCF2Serial module relays messages received via XCF to a serial port and transmits received data from the serial port back into the XCF architecture. Once the command-line only module is started, the module looks for a USB attached serial port, such as */dev/ttyUSB0* and starts listening for incoming bytes. The module is designed to be as common as possible, so it is possible to control nearly every serial device available, e.g. the Ambient Lights [HHK+08], introduced in the AmbiD system [BHR06], or digital scales as another example. The following code examples describe the XML structure, used to exchange data with the serial port from XCF enabled programs.

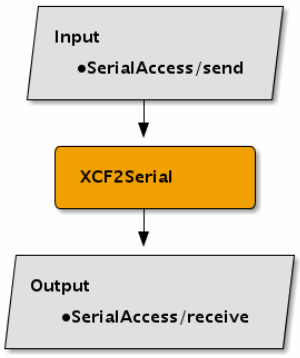


Figure 5.4: XCF2Serial XCF In- and Output diagram

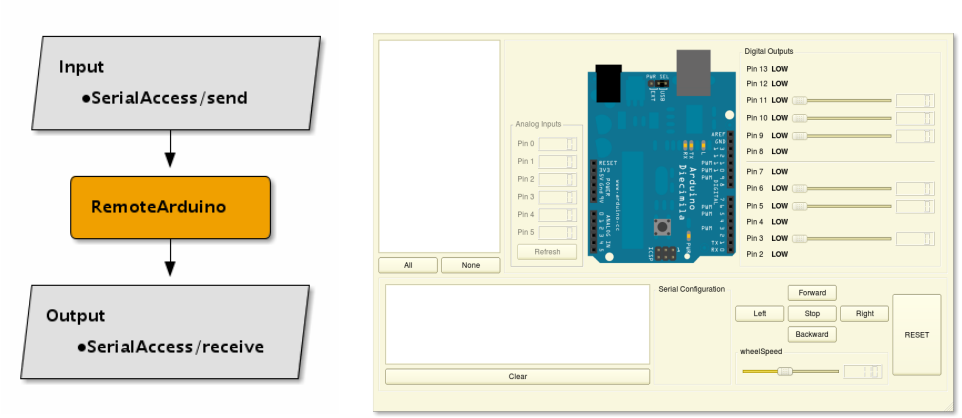
```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SerialAccess>
  <send message="0 wheelSpeed 255 255;" />
</SerialAccess>
```

Of course the XCF2Serial module is enabled to read bytes from the serial port and transmit them back via the following XCF string:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SerialAccess>
  <receive message="0: Set wheel speed to 255 255" />
</SerialAccess>
```

5.2.3 Arduino Monitoring and Administration: RemoteArduino

For debugging purposes, the RemoteArduino module is created to monitor serially attached TAOs at the XCF-Layer (see Fig. 5.5(b)). It views a list of all IDs of recognized objects in the XBee network. When selecting one object, the pin states of the object are represented in the GUI. Additionally it is possible, to change this state directly from within the GUI. This can be done with one object, a selection of a couple of objects and the changes can be broadcast to all objects in the network.



(a) RemoteArduino XCF In- and Output diagram

(b) Screenshot

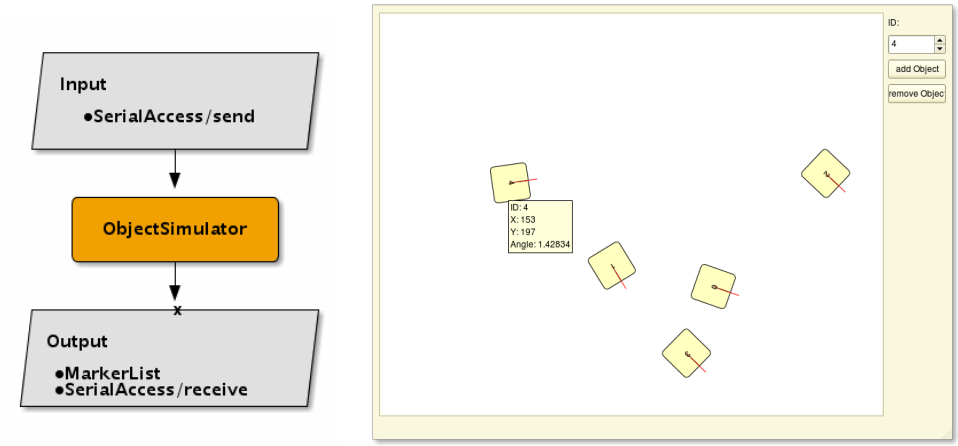
Figure 5.5: RemoteArduino module

While RemoteArduino wraps all commands, provided by the serial protocols, not only direct pin control is possible. Also higher-level control, such as remote controlling the TAOs' moving abilities by hand. There are buttons for driving forward, backward and steering commands, a stop and a reset button button and a slider for determining the desired velocity in PWM values. Finally there is a logging widget, that logs all transmissions received over XCF.

5.2.4 When no real Objects are available: ObjectSimulator

The ObjectSimulator module acts like a virtual replacement of the table-top interface and the active TUIOs. Figure 5.6(a) shows that the module receives the

/SerialAccess/send as input and sends in response */SerialAccess/receive* and */MarkerList* as output messages. The module administrates a list of objects that the user can create instantly using the GUI, depicted in Figure 5.6(b). The user has the opportunity to move the objects around using the mouse and drag and drop. With the mouse wheel it is possible to rotate the objects. The GUI shows the virtual objects mirrored (due to the simulation of a camera looking from underneath a transparent table-top surface). Of course, these virtual objects are controllable with the SerialAccess messages and react just like real objects would. Like the vision module, this module stores the object configurations and it's changes in the active memory in every iteration.



(a) ObjectSimulator XCF In- and Output diagram

(b) Screenshot

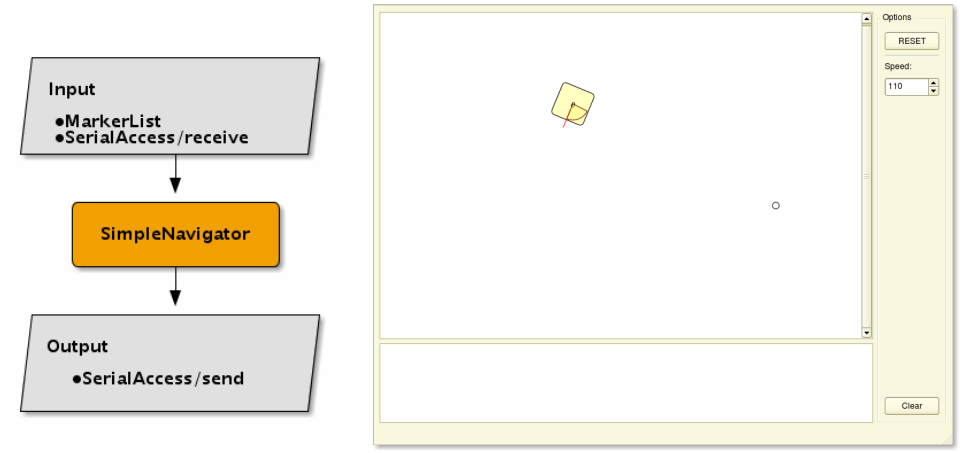
Figure 5.6: ObjectSimulator module

With these abilities this module replaces completely the hardware setup and allows to use the system without having access to the not very portable hardware. It was also very useful as long the parts for the series production of the dozen of objects did not have arrived.

5.2.5 Demonstrating Object Navigation: SimpleNavigator

This module demonstrates a naive navigation scenario. SimpleNavigator only copes with one TAO. By clicking a desired target position in the GUI, the SimpleNavigator module measures the angle between the heading direction of the TAO. At first this angle gets minimized by rotating the object in the direction of the goal position. When the direction is reached ($\pm 3^\circ$ left/right tolerance) the object moves forward. If the angle between heading direction and target increases and crosses 45°, again this angle gets minimized by rotation and so on. When the

distance between the object and goal is almost zero (again with a small tolerance), the goal is reached and the TAO stops.



(a) SimpleNavigator XCF In- and Output diagram

(b) Screenshot

Figure 5.7: SimpleNavigator module

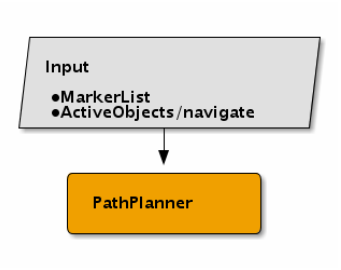
With this simple toy example it is possible to demonstrate the correct motion of the active TUIOs.

5.2.6 Graph-based Path Planning Control Module: GraphPlanner

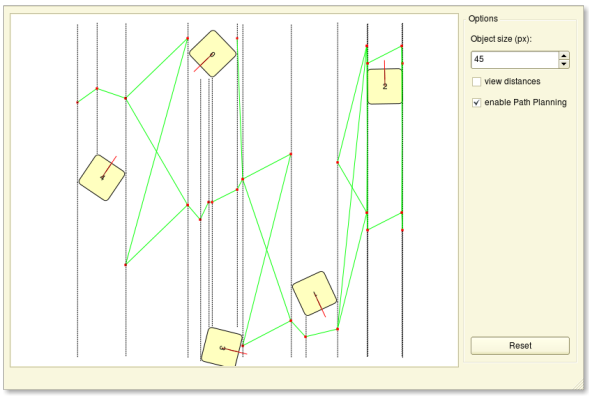
The GraphPlanner module implements an approximate cell decomposition approach (see section 3.5.1) almost completely. In every iteration, the path-planning module generates graphs from the vision-input that covers C_{free}.
As Figure 5.8(b) shows, the so far computable path elements are not optimal. Except the active object with ID 4, all objects are tightly surrounded by path elements. Using these paths would unavoidable result in a collision between TAOs. This is the reason why the development of this module was canceled.

5.2.7 Potential Field-based Path Planning Control Module: PotentialPlanner

This navigation approach proposed in [Kha86] uses artificial potential field simulation to guide mobile robots around obstacles to the desired goal (see section 3.5.3). By introducing an attractive force between the TAO and the target and repulsive forces between the TAO and other TAOs, it is guided around them to the goal position. After adding the attractive and the repulsive potential field, descending on the gradient of the field creates a trajectory, that guide the TAO to the desired target. It is necessary to derive the potential field to get the gradient, which results in a vector field of forces.

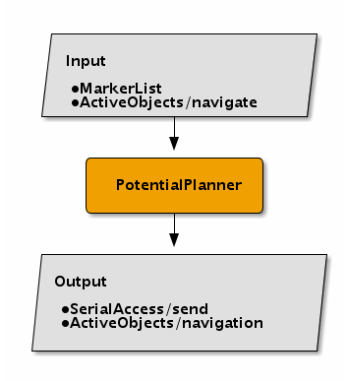


(a) GraphPlanner XCF In- and Output diagram

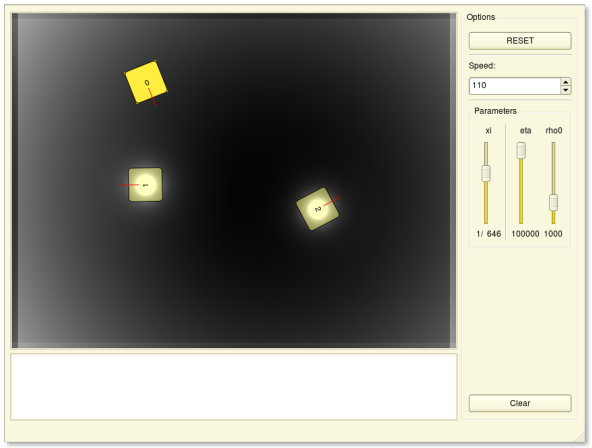


(b) Screenshot

Figure 5.8: GraphPlanner module



(a) PotentialPlanner XCF In- and Output diagram



(b) Screenshot

Figure 5.9: PotentialPlanner module

To make the PotentialPlanner start navigating objects to new positions, it awaits incoming XML documents in the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<ActiveObjects>
  <navigate id="0" toX="137" toY="423" toAngle="1.56294" />
  <navigate id="1" toX="26" toY="115" toAngle="3.05406" />
  <navigate id="2" toX="309" toY="241" toAngle="4.02461" />
  <navigate id="3" toX="296" toY="41" toAngle="0.23969" />
</ActiveObjects>
```

