UvA Rescue
Team Description Paper
Virtual Robot competition
Rescue Simulation League
RoboCup 2012

Arnoud Visser, Nick Dijkshoorn, Sander van Noort, Olaf Zwennes,
Maarten de Waard, Sammie Katt, and Richard Rozeboom

Universiteit van Amsterdam, Science Park 904, 1098 XH Amsterdam, NL
http://www.jointrescueforces.eu

Abstract. This year's contribution of the UvA Rescue Team is twofold. On the one hand a contribution is made to infrastructure of the Virtual Robot competition [1]. On the other hand progress is made with visual mapping with a flying platform. The progress was not only shown on the RoboCup, but in addition also at the International Micro Air Vehicle competition [2]. Further improvements will be related with the use of radar for navigation, improved autonomous behaviour and the modeling of mobile victims.

Introduction

The RoboCup Rescue competitions provide benchmarks for evaluating robot platforms' usability in disaster mitigation. Research groups should demonstrate their ability to deploy a team of robots that explore a devastated area and locate victims. The Virtual Robots competition, part of the Rescue Simulation League, is a platform to experiment with multi-robot algorithms for robot systems with advanced sensory and mobility capabilities.

The shared interest in the application of machine learning techniques to multi-robot settings has led to a joint effort between the laboratories of the Oxford and Amsterdam Universities. The result of this four year collaboration has boiled down in many shared publications [3–7] and a thesis [8] from Oxford's Exeter College. This year only the Universiteit van Amsterdam will go the competition, although they are still open for further collaborations. This year's challenge will be to make a more agile user interface and to make the world modeling truly 3 dimensional [9].

To be able to efficiently coordinate a team of robots in a disaster situation many state-of-the-art robotic techniques have to be integrated. Our approach is extensively described in previous Team Description Papers (see [10] and the references within). In this paper we will concentrate on this year's innovations.

1 Team Members

UsarCommander was originally developed by Bayu Slamet and all other contributions have been integrated into this framework. Many other team members have contributed to perception and control algorithms inside this framework.

The following contributions have been made and will be made this year:

Arnoud Visser	: autonomous exploration [4–6]
Nick Dijkshoorn	: Visual SLAM [1, 2, 11, 12]
Sander van Noort	: Nao model [13–15]
Olaf Zwennes	: automatic map generation [16, 17]
Maarten de Waard	: XABSL based behaviors
Sammie Katt	: modeling mobile victim
Richard Rozeboom	: radar based navigation

The latter three studies are in the initial phase, so the results will be discussed in another overview paper.

2 Visual Localization And Mapping

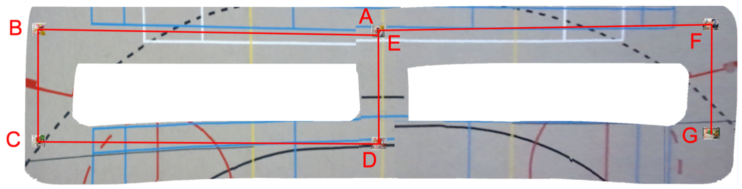
One of the most fundamental problems in robotics is the Simultaneous Localization And Mapping problem (SLAM). In SLAM, the robot acquires a map of its environment while simultaneously localizing itself relative to the map. This knowledge is critical for robots to operate autonomously. For small (flying) vehicles, such as the affordable AR.Drone quadrotor, researchers focused on solution based on vision sensors. Vision seems to offer a good balance in terms of weight, accuracy and power consumption.

For a human operator a texture map is essential for navigation. This map can be built once the local perspective transformation is estimated [11]. For a robot a feature map is easier interpretable. If the AR.Drone is able to relate a video frame to a position inside the feature map, the vehicle is able to correct the drift in its internal inertia sensors long enough to build a map as large as $50m^2$.

