The 3rd AAU Workshop on Robotics

Proceedings

Tan, Zheng-Hua; Bai, Shaoping; Bak, Thomas; Rehm, Matthias; Jochum, Elizabeth Ann

Publication date:
2015

Document Version
Publisher's PDF, also known as Version of record

AALBORG UNIVERSITY
DENMARK

The 3rd AAU Workshop on Robotics

October 30, 2014, Aalborg University, Denmark

Proceedings

Edited by
Zheng-Hua Tan, Shaoping Bai, Thomas Bak, Matthias Rehm and Elizabeth Ann Jochum

AALBORG UNIVERSITY PRESS

The 3rd AAU Workshop on Robotics

Edited by Zheng-Hua Tan, Shaoping Bai, Thomas Bak, Matthias Rehm and Elizabeth Ann Jochum

Cover pictures from left:

- Little helper, www.en.m-tech.aau.dk/Research+Groups/Robotics+and+Automation
- Light weight robotic arm: http://homes.m-tech.aau.dk/shb/
- iSocioBot, http://socialrobot.dk
- Prosthetic hand illustration, photo by Dalila Lepirica and Shellie Boudreau
- Henrik & Henrik, http://www.hum.aau.dk/~scharfe/

ISBN: 978-87-7112-433-0

Published by:
Aalborg University Press Skjernvej 4A, 2nd floor
DK – 9220 Aalborg
Denmark
Phone: (+45) 99 40 71 40 aauf@forlag.aau.dk http://forlag.aau.dk/forside.aspx

The 3rd AAU Workshop on Robotics

Research and application development activities in robotics are rapidly growing at Aalborg University (AAU). This is witnessed by high-impact publications and robotic systems such as Geminoid.DK, LittleHelper, AAUBot, iSocioBot, to name a few. We anticipate that robotics, and especially robotics concerned with collaborative interaction between humans and robots to achieve common goals, will become a central area of research at AAU.

The 3rd AAU Workshop on Robotics was held on October 30, 2014 at Aalborg University, Aalborg, Denmark. The workshop was very successful with two invited talks and ten oral presentations and with a large audience from both academy and industry.

This was the third workshop in the series, following the success of the first two occasions: AAURob2012 and AAURob2013. It aimed at providing a platform for researchers in robotics, including professors, PhD and Master students, to exchange their ideas and results. Our goals were to foster close interaction among researchers from multiple relevant disciplines in the field of robotics, and consequently, to promote collaboration across departments of all faculties towards making AAU's robotics program – Aalborg U Robotics – a successful story in robotics research.

After the workshop, full papers were submitted on the basis of the abstract submissions to the workshop. Each full paper submission was peer-reviewed by at least two reviewers external to AAU. Finally six accepted papers are included in the proceedings. The external reviewers include

- Hong Kook Kim, Gwangju Institute of Science and Technology, Gwangju, Republic of Korea
- Zhanyu Ma, Beijing University of Posts and Telecommunications (BUPT), China
- Trung Dung Ngo, University of Brunei Darussalam, Brunei
- Yongsheng Ou, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
- Søren T. Hansen, Danish Ministry of Foreign Affairs, Denmark
- Jose C. Rocha, Universidade Federal do Rio de Janeiro, Brazil
- Thiusius Rajeeth Savarimuthu, Mærsk Mc-Kinney Møller Instituttet, University of Southern Denmark, Denmark
- Leon Bodenhagen, Mærsk Mc-Kinney Møller Instituttet, University of Southern Denmark, Denmark

We warmly thank these reviewers for their precious time in reviewing the papers and serving on the technical committee. We appreciate the effort and contribution from the keynote speakers, the authors, our colleagues reviewed the abstract submissions, and our colleagues supported the workshop in various ways, particularly, Jane Ehrenskjold Tymm-Andersen. Finally AAU U Robotics financed the event, for which we are grateful.