Once, navigation has started, the module reports about the state of each object and the complete navigation task by transmitting XML documents with the following patterns until the navigation task is completed.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<ActiveObjects>
  <navigation id="2" state="started"/>
</ActiveObjects>
<ActiveObjects>
  <navigation id="1" state="reached"/>
</ActiveObjects>
```

When the navigation task is completed and all objects have arrived at their new positions, the PotentialPlanner transmits the following XML document to the ActiveMemory:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<ActiveObjects>
  <navigation state="complete"/>
</ActiveObjects>
```

Thereby all modules awaiting XCF messages with the XPath */ActiveObjects/navigation* know about the state of running navigation tasks. This is important, if module e.g. sonify the position or interaction with TAOs being moved. During navigation these sonifications can then be diabled and started again when it is completed. By reacting on those messages other events could be initiated, such as speech output to signal the user that something is happening and interaction with system is temporally not possible, etc.

6 Applications

This chapter describes two applications for TAOs that have been developed in the course of this work to demonstrate the approach and potential of TAOs for HCI. Every application is implemented as an additional module that is integrated into the XCF communication of the existing modules. The first one allows to save and restore arrangement of TAOs used as ordinary TUIOs in existing TUIs. The second application proposes TAOs for exploratory data analysis.

6.1 Saving and Restoring TAO Locations for existing TUIs

Except the actuated workbench (see section 2), there is no table-top-based TUI known to us that is able to save and restore the state of the TUIOs' positions on the table. Comparable to a 'save as' function in GUI programs, the TAO system enables the user to save and restore positions and orientations of TAOs. This capability allows to save the current work for later continuance. This application can be used in combination with existing TUIs that fulfill the criteria of function assignment [FIB95] to be permanent (see section 3.2). If the function assignment is programmable or even transient and not space-multiplexed, the controlled system needs to be able to save the assignment, because the controlling TAO system is not able to do this without repeating all interactions that led to the state. For this reason only systems with permanent function assignment are considered.
Examples for existing controllable systems:

AmbiD The Ambient Information Display (AmbiD) [BHR06] allows the user to arrange data sources and sinks (ambient information displays).

TI-Son The Tangible Interaction Sonification System (TI-Son) [HBRR07] is a TUI that allows to spatially arrange channels of an Electroencephalography (EEG) sonification and to place a representation of the listener in this arrangement. The sonification runs in real-time on a 8-channel audio setup, that is installed around the tDesk running the TI-Son TUI.

AudioDB The Audio-Database System (AudioDB) [BEHR08] is used to explore and group and organize large sets of audio chunks. The user can group similar sounds into clusters of TUIOs.