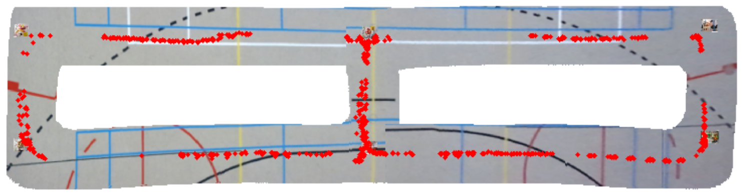
From each camera frame we extract Speeded-Up Robust Features (SURF) [18] that are invariant with respect to rotation and scale. Those features are stored in a 2D grid with a fixed resolution of $100 \times 100mm$ per cell. Each feature is an abstract description of an "interesting" part of an image (e.g., corners). A feature is described by a center point in image sub-pixel coordinates and a descriptor vector that consists of 64 floats.

Each feature that is detected in a camera frame is mapped to the corresponding cell of the feature map. This is done by casting a ray from the features pixel coordinates in the frame. For each cell, only the best feature (e.g. with the highest response) is kept and the other features are dropped.

The feature map can be used for absolute position estimates, at the moment of loop-closure (when the AR.Drone is above a location where it has been before). This allows to correct the drift that originates from the internal sensors of the AR.Drone. When a camera frame is received, SURF features are extracted.



(a) Visual map created by the mapping method



(b) Localization based on visual map

Fig. 1. Localization and mapping based on the images of the bottom camera of the simulated AR.Drone, flying at approximately 85cm altitude.

A feature's center point is transformed to its corresponding position in 2D world coordinates. The next step is matching the feature descriptors from the camera frame against the feature descriptors from the feature map. Matching is done using a brute force matcher that uses the least-squares norm as similarity measure. Each match corresponds with a transformation of the robot through 3 dimensions. The combination of a number of reliable matches (see [11] for more details) can be used to estimate an absolute location with respect to the visual map.

Figure 1(b) indicates the positions where the AR.Drone was able to localize itself. Localization was performed at almost every position of the 8-shape trajectory. At some positions localization was not possible, because the camera observed insufficient image features. For example, localization failed when only lines without intersections are observed. These results confirm that the localization method is able to use an autonomously build map for navigation purposes based on visual clues.

3 Infrastructure Contribution

The Universiteit van Amsterdam has contributed on several aspects of the competition environment:

3.1 Image Server

The task of the Image Server is to send camera frames of a simulated robot to the robot controller. Within USARSim this was previously implemented as a separate tool, because it was not possible in Unreal Script to directly access the image buffer of a simulated camera. Due this limitation it was not possible to directly send the image data from unreal script to the robot controller. To come around this issue the function call of DirectX that draws the frame to the screen is hooked. *Hooking* is a term that refers to altering the behavior of software, which includes intercepting function calls. In the hooked function the specific parts of the backbuffer containing the camera frame are copied and then finally send to the robot controller. The robot controller can then use the image for processing.

Previously the image server UPIS was a standalone program. It used the Microsoft Detours library to intercept the Direct3D calls that Unreal uses to finalize its screen drawing. It then copies the display to an external buffer and serves the image upon request to any clients. UPIS can be either compiled directly from source, or installed using precompiled 32-bit binaries available for Windows XP and Vista. Due to restrictions on Microsoft Detours, the standalone version can only run in 32-bit mode.

To circumvent this restriction, the UvA Rescue Team incorporated the image server inside the main code of USARSim in UDK. The usage of the image server is switchable by the option bEnableImageServer in the config files. The server only hooks and captures frames when a client is connected (so there should be no overhead for having it always running). This is possible because UDK itself includes the EasyHook library, which intercepts Direct3D calls (which works for both 32 bits and 64 bits machines). The ImageServer is automatically started when an USARBotAPI.BotDeathMatch is started.

3.2 PhysX proxy

In the new design of USARSim (UDK) wheeled, legged and aerial robots are modeled using *parts* and *joints*. The robot models are defined by adding the specific part and joints as a number of objects in the default properties of their class in Unreal Script. The robot models are not defined in the configuration files because this would require a complete redefinition of the configuration file format.