Zheng-Hua Tan, Shaoping Bai, Thomas Bak, Matthias Rehm and Elizabeth Ann Jochum
Aalborg, Denmark, 2015.

Program:

9.00-9.15 Welcome

9.15-10.15 Keynote: Prof. Dr. Tim Lüth, Technical University of Munich, Germany

10.15-10.45 Coffee Break

10.45-12.05 (Oral Session I)
• Guanglei Wu, Shaoping Bai and Preben Hjørnet: Development of a Fast Pick-and-Place Parallel Robot
• Nick Østergaard, Jeppe Dam and Jesper A. Larsen: Formation Control of AAUSHIP
• Casper Schou, Jens Skov Damgaard and Ole Madsen: Robotic Skill for Object Classification
• Karl Damkjær Hansen, Simon Jensen, Christoffer Sloth and Rafael Wisniewski: Instrumentation of the da Vinci Robotic Surgical System

12.05-13.15 Lunch: Novi Canteen

13.15-14.00 Guest lecture: Dr. John Erland Østergaard, Blue Ocean Robotics, Denmark

14.00-15.00 (Oral Session II)
• Nicolai Bæk Thomsen, Zheng-Hua Tan, Børge Lindberg and Søren Holdt Jensen: Learning Direction of Attention for a Social Robot in Noisy Environments
• Bent Thomsen, Kasper Søe Luckow, Thomas Bøgholm, Lone Leth Thomsen and Stephan Erbs Korsholm: Safety Critical Java for Robotics Programming
• Martins Paberzs, Søren Grynnerup, Stanislaw Bobela, Athanasios Polydoros and Lazaros Nalpantidis: Experiences from the development of an automated production cell using 3D printing and robot assembly

15.00-15.20 Coffee Break

15.20-16.20 (Oral Session III)
• Martin Kibsgaard and Martin Kraus: Real-Time Augmented Reality for Robotic-Assisted Surgery
• Elizabeth Jochum and Lance Putnam: Robotic Art: Interactive Art and Robotics Education Program in the Humanities
• Jens Dinesen Strandbech: Toward using humanoids to alleviate dementia symptoms

16.20-16.30 Goodbye

List of papers

1. Learning Direction of Attention for a Social Robot in Noisy Environments
2. Instrumentation of the da Vinci Robotic Surgical System
3. Real-Time Augmented Reality for Robotic-Assisted Surgery
4. Formation Control of AAUSHIP
5. Robots and Art: Interactive Art and Robotics Education Program in the Humanties
6. Safety Critical Java for Robotics Programming

THIS PAGE IS INTENTIONALLY LEFT BLANK.

Learning Direction of Attention for a Social Robot in Noisy Environments

Nicolai Bæk Thomsen, Zheng-Hua Tan, Børge Lindberg and Søren Holdt Jensen
Department of Electronic Systems, Aalborg University
Aalborg Ø, Denmark
Email: {nit,zt,bli,shj}@es.aau.dk

***Abstract*—It is essential for social robots to be able to locate and direct attention towards communicating persons, however the operating environments can be challenging. When using sound source localization (SSL) acoustic noise sources can distract the robot, which interrupts the desired interaction with people. Since the noise sources can be of many different kinds, it is important for the robot to adapt to any environment. In this paper we present a simple strategy for a robot to adapt to the environment using feedback from a face detection routine, thus eventually only directing attention towards humans. Four experiments with different noise types and different strategies for using feedback from face detection show the effectiveness of the proposed strategy.**