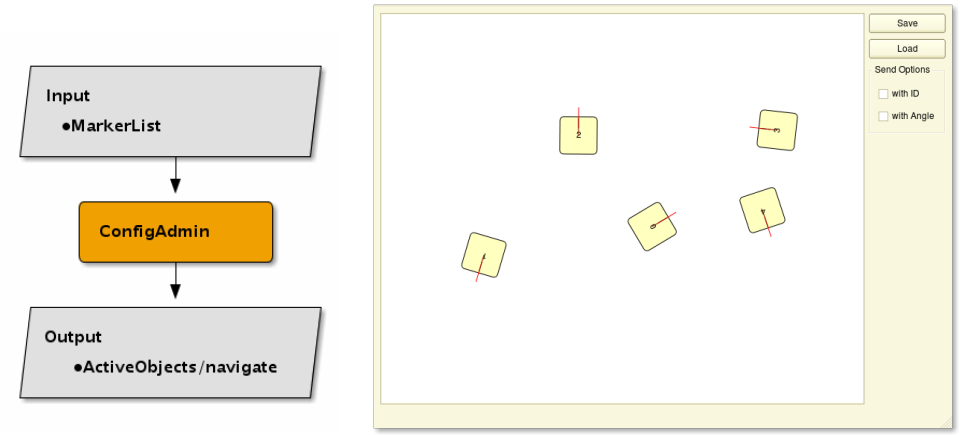
The TAO system can immediately be intrgrated in these existing applications and offers the service to save and restore object arrangements.

6.1.1 Interaction Design

In addition to the existing TUI, e.g. AmbiD, the user starts the TAO system and this application module, called 'ConfigAdmin'. The user works with the TUI as normal, except that the TAOs are used as TUIOs of the existing TUI. To store a TAO configuration in a file, the user presses the 'Save' button in the GUI of the module. A file dialog opens, asking the user to specify a filename under which the configuration is saved.
Later it is possible to restore a configuration using this application module. By pressing the 'Load' button, the user is asked by a file dialog to determine a file containing a configuration. The configuration is then loaded and the TAOs move to the positions stored in the file. The use of the existing TUI can continue using the TAOs at the restored positions.

6.1.2 Implementation

The ConfigAdmin module, depicted in Figure 6.1(b), allows to store object configurations on the interactive table-top surface in a simple text file from the `/MarkerList` messages. For storage it is irrelevant if the tracked objects are active or passive TUIOs. Only when loading configurations to restore the configurations using the `/ActiveObjects/navigate` message (see section 5.2.7), the TUIOs need to be able to move. This allows to store configurations of TUIs and reload them later again.



(a) ConfigAdmin XCF In- and Output diagram

(b) Screenshot

Figure 6.1: ConfigAdmin module

The file format is a simple Comma-Separated Value (CSV) text format. The following lines give an example:

```
1 0;494;290;0.437307
2 1;331;284;4.33668
3 2;101;83;1.42768
```

Every line represents a position and orientation of a certain TAO. The values are delimited by the colons. The first value is the ID of the TAO. The second and the third value describe the X and Y value of the TAOs' position. The position is stored in pixel-coordinates of the camera image. The last value describes the orientation of the TAO in radians between 0 an 2π relatively to the camera image. For instance the TAO with the ID 1 is located at pixel coordinates 331, 284 and heads to 284 degrees. For the prove of concept the use of the camera pixel coordinates is sufficient. But the development of later applications can benefit from a coordinate system, that is aligned to the tDesk and uses metric units.

6.1.3 Discussion

As already mentioned, the combination with TUIs using TUIOs with non-permanent function assignment is difficult. The existing TUI would have to support the saving of function assignment, which requires modification of the TUI. This first implementation needs an additional GUI for the ConfigAdmin module. The user has to switch between the TUI and the GUI, which is not desirable. To get rid of this problem, additional TUIOs can be introduced in a later implementation that can act as a container for configurations. To save an arrangement, such a TUIO can be placed on the table-top surface. Placing such a TUIO that represents a configuration, restores that arrangement. This problem can also be solved by making use of the multi-touch capabilities of the tDesk [Tü09]. If these additional TUIOs or the multi-touch projection disturb the controlled TUI, speech recognition can be another way to cope with that problem by introducing a simple command system, which can make the GUI almost unnecessary.
Nevertheless this simple application demonstrates the great potential of the TAO system. New scenarios are possible, that were so far not considered.

6.2 Dataset Visualization: DataProcessor

TUIs take advantage of the human's spatial awareness. This can be very useful for exploratory data analysis. The use of TUIOs lets the user bind 'his immediate environment to dedicated points in model-space and therefore constructs a virtual map of the data itself' [BHR]. By clustering high-dimensional datasets using cluster-algorithms it is possible to use TAOs to represent the found centroids on the table-top surface and give the user a physical handle for active exploration of these data. To enable the exploration, different interactive sonification approaches can be utilized.

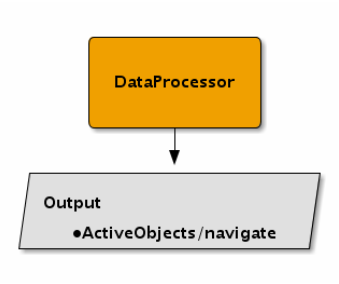
6.2.1 Interaction Design

In a typical interaction scenario, the user selects a dataset by pressing the 'load' button and selecting a CSV file from the opening file dialog. The dataset is then instantly visualized in a 2D scatter plot in the GUI (see Fig. 6.2(b)). By changing the plotted dimensions, the visualization can be adjusted. Before the clustering can be started by clicking the 'Cluster' button in the GUI, the number of desired clusters can be adjusted using the appropriate widget. After the clustering process, the TAOs are navigated into the computed cluster centroids as their prototype representations.
Two interaction scenarios are considered so far:

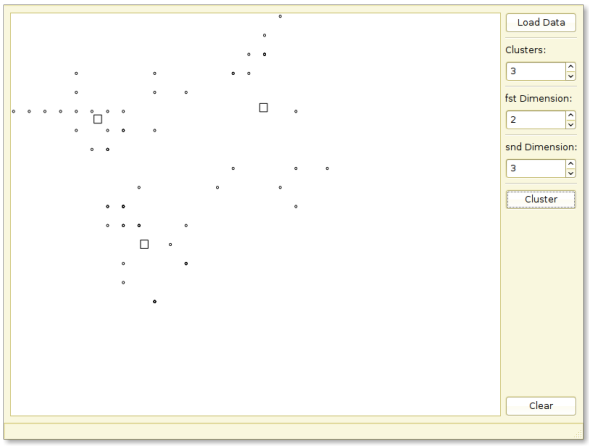
- The first scenario utilizes different model-based sonifications [HR99]. By shaking single TAOs representing cluster centroids, the sonification attached to the data model can be excited for data exploration. Model-based Sonification (MBS) provides data-driven sonification techniques, that allow to analyze the underlying data exploratory.
- Another interaction scenario describes the exploration of clusters by sonifying the cost/energy change while moving the cluster's centroid representative TAO and thereby it's prototype. This allows the exploration of the cluster structures.

6.2.2 Implementation

This application can be used to visualize datasets with the TAOs. The TAOs can be remotely moved to the cluster centers of a dataset to visualize the data. This application module uses the Cluster library (see section 3.4.5), to process the loaded dataset.
The user is able to load a dataset and to visualizes it in a 2D scatter-plot, which is part of the GUI, depicted in Figure 6.2(b). It is possible to choose the number of clusters and the desired dimensions for the visualization. Clustering can be started by clicking the 'Cluster' button, as described in the Interaction Design section. The cluster centers calculated by the Cluster library are used to generate a 'Navigate' XML document string in a format such as introduced in section 5.2.7. If the PotentialPlanner module is running, the available TAOs are navigated to the cluster centers.
Videos on the enclosed CD demonstrate this application (see `/videos/` directory of the CD, files `appDemo1.mp4` to `appDemo3.mp4`). All three videos show the selection of a dataset in the GUI, it's clustering and the navigation of two TAOs to the computed cluster centroids from two perspectives. The first perspective shows the navigating TAOs from above the tDesk, the second perspective shows them from the user's point of view.
Due to lack of time, the interaction methods of this application could not be implemented, yet. To make the application interactive, the module must be