Parts are always modeled by a *PhysicalItem*. These are static meshes with physics and represent a rigid body. These parts are connected by joints, which are modeled by different types (i.e. RevoluteJoint, PrismaticJoint, WheelJoint). For legged and wheeled robots the most important joint type is the *RevoluteJoint*.

A specific issue, that arises when simulating a complex legged robot such as the Nao, is the presence of *complex joints*. Basically these type of joint constructions occur when multiple joints are located in almost the same location in the robot. An example of such a joint is the Nao shoulder joint with two degrees of freedom. In reality such joints consist of two revolute joints with a small intermediate limb (rigid body).

To overcome this issue a few simplifications are made to the model by disabling contact generation between parts that are located close to each other. This allows the collisions between the parts to be ignored. Additionally the masses and inertia tensors are manually determined to ensure improved behavior.

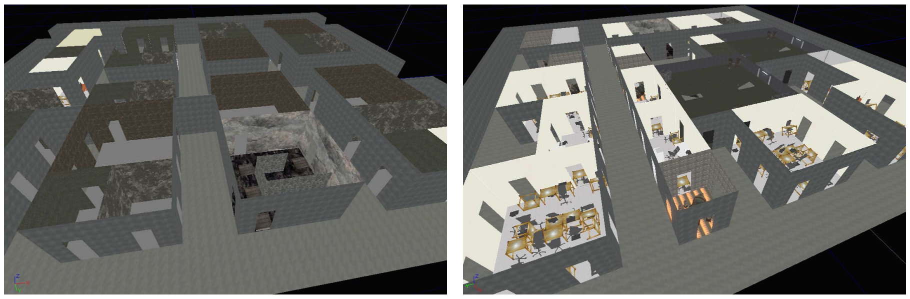
This solution introduced another problem because Unreal Script only allows to disable contact generation between rigid bodies that are connected through a joint. To overcome this problem a *PhysX proxy DLL* was introduced, which directly uses the PhysX API to modify these physics properties of the rigid bodies. Unreal Script then communicates with PhysX using *DLLBind*[1], which like the name implies allows binding C functions to Unreal Script code. DLLBind only allows calling C functions from Unreal Script.

Inside a proxy class you can make all functions from the PhysX API accessible. The functions which are made available in `USARUtility:PhysXProxy.uc` are for instance SetIterationSolverCount, SetActorPairIgnore and SetMassSpaceInertiaTensor. These functions are used in `USARBot:USARVehicle.uc` and `USARBot:Nao.uc`.

3.3 Automatic map generator

One of the contributions this year is the extension of the automatic map generator currently available in USARSim. The map generator is extended in such a way that the difficulty of the environment can be gradually increased. Difficulty can be expressed along several aspects, such as indicated in the *a priori* information of previous competitions (mobility, communication, victims). This time the focus will be on another aspect, the difficulty to map the environment.

Based on a literature study, several aspects which influence the mapping difficulty are identified and translated in a number of map-generation rules. With those rules multiple maps with various difficulties are generated (see Fig. 2). The difficulty of those maps was evaluated by several experienced robot operators. The conclusion could be made that there was a strong correlation between the intended and perceived difficulty [16, 17].



(a) map generated with a low difficulty ($d = 1$).

(b) map generated with a high difficulty ($d = 9$).

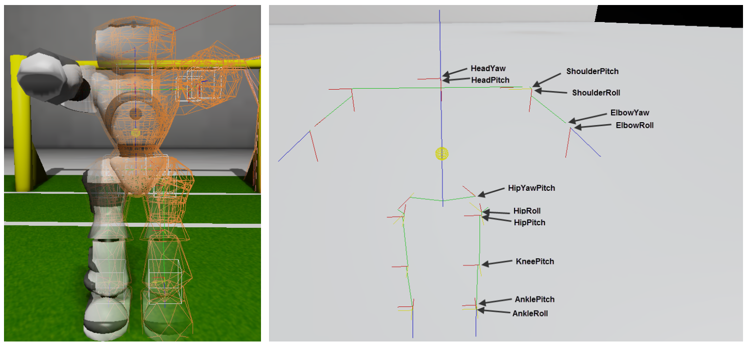
Fig. 2. An eagle-eye view of generated maps with different difficulty settings.