(a) DataProcessor XCF In- and Output diagram



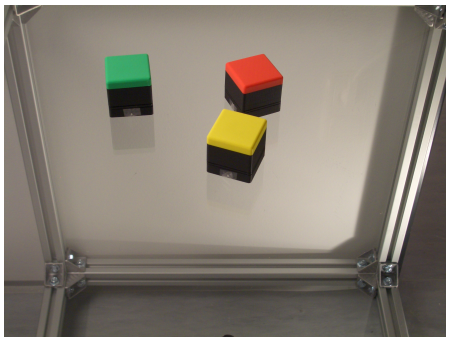
(b) Screenshot

Figure 6.2: DataProcessor module

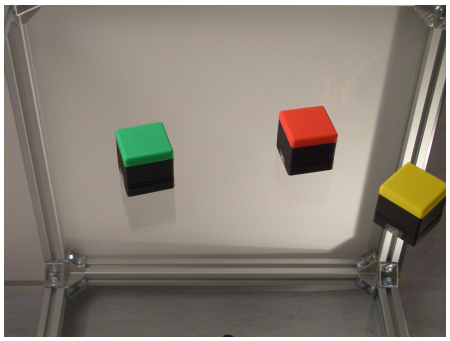
made aware of the TAOs' tracked positions. Now these data can be used to determine the inter-cluster distance of the data points and the prototype to sonify the cost/energy change. Here an Open Sound Control (OSC) connection to the sound server of the AmI-Lab [ami08] running an instance of the SuperCollider [sc08] synth server would be beneficial.
In the second interaction scenario such a connection could be used. The application module computes the model of the sonification and control the synth server. E.g. a Particle Trajectory Sonification [HR99] can be created from each cluster. The shaking of the TAO representing the cluster would excite the sonification.

6.2.3 Interaction Example

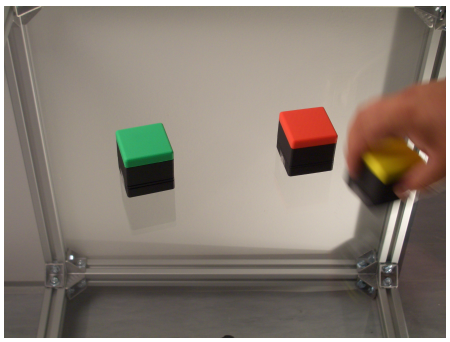
The following interaction example is illustrated here with photos (see Fig. 6.3). Photo 6.3(a) shows the TAOs unarranged and unassigned. As described above, loading a dataset and clustering it makes the TAOs move to their assigned centroids (see Fig. 6.3(b)). When the centroids are reached, interaction can start. By shaking a TAO its corresponding cluster is excited for instance by injecting energy to all data points belonging to the cluster, e.g. using the data sonogram model described in [Her02] (see Fig. 6.3(c) and Fig. 6.3(d)). In result, a sound would be generated that allows to distinguish the distribution of points to classes, in case that the class feature would have been responsible for the spring stiffness in the Data Sonogram model. As benefit compared to traditional interfaces, two handed interaction can be used to quickly compare sonification of different clusters.



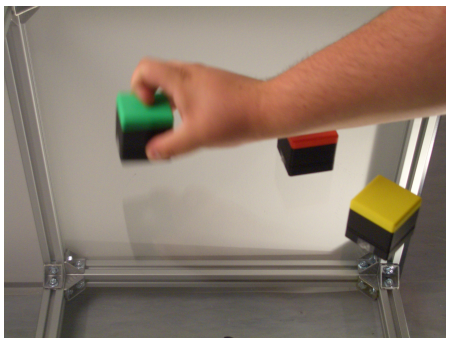
(a) Unarranged TAOs are on the table that do not represent data so far



(b) After loading and clustering a dataset, the TAOs move to the cluster centroids



(c) Excitation of the MBS attached to the yellow TAO by shaking it



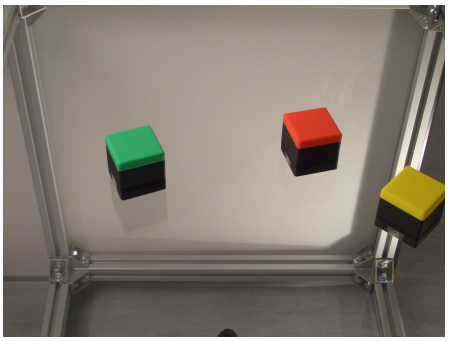
(d) Excitation of the MBS attached to the green TAO by shaking it

Figure 6.3: Interaction Example: Exploring a dataset

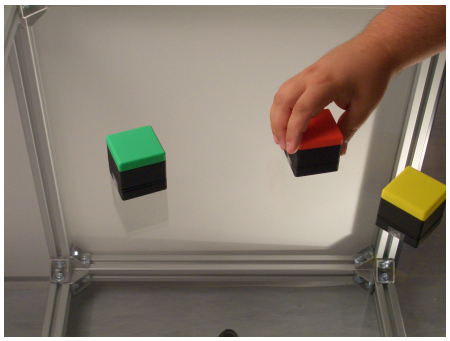
When the user moves a TAO it is propably not returned to the exact position of the clusters' prototype, it represents. Therefore the TAOs are navigated back to their prototypes' position when put back on the table surface. Figure 6.4 shows the sequence of this interaction. The file *appDemoDetail.mp4* in the *videos* directory demonstrates the interaction with a single TAO as described here.

6.2.4 Discussion

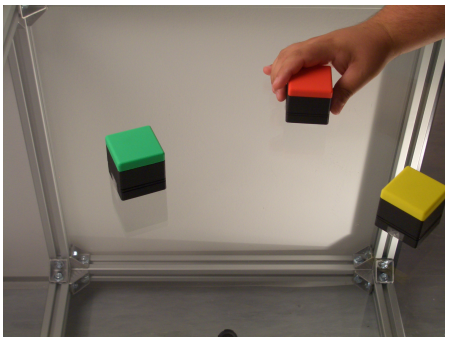
This application requires a cognitive switch between the TUI and the GUI. This application can be extended to cope with this problem by introducing an interactive projection, that allows load and visualize the dataset on the table-top surface. A similar concept has already been introduced in the TRecS project [DHKR08], that allows to assign TUIOs with functionality or data and interact with them. All this is done using a TUI. The multi-touch application described in [Tü09] can be used as an alternative to solve the problem of switching between the TUI and the GUI.



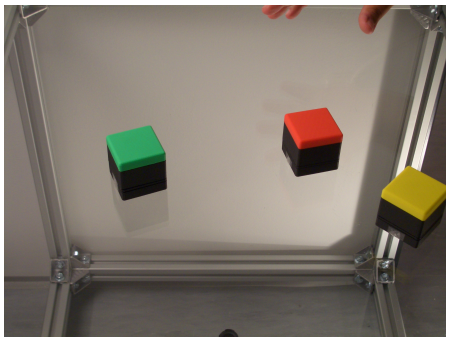
(a) Assigned TAOs at the centroid positions



(b) Interaction with a TAO (moving)



(c) Moving the TAO away from it's centroid



(d) The TAO moves back to it's old position

Figure 6.4: Interaction Detail: After moving a TAO to explore its underlying data, it moves back to the exact position of its prototype, the TAO represents.

Although it is not completely implemented, this application proposes rich interaction methods, that underline the potential of the TAO system. It gives a first insight into this pretty novel interactions.

7 Evaluation and Observations

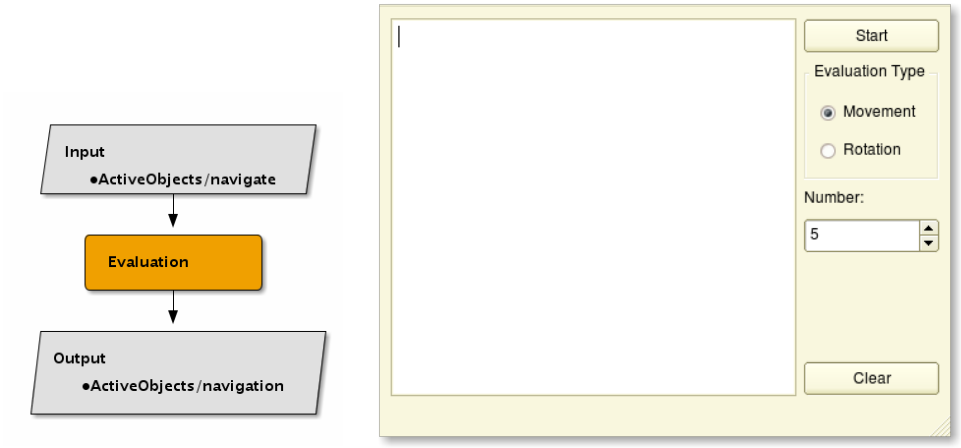
This chapter discusses the performance of the TAOs that were designed in the curse of this project according to aspects such as their accuracy in reaching targets, velocity and operating time. Additionally suggestions for the evaluation of interactions with TAOs are given. The following subsections cover the implementation of an additional module for evaluating these parameters and the results of the evaluation. Finally suggestions for further evaluation of interactions with the TAOs are considered.

7.1 Performance Specifications

Linear motion and rotation are the movement options that are implemented as active modes in the TAOs. In this section the accuracy of this motion and it's linear and angular velocity, as well as the operating time of the batteries are discussed. These parameters are used to evaluate the performance of the TAOs.

7.1.1 Implementation

For evaluation purposes, a module, that allows to generate a random trajectory and navigates a TAO along the different targets of the trajectory, was developed. The distance between two successive points on the trajectory as well as the time a TAO needs to travel from one point to the other is measured. All measurements are written into a file of the format CSV for later analysis.
To evaluate the performance of a single TAO, it is placed in the middle of the table-top surface. Two kinds of evaluation data can be collected by, checking either the 'movement' button or the 'angle' button, of which only one button can be selected at the same time. The number of points or angles of the trajectory can be adjusted by changing the number in the designated widget before starting the evaluation. By clicking the 'start' button in the GUI of the module, depicted in Figure 7.1(b), the evaluation starts. The evaluation runs until the end of the trajectory is reached. If the 'movement' button is checked, the euclidean distance and the time between two targets are stored in a file. Accordingly when the 'angle' button is checked, the angle difference and the elapsed time between two targets are stored.



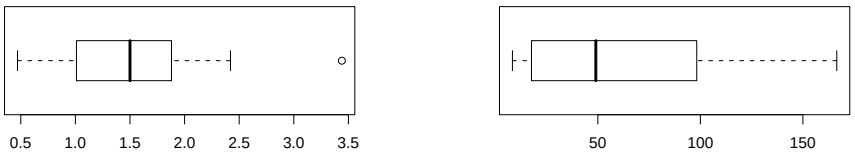
(a) Evaluation XCF In- and Output Diagram

(b) Screenshot of the module's GUIs

Figure 7.1: This Figure shows (a) a Diagram to depict the XCF in- and ouptu streams and (b) a screenshot of the Evaluation module's GUI.

7.1.2 Results

For the experiments that were conducted to evaluate the performance of the TAO, five trajectories consisting of five target positions and three trajectories consisting of six target angles respectively were analyzed. The evaluation was filmed with a video camera. The videos can be found in the */videos/* directory on the enclosed CD. The MPEG4 encoded video files *evalLinear1.mp4* to *evalLinear5.mp4* show the evaluation of the linear velocity and the video files *evalRotation1.mp4* to *evalRotation3.mp4* show the evaluation of the angular velocity. After the removal of outliers, caused by implausible measurement, 18 points for each movement and rotation remained. By adding the distances of



(a) Velocity (cm/s)

(b) Angular Velocity (degrees/s)

Figure 7.2: Evaluation Results

	Linear Velocity [cm/s]	Angular Velocity [degrees/s]
Mean	1.56	62.31
Median	1.5	49.01
Minimum	0.47	8.34
Maximum	3.44	166.53
Variance	0.58	2270.28
Standard deviation	0.76	47.65

Table 7.1: Statistical values computed from the gathered raw data

difference angles, the evaluated TAO travelled approximately 1,60 m or turned about six times around it's own axis for rotation evaluation. The following results are gathered with a PWM speed of 90, using a series type TAO.
The accuracy according to reaching a certain target position or angle is determined by the path planning and the vision module. It is possible to reduce the threshold down to one pixel (then the accuracy is only determined by the vision). For the evaluation the threshold around the target position is set to 10 pixels in diameter, which corresponds to 0.8 cm on the surface of the tDesk. The rotation threshold was set to 5° around the target angle.

Velocity The evaluation of the movement velocity (see the box-whisker-plot Figure 7.2(a) and the Table 7.1) showed that to reach a goal that is 1.5 cm afar, a TAO needs and average of approximately one second. The average velocity is 1.56 cm/s with a standard deviation of 0.76 and a variance of 0.58. When there is much need for direction readjustments, the approximate velocity decreases to a minimum of 0.47 cm/s. The outlier at 3.44 cm/s resulted from an observed trajectory stretch, for which no readjustment of the direction was needed. Perfect linear movement of the current TAO generation is unlikely. Due to drifting, caused by inaccuracy in the mechanics, the TAOs drive curves. This shows that there is much room for improvement.

Angular Velocity The box-whisker-plot in Figure 7.2(b) and the Table 7.1 present the values that were measured in the evaluation of angular velocity. The median angular velocity is $w = 49^\circ/s$ and the average angular velocity is $w = 62,3^\circ/s$. The standard deviation shows that much readjustment is needed, to exactly stop at a certain angle.

From the evaluation can be read that a crucial factor for a fast trajectory is the number of necessary rotations. Decreasing the need of direction readjustments can increase the velocity. The angular velocity can be slightly increased by reducing the PWM velocity. Another improvement may be achieved through a faster marker tracking allowing higher frame rates and lower latency to allow faster adaption. Due to the latency of the inter-module communication, the TAOs can have turned

further, until the stop command is received by the TAOs, which again requires readjustment of the direction.
Further evaluations can consider other parameters, such as the average distance from the ideal trajectory. Because this parameter is not important for the presented applications. Accurate trajectory reproduction is not required, but exact reaching of target positions. It is not carried out in this evaluation.

Operation time Experiments have shown, that under permanent load (continuous use with direction changes about every half second and continuous transmission and reception of messages over the XBee radio module) the batteries allow an operation time of about 30 minutes. When the batteries run low, the motors stop and the XBee connection breaks down after two additional minutes. It can be assumed, that a longer operation time is possible, if the system is not used continuously.

7.2 Suggestions for the Evaluation of Interaction

User-studies need to be done to evaluate the interaction with the new TAOs. The reaction of users that are confronted with TAOs for the first time can be investigated by observation and questionnaires. Some users may be frightened of self-moving objects, but the active feedback, given by the TAOs, may also encourage the user to interact with the TUI more than passive TUIOs would do. An additional questionnaire can help to investigate the personal impressions of the users and thus to get suggestions for improvements and extensions of the TAOs.

8 Conclusion

This chapter summarizes the outcome of this work, discusses the results and presents an outlook into possible future research. In this work TUIOs with active feedback capabilities, called TAOs were introduced. To support this idea, a detailed theoretical background and examples of the state of the art were presented. After describing the components, the hardware design was described in detail, followed by the software implementation. Further, the base classes and all developed software modules were explained. Two applications for the proposed system were presented in the applications chapter. Finally the developed hardware was evaluated.