[1] http://udn.epicgames.com/Three/DLLBind.html

This research introduced the concept of adaptive map generation, to generate functional indoor environments within a robot simulation environment. Future work could build upon this concept and expand it to encompass different functional needs or refine the concept to to higher resolutions.

3.4 Nao humanoid robot

The model of the humanoid robot Nao is completed [14]. The model consists of 21 links between joints, including collision frames and mass distribution (see Fig. 3). The dynamics of the robot is extensively tested. First a number of basic experiments are done, The initial test was to determine the right gravity and force constants. Those experiments were continued with testing the settings of the constraints between joints. This was done with a chain of four rigid bodies and four joints. The motor spring variable of all joints is set to a very high value, to ensure the joints should be able to satisfy their constraints. Based on these results of this test a physics' `TimeStep` of $1/200s$ was chosen, combined with a solver iteration count of 32.



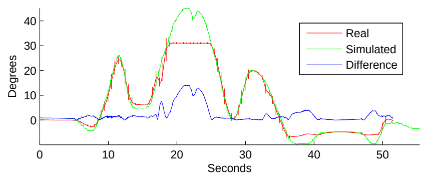
(a) Collision frame (b) Denavit Hartenberg chains

Fig. 3. Details of the Aldebaran Nao robot model.

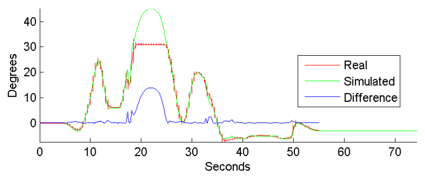
The basic experiments were followed by a number of more advanced experiments. Those experiments were performed with both the simulated and real Nao. Those experiments were a single step, a kick and the Tai Chi balance act which is used by the manufacturer Aldebaran as diagnostic behavior. The later experiment was repeated with the commercial simulator NaoSim, provided by Cogmation Robotics[2]. The differences between real and simulated robot are quite minimal (indicated in blue in Fig. 4). The only exception is for the larger angles; both simulation models did not include the restrictions of the collision hull of this particular joint in our model (which is a function of the ankle pitch[3]).

[2] http://www.cogmation.com/naosim.html

[3] http://users.aldebaran-robotics.com/docs/site_en/reddoc/hardware/joints-names_3.3.html



(a) USARSim versus real robot



(b) NaoSim versus real robot

Fig. 4. The validation of the Aldebaran Nao robot model; the ankle roll during Tai Chi balance act

The final test was a full scale application: the simulation of a Soccer team consisting of 4 Nao robots, which were running the code of the Dutch Nao Team [19]. The behavior of the simulated Nao robots running the Dutch Nao Team code is quite convincing. When a Nao falls the robot is able to stand up correctly. Using their cameras they are able to track the ball (as long as the right ball color is chosen). The main problem encountered is when the Nao moves close to the ball to perform a kick motion; it does so by taking tiny steps. The tiny steps of the simulated robot are too small, what results that it takes too long to correctly position in front of the ball, longer than it would take the real Nao to position correctly.

The development of this model for a humanoid robot will allow the development of many other walking and humanoid robots (based on the same principles) [14].

4 Conclusion

This paper summarizes improvements in the algorithms of the UvA Rescue Team since RoboCup 2011 in Istanbul. Many developments are not only valuable inside the Rescue Simulation League, but also valuable for the Soccer Simulation, the RoboCup@Home and the Standard Platform League [15]. For the Virtual Robot competition, developments in the user interface and full 3D mapping are important. The UvA Rescue team has participated this year successfully in both the Iran and Dutch Open competition, where respectively the award for the best scientific presentation and the 1st prize were won[4].