8.1 Discussion

The evaluation of the TAOs showed that even though the system fulfills it's purpose, there is still room for improvement. The wireless communication using XBee modules is good, but when multiple TAOs have joined the network, messages can be lost due to transmission collisions (see Section 4.1.2). A readjustment of the XBee modules' configuration may solve this problem. Otherwise further means such as detailed analyze of the wireless communication may have to be taken to make the connection more robust. Beside the electronics, the mechanic of the TAOs has potential for improvements. Since all parts are produced with a 3D rapid-prototyping printer, the results are not as accurate as the models that were constructed in the CAD program. For this reason the wheels of the TAOs can seize or the rubber belts can slip. In both cases the TAO is restricted in it's movement of may not move at all. These problems can be solved by turning new wheels on a turning lathe and installing ball bearings on the wheels' axes, to make the mechanics move much smoother. Additionally the software implements the potential field approach in a direct way, which leaves room for further improvements.
While these suggested improvements offer a stating point for further research, this work successfully implemented the basic concept of TAOs and discussed their possible applications, as the demonstration and evaluation videos show. The new kind of TUIO that was introduced and described in detail gives the basis for extensive further research in this area as part of the research field of Tangible Interaction.

8.2 Outlook

The first next step of further research would be the implementation of the proposed interaction methods of the application for exploratory data analysis. A new point-to-point navigation that controls the wheels of the TAOs dependent on the angle to the target, and thus adjusts the rolling direction accordingly to the linear movement may lead to significant performance improvement. This would result in curved trajectories and probably a much faster target attainment. A detailed analysis of the wireless serial communication and the XBee radio modules can disclose reasons for the transmission problems that were mentioned.
Beside these general improvements, the functionality of the TAOs could be extended by introducing new active modalities. So far only the capability of linear movement and rotation is implemented; however visual feedback could be beneficial to support multi-modal feedback. It could be implemented by introducing a new hardware layer, that contains a small display (e.g. an Organic Light Emitting Diode (OLED) display) on top of the TAOs. This would allow to display words, symbols or even simple animations.
To include the proposed sonification into the TAOs, another hardware layer equipped with a small piezoelectric speaker and a synthesizer chip could be developed. We believe, that this may lead to a shorter interaction loop, because the sound comes directly from the TAOs, which represents the underlying data of the sonification model.
Tactile feedback could add an additional output modality. A new hardware layer that contains pager motors, such as the ones that are integrated in common cellular phones can be used to allow tactile feedback. Providing tactile feedback can encourage the user to interact with TAOs physically.
All these extensions require adaption of the used communication protocols. More commands need to be introduced to allow the handle the additional modalities. The resulting new interaction possibilities promise to allow a wide range of new applications. The TAOs could for instance be used to synchronize the configuration of two or more tDesks and this would enable long-distance cooperative work and communication. Regarding the evaluation of the system, interaction recording and replay can be interesting features that could be used to learn and analyze how TUIOs and TAOs are used and about how they can be improved to better suit the needs of the user. We hope that TAOs will make a useful contribution to improve TUIs and HCI in general.

Enclosed CD

The enclosed CD contains the complete source code of this work including the API documentation and firmware. It also includes technical drawings of the hardware and circuit diagrams, demonstration videos of the developed system and this thesis in Portable Document Format (PDF). The complete directory tree of the enclosed CD is as follows:
CD root

- code
  - bin
  - doc
  - firmware
  - src
    - configadmin
    - dataprocessor
    - marker2cf
    - objectsimulator
    - potentialplanner
    - remotearduino
    - simplenavigator
    - tools
    - xcf2serial
- drawings
  - housing
  - schematics
- thesis
- videos

Installation Remarks

The TAO system requires several different software components, the installation of which on a Linux workstation in the network of the Faculty of Technology is explained in the following. These components include Qt version 4.4 and ICL 4.1 revision 1535 (see Sections **??** and 3.4.1). To install the system on a Linux workstation in the network of the Faculty of Technology, a self build version of the ICL is needed. The installation of the complete system (including the ICL and the ICLProjects) will be described here (see 3.4 for details on the software components). All other used components are installed in the network of the Faculty of Technology by default.

ICL

To install the the system the following lines have to be added to the `.bashrc` file in the home directory:

```
1  export PKG_CONFIG_PATH=/usr/lib/pkgconfig
2  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib
3
4  # qt
5  export QTDIR=/vol/qt/4.4
6  export PATH=$QTDIR/bin:$PATH
7  export PKG_CONFIG_PATH=$QTDIR/lib/pkgconfig:$PKG_CONFIG_PATH
8  export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
9
10 # XCF
11 export XCF_ROOT=/vol/xcf
12 export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:$XCF_ROOT/lib/
       pkgconfig
13 export PATH=$XCF_ROOT/bin:$PATH
14
15 # ICL
16 export NIVISION_ROOT=/vol/nivision
17 export UNICAP_ROOT=/vol/video
18 export ICL_INSTALL_ROOT=/localvol/ICL/build # adapt path
19 export LD_LIBRARY_PATH=$ICL_INSTALL_ROOT/lib:$LD_LIBRARY_PATH
20 export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:$ICL_INSTALL_ROOT/lib
       /pkgconfig
21 export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:$NIVISION_ROOT/lib/
       pkgconfig
22
23 # ICLProjects
24 export ICL_PROJECTS_INSTALL_ROOT=/localvol/ICLProjects/builds
25 export LD_LIBRARY_PATH=$ICL_PROJECTS_INSTALL_ROOT/ICLProjects
       /ICLFiducialTracker/1.0/lib:$LD_LIBRARY_PATH
26 export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:
       $ICL_PROJECTS_INSTALL_ROOT/ICLProjects/pkgconfig
```

After sourcing the new configuration of your Bash, the source code of all components needs to be checked out from their source code repositories, using Subversion, a version control system.

ICL

The ICL is checked out with the following command:

```
1 svn co https://svn.techfak.uni-bielefeld.de/cor-lab/
    projects/icl/trunk /localvol/ICL -r 1535
```

It is configured and build at */localvol/ICL* with

```
1 autoreconf --install --force
2 ./configure --with-IPP-Root=/vol/nivision/IPP/6.0 --
    with-LIBDC-Root=/vol/nivision --with-UNICAP-Root=/
    vol/nivision --with-QT-Root=/vol/qt/4.4 --prefix=/
    localvol/ICL/build
3 make all install
```

After these initial steps the ICL is installed and ready for use.

ICLProjects

Thereafter the ICLProjects can be checked out to install the ICLFiducialTracker (see Section 3.4.1).

```
1 svn co svn+ssh://priamos.techfak.uni-bielefeld.de/vol
    /ni/src/svnroot/vision/ICLProjects/trunk /localvol
    /ICLProjects
```

The ICLFiducialTracker is activated, build and installed by typing the following commands in the ICLProjects directory:

```
1 ./enableProject ICLFiducialTracker
2 make
3 make install
```

Active Objects System

To build the TAO system, copy the source code from the CD to a writable directory and execute the script

```
1 build.sh
```

This script can be found in the root of the source directory (*/code/src/*).
For convenience, it is recommendable to add the newly created binary directory */code/bin/* directory to the PATH environment variable, defined in the *.bashrc* file. All components are installed and ready for use.

The firmware running on the TAOs has been developed using the Arduino Integrated Development Environment (IDE) version 0012 alpha. The code of the firmware can be found in the firmware subdirectory in the code directory on the enclosed CD.

Setting up the System

After building all pieces of software and writing the firmware into a few TAOs, the system can be started by running the XCF infrastructure and the needed TAO modules.
The XCF infrastructure consists of three programs, which are part of the XCF distribution. The first needed program is the *dispatcher*. By simply running this program the first component is set up properly. The dispatcher acts as the naming service, that transparently connects all programs that take part at the XCF communication.
The second program needed is *spread*. This allows to virtually spread the communication over different computers to enable distributed computing. This is quite useful when the number and computational complexity of the modules rise. *spread* needs to be configured by creating a configuration file, that specifies the computers that take part at the communication. The following lines give an example of the structure of this configuration file.

```
1 Spread_Segment 127.0.0.255:4803 {
2   localhost 127.0.0.1
3   leonardo 127.0.1.1
4 }
```

The file specifies two nodes (which both are the same in this example). The first line describes the network segment and port, where the communication will take place and opens the description of the part taking nodes. The second and the third lines describe the nodes with name and Internet Protocol (IP) address. The last line closes the description of nodes and thereby marks the end of the configuration file. Now it is possible to run spread by simply calling *spread* from the command line.
Finally the hart of XCF needs to be started, the ActiveMemory. This allows to transparently subscribe on certain content, not on processes data streams, which makes the communication much more transparent. The ActiveMemory needs a directory to store it's XML database and a Uniform Resource Identifier (URI) root and can be started by calling *memory_server /tmp/dump/ ActiveObjects* from the command line, assuming that the given directory exists.
All components of the infrastructure are running and the needed TAO modules can be started. For instance the modules Marker2XCF, XCF2Serial, PotentialPlanner and an application module would make a usable system. Alternatively the ObjectSimulator can be used instead of Marker2XCF and XCF2Serial, when no real TAOs are available.

API Documentation

The API documentation is included in the */doc/* directory. It can alternatively be build using doxygen, a documentation generator for C++ and other programming languages, which is installed in the network of the Faculty of Technology by default, by calling

```
1 doxygen doxyfile
```

from the src directory. A complete API documentation in Hypertext Markup Language (HTML) format.

List of Figures

List of Abbreviations

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CAD	Computer Aided Design
CI	Computer Interface
CORBA	Common Object Request Broker Architecture
CSV	Comma-Separated Value
DNA	Deoxyribonucleic Acid
DOF	degrees of freedom
DOM	Document Object Model
DWENN	Dynamic Wave Expansion Neural Network
EEG	Electroencephalography
FTP	File Transfer Protocol
GPL	GNU General Public License
GUI	Graphical User Interface
HCI	Human Computer Interaction
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
ICE	Internet Communication Engine
ICL	Image Component Library
IDE	Integrated Development Environment
IP	Internet Protocol

IR Infra-Red

IRDA Infrared Data Association

KDE K Desktop Environment

LED Light Emitting Diode

LGPL GNU Lesser General Public License

MBS Model-based Sonification

MCRpd Model-Control-Representation (physical and digital)

MVC Model-View-Controller

OCI Oracle Call Interface

ODBC Open Database Connectivity

OLED Organic Light Emitting Diode

OSC Open Sound Control

PCB Printed Circuit Board

PIC Programmable Integrated Circuit (Microcontroller)

PDF Portable Document Format

PF Physical Functionality

PMD Planar Manipulator Display

PWC Philips Web Cam

PWM Pulse-Width Modulation

RMI Remote Method Invocation

SNUI Sensor Network User Interface

SAX Simple API for XML

SOM Self-organizing Map

SQL Structured Query Language

SSL Secure Sockets Layer

SVG Scalable Vector Graphics

TAO Tangible Active Object

TCP Transmission Control Protocol

tDesk Tangible Desk

TTL Transistor Transistor Logic

TUI Tangible User Interface

TUIO Tangible User Interface Object

UART Universal Asynchronous Receiver Transmitter

UDP User Datagram Protocol

UI User Interface

UML Unified Modeling Language

URI Uniform Resource Identifier

USB Universal Serial Bus

WMR Wheeled Mobile Robot

XCF Extended Communication Framework

XML Extensible Markup Language

XMLTIO XML Template I/O

Bibliography

[ami08] Ambient Intelligence Group, November 2008. http://www.techfak.uni-bielefeld.de/ags/ami/.

[ard08] Arduino - ArduinoBoardProMini, November 2008. http://www.arduino.cc/en/Main/ArduinoBoardProMini.

[BEHR08] T. Bovermann, C. Elbrechter, T. Hermann, and H. Ritter. Audiodb: Get in touch with sounds. In *Proc. of the Int. Conf. on Auditory Display 2008*, 2008.

[BHR] T. Bovermann, T. Hermann, and H. Ritter. Tangible Data Scanning Sonification Model. In *Proceedings of the 12th International Conference on Auditory Display*, pages 20–23.

[BHR06] T. Bovermann, T. Hermann, and H. Ritter. A tangible environment for ambient data representation. In D. McGookin and S. Brewster, editors, *First International Workshop on Haptic and Audio Interaction Design*, volume 2, pages 26–30, Glasgow, UK, 2006. www.multivis.org.

[BKHR08] T. Bovermann, R. Koiva, T. Hermann, and H. Ritter. TUImod: Modular objects for tangible user interfaces. In *Proceedings of the 2008 Conference on Pervasive Computing*, 2008.

[BKJ05] R. Bencina, M. Kaltenbrunner, and S. Jorda. Improved Topological Fiducial Tracking in the reacTIVision System. In *Computer Vision and Pattern Recognition, 2005 IEEE Computer Society Conference on*, volume 3, 2005.

[clu08] Open source Clustering software, 2008. http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/software.htm.

[CvDHM06] R. Companje, N. van Dijk, H. Hogenbirk, and D. Mast. Globe4D: time-traveling with an interactive four-dimensional globe. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 959–960. ACM New York, NY, USA, 2006.

[DHKR08] J. R. Dawin, D. Hartwig, A. Kudenko, and E. Riedenklau. TRecS: Ein tangibles, rekonfigurierbares system zur explorativen datenanalyse.

Technical report, Bielefeld University, 2008. Betreuer: T. Bovermann and T. Hermann.

[Dou04] P. Dourish. *Where the Action Is : The Foundations of Embodied Interaction (Bradford Books)*. The MIT Press, September 2004.

[eag08] CadSoft Online: EAGLE Layout Editor, 2008. `http://www.cadsoft.de/`.

[ecm08] Standard ECMA-262, 2008. `http://www.ecma-international.org/publications/standards/Ecma-262.htm`.

[far08] Farnell Deutschland, 2008. `http://de.farnell.com/`.

[FIB95] G.W. Fitzmaurice, H. Ishii, and W.A.S. Buxton. Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 442–449. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1995.

[fir08] Arduino - Firmata, 2008. `http://arduino.cc/en/Reference/Firmata`.

[FSMI00] P. Frei, V. Su, B. Mikhak, and H. Ishii. curlybot: Designing a new class of computational toys. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 129–136. ACM New York, NY, USA, 2000.

[hak08] HAKA Elektronik-Leiterplatten GmbH, 2008. `http://www.haka-lp.de/home.html`.

[HBRR07] T. Hermann, T. Bovermann, E. Riedenklau, and H. Ritter. TANGIBLE COMPUTING FOR INTERACTIVE SONIFICATION OF MULTIVARIATE DATA. 2007.

[Her02] T. Hermann. *Sonification for exploratory data analysis*. Bielefeld, 2002.

[HHK+08] Felix Hagemann, Thomas Hermann, Risto Kõiva, Eckard Riedenklau, Patrick Mai, Stefan Wieschebrink, Oliver Lieske, and Till Bovermann. Ambient Lights, December 2008. `http://www.lfsaw.de/hardware/ambient_lights.shtml`.

[HHR04] T. Hermann, T. Henning, and H. Ritter. Gesture desk - an integrated multi-modal gestural workplace for sonification. 2915:369–379, 2004.

[HHR06] T. Hermann, O. Honer, and H. Ritter. Acoumotion-an interactive sonification system for acoustic motion control. *Lecture Notes in Computer Science*, 3881:312, 2006.

[HJBdC06] M. Hauenstein, T. Jenkins, M. Baalman, and A. de Campo. Audio Shaker. *NIME. Paris, France*, 2006.