References

1. Dijkshoorn, N., Visser, A.: Urban Search and with micro aerial vehicles. In: Proc. CD of the 16th RoboCup International Symposium. (2012)
2. Visser, A., Dijkshoorn, N., van der Veen, M., Jurriaans, R.: Closing the gap between simulation and reality in the sensor and motion models of an autonomous

[4] See for an overview http://www.jointrescueforces.eu/wiki/tiki-index.php?page=Achievements

ar.drone. In: Proceedings of the International Micro Air Vehicle Conference and Flight Competition (IMAV11). (2011)

3. de Hoog, J., Cameron, S., Visser, A.: Role-based autonomous multi-robot exploration. In: Proceedings of the International Conference on Advanced Cognitive Technologies and Applications (Cognitive 2009). (2009)
4. de Hoog, J., Cameron, S., Visser, A.: Autonomous multi-robot exploration in communication-limited environments. In: Proceedings of the 11th Conference Towards Autonomous Robotic Systems (Taros 2010). (2010)
5. de Hoog, J., Cameron, S., Visser, A.: Dynamic team hierarchies in communication-limited multi-robot exploration. In: Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR 2010). (2010)
6. de Hoog, J., Cameron, S., Visser, A.: Selection of rendezvous points for multi-robot exploration in dynamic environments. In: International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS). (2010)
7. Wellman, B.L., de Hoog, J., Dawson, S., Anderson, M.: Using rendezvous to overcome communication limitations in multirobot exploration. In: Proceedings of SMC (IEEE International Conference on Systems, Man and Cybernetics). (2011)
8. de Hoog, J.: Role-Based Multi-Robot Exploration. PhD thesis, University of Oxford (2011)
9. Nelson, P.: 3d mapping for robotic search and rescue. 4th year project report (2011)
10. Dijkshoorn, N., Flynn, H., Formsma, O., van Noort, S., van Weelden, C., Bastiaan, C., Out, N., Zwennes, O., Otárola, S.S., de Hoog, J., Cameron, S., Visser, A.: Amsterdam Oxford Joint Rescue Forces - Team Description Paper - RoboCup 2011. In: Proc. CD of the 15th RoboCup International Symposium. (2011)
11. Dijkshoorn, N., Visser, A.: Integrating sensor and motion models to localize an autonomous ar.drone. International Journal of Micro Air Vehicles **3** (2011) 183–200
12. Dijkshoorn, N.: Simultaneous localization and mapping with the ar.drone. Master's thesis, Universiteit van Amsterdam (2012)
13. van Noort, S., Visser, A.: Validation of the dynamics of an humanoid robot in usarsim. In: Proceedings of Performance Metrics for Intelligent Systems Workshop (PerMIS12). (2012)
14. van Noort, S.: Validation of the dynamics of an humanoid robot in usarsim. Master's thesis, Universiteit van Amsterdam (2012)
15. van Noort, S., Visser, A.: Extending virtual robots towards robocup soccer simulation and @home. In: Proceedings of the 16th RoboCup Symposium. (2012) To be published in the Springer Lecture Notes on Artificial Intelligence series.
16. Zwennes, O.: Adaptive indoor map generator for usarsim. Bachelor's thesis, Universiteit van Amsterdam (2011)
17. Zwennes, O., Weiss, A., Visser, A.: Adapting the mapping difficulty for the automatic generation of rescue challenges. In: RoboCup IranOpen 2012 Symposium (RIOS12). (2012)
18. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-up robust features (surf). Computer Vision and Image Understanding **110** (2008) 346 – 359
19. Verschoor, C., Wiggers, A., ten Velthuis, D., Keune, A., Cabot, M., Nugteren, S., van Egmond, E., van der Molen, H., Iepsma, R., van Bellen, M., de Groot, M., Fodor, E., Rozeboom, R., Visser, A.: Dutch nao team - code release 2011 and technical report 2011. Published online, Universiteit van Amsterdam (2011)