[HR99] T. Hermann and H. Ritter. Listen to your data: Model-based sonification for data analysis. *Advances in intelligent computing and mulimedia systems*, pages 189–194, 1999.

[icl08] ICL, 2008. `https://trac.cor-lab.uni-bielefeld.de/icl`.

[Igo07] T. Igoe. *Making Things Talk*. O'Reilly, 2007.

[ipp08] Intel IPP - Intel Software Network, 2008. `http://software.intel.com/en-us/intel-ipp/`.

[JKGB05] S. Jorda, M. Kaltenbrunner, G. Geiger, and R. Bencina. The reactable*. In *Proceedings of the International Computer Music Conference (ICMC 2005), Barcelona, Spain*, pages 579–582, 2005.

[Joh87] R.E. Johnson. Model/View/Controller. *Department of CS, University of Illinois, Urbana-Champaign*, 1987.

[kde08] The KDE Project, 2008. `http://kde.org`.

[Kha86] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90, 1986.

[KIMK00] Y. Kitamura, Y. Itoh, T. Masaki, and F. Kishino. ActiveCube: a bi-directional user interface using cubes. In *Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on*, volume 1, 2000.

[kon08] Konqueror - Web Browser, 2008. `http://konqueror.kde.org/features/browser.php`.

[KSN+06] M. Kojima, M. Sugimoto, A. Nakamura, M. Tomita, H. Nii, and M. Inami. Augmented Coliseum: An Augmented Game Environment with Small Vehicles. In *First IEEE International Workshop on Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006*, pages 3–8, 2006.

[Lat93] Jean-Claude Latombe. *Robot motion planning*. The Kluwer international series in engineering and computer s cience ; 124. Kluwer Acad. Publ., Boston [u.a.], 3. print. edition, 1993.

[leg08] LEGO.com The Official Web Site of LEGO products!, 2008. http://www.lego.com.

[lib08] LibSerial, 2008. http://libserial.sourceforge.net/mediawiki/index.php/Main_Page.

[Loc06] Philipp Locher. Tangible User Interfaces - Classification. 2006.

[LSR03] DV Lebedev, JJ Steil, and H. Ritter. Real-time path planning in dynamic environments: a comparison of three neural network models. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 4, 2003.

[McN00] Timothy Scott McNerney. Tangible Programming Bricks: An approach to making programming accessible to everyone, February 2000.

[mes08] Arduino playground - Messenger, 2008. http://www.arduino.cc/playground/Code/Messenger.

[MKM07] D. Merrill, J. Kalanithi, and P. Maes. Siftables: towards sensor network user interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 75–78. ACM New York, NY, USA, 2007.

[NDNG03] H. Newton-Dunn, H. Nakano, and J. Gibson. Block jam: a tangible interface for interactive music. In *Proceedings of the 2003 conference on New interfaces for musical expression*, pages 170–177. National University of Singapore, Sony Corporation, 2003.

[Phi04] R. Philippsen. *Motion Planning and Obstacle Avoidance for Mobile Robots in Highly Cluttered Dynamic Environments*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2004.

[PI07] J. Patten and H. Ishii. Mechanical constraints as computational constraints in tabletop tangible interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 809–818. ACM Press New York, NY, USA, 2007.

[pic08] pico, November 2008. http://poor-robot.com/pico/.

[PMAI02] G. Pangaro, D. Maynes-Aminzade, and H. Ishii. The actuated workbench: computer-controlled actuation in tabletop tangible interfaces. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 181–190. ACM New York, NY, USA, 2002.

[pre08] Precision Miniature Motors, Gearmotors & DC motor control by Precision Microdrives, 2008. http://www.precisionmicrodrives.com/product_info.php?products_id=108.

[PRI02] J. Patten, B. Recht, and H. Ishii. Audiopad: A tag-based interface for musical performance. In *Proceedings of the 2002 conference on New interfaces for musical expression*, pages 1–6. National University of Singapore Singapore, Singapore, 2002.

[pro08] Processing 1.0, 2008. http://www.processing.org/.

[rea08] racTIVision, 2008.

[rei08] reichelt elektronik, 2008. http://www.reichelt.de.

[rob08] ROBOTmaker - Hobby Robots and Robotics, 2008. http://www.robotmaker.co.uk/.

[Roy08] Michael Roybal. One Inch Robot, November 2008. http://www.inklesspress.com/robots_4.htm.

[RZSP04] D. Rosenfeld, M. Zawadzki, J. Sudol, and K. Perlin. Physical Objects as Bidirectional User Interface Elements. *IEEE COMPUTER GRAPHICS AND APPLICATIONS*, pages 44–49, 2004.

[sc08] SuperCollider, 2008. http://supercollider.sourceforge.net/.

[sol08] Solid Edge: Velocity Series: Products and Solutions: Product Lifecycle Management (PLM): Siemens PLM Software, 2008. http://www.plm.automation.siemens.com/en_us/products/velocity/solidedge/index.shtml.

[spa08] Sparkfun Electronics, 2008. http://www.sparkfun.com.

[SV08] B. Schiettecatte and J. Vanderdonckt. AudioCubes: a distributed cube tangible interface based on interaction range for sound design. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 3–10. ACM New York, NY, USA, 2008.

[swa08] swarmrobot.org, November 2008. http://www.swarmrobot.org/.

[tx-08] TX-IRHS High-Speed Serial Infrared Communication & Remote Control IC, 2008. http://www.rentron.com/remote_control/TX-IRHS.htm.

[Tü09] René Tünnermann. Modular Multi-Touch Interface for Direct Closed-Loop Interactions, January 2009. Diplomarbeit; Bielefeld University, Faculty of Technology, Ambient Intelligence Group; supervised by Dr. Thomas Hermann and Prof. Helge Ritter.

[UI99] J. Underkoffler and H. Ishii. Urp: A luminous-tangible workbench for urban planning and design. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, pages 386–393. ACM New York, NY, USA, 1999.

[UI00] B. Ullmer and H. Ishii. Emerging frameworks for tangible user interfaces. *IBM systems journal*, 39(3):915–931, 2000.

[UIG98] B. Ullmer, H. Ishii, and D. Glas. mediaBlocks: physical containers, transports, and controls for online media. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 379–386, 1998.

[vdHE04] E. van den Hoven and B. Eggen. Tangible Computing in Everyday Life: Extending Current Frameworks for Tangible User Interfaces with Personal Objects. *Lecture notes in computer science*, pages 230–242, 2004.

[web08] The WebKit Open Source Project, 2008. `http://webkit.org/`.

[wes08] Batteries (WES-Technik), 2008. `http://www.wes-technik.de/English/Battery.htm`.

[xbe08] XBee & XBee-PRO 802.15.4 OEM RF Modules - Digi International, 2008. `http://www.digi.com/products/wireless/point-multipoint/xbee-series1-module.jsp`.

[xcf08] The XCF Integration Toolkit - Trac, November 2008. `https://code.ai.techfak.uni-bielefeld.de/trac/xcf`.

[xer08] Verces-C++ XML Parser, 2008. `http://xerces.apache.org/xerces-c/`.

[zig08] ZigBee Aliance, 2008. `http://www.zigbee.org`.

Acknowledgments

I have to thank Dr. Thomas Hermann and Prof. Helge Ritter for giving me the possibility to do this work. Additionally I thank Tobias Großhauser for his great support as another supervisor. Furthermore I want to thank the members of the Ambient Intelligence Group and the members of the Neuroinformatics Group, namely Jan Anlauff, Till Bovermann, Christof Elbrechter, Jessica Hummel, Florian Schmidt and René Tünnermann for always lending me an ear when I had problems. Last but not least I have to thank my family and Udo Schröter for the great endurance, support and encouragement.

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where the acknowledgment has been made in the text.

Bielefeld, June 2, 2009

Eckard Riedenklau