I. Introduction

The field of social robots has gained a lot of attention from both industry and academia in recent years. Social robots are to interact closely with humans in private homes, nursing homes etc. In order to interact well with the sorrounding persons it is fundamental that the robot can localize desired sources (humans) and ignore undesired sources (radio, tv, door slamming etc.), and based on this direct attention towards only humans. It is common for a robot to be equiped with multiple microphones, a camera and possibly a laser range finder, but here we consider only the two first. Most methods are based on using SSL as a first step in the localization process, since this can potentially span 360° as opposed to a camera which is limited by field-of-view. In [1] the authors propose a robot system which is able to localize and track a sound human speaker and then turn towards it in real-time, however noise and speech-like noise sources are not considered. In [2] the term *audio proto object* is introduced, which is basically a segmented audio signal and features derived from this. Based on this, the audio is classified as speech or non-speech and SSL can be performed to locate the source and direct attention. The drawback of this is that it requires offline training to find a classifier and that it cannot adapt to the environment while operating. Another approached is proposed in [3], where a hierarchical Gaussian Mixture Model (h-GMM) is trained offline to classify audio segments as voice, music, environmental or silence. This is tested offline and shows good performance, however it requires offline training. Furthermore, a Generalized Eigenvalue Decomposition version of MUltiple SIgnal Classification (GEVD-MUSIC) is used in order to achieve selective listening on a frame-basis, such that only the direction of speech sources are found. The selective listening combined with audio classification is tested using white noise and speech played simultaneously from two different directions, and it shows good ability estimate the direction of both sources and to classify noise correctly, but speech is only classified correctly in 40.6% of the frames. A decision on when to turn to a speaker is also not considered. An audio-visual tracking system for a robot torso with the ability to turn towards a speaker is proposed in [4]. The system is able to track and turn towards speakers, and also use face detection as feedback to judge whether a sound came from a person or a noise source. In [5] the authors proposed a voice activity detection (VAD) approach, where a running SSL histogram was computed using only the frames marked as speech by the VAD. If a bin/direction exceeded a predefined threshold the robot would direct attention towards this direction. The method was successful for rejecting types of interfering/environmental sounds. One weakness of using these methods for this application is that they will make errors (false alarms or misdetections) when introduced to some environmental sounds and persistently turn towards the direction of the source. Another weakness is when speech is coming from non-communicating sources, e.g., when a radio or TV is turned on. In this case, a robot should not turn toward these since these are merely broadcasting and not trying to communicate with anyone.

In this work we propose a method which is able to consider the case where the robot makes a wrong decision and directs attention to a non-communicative source. This is done by posing the decision as a Bayesian decision problem where we use the conditional density of the hypothesis given the direction from SSL. After having made a decision and turned toward a person we use face detection to evaluate whether the decision was good or bad. Based on this we update the density to incooporate this newly acquired knowledge and the process starts over.

The outline of the paper is as follows: in Sect. II the proposed system is described, Sect. III presents some results obtained from testing the system, and finally in Sect. IV we conclude on the work and discuss directions for further research.

II. Proposed Method

Figure 1 shows the overall structure of the proposed system. When the robot detects an audio segment (e.g., using VAD) it must decide whether to turn towards the sound source or not. In this system this is done by extracting observations from the audio segment and then making a decision based on

This work is supported by the Danish Council for Independent Research - Technology and Production Sciences under grant number: 1335-00162 (iSocioBot)

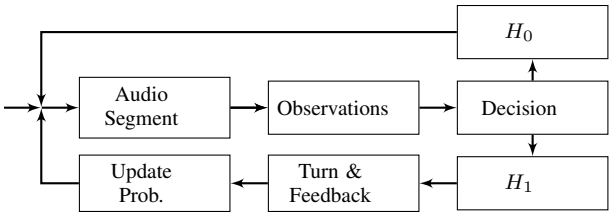


Fig. 1. Block diagram of proposed system.

these. In this system we use the VAD proposed in [6], with the requirements that a speech segment must be longer than 1s and maximum 2s in order to react fast enough. If it is decided not to turn, then no further processing is done and the system waits for the next audio event. If however it is decided to turn the robot turns and uses feedback to evaluate if this was a good decision or not. This information is then used as input in the next decision process.

A. Observations: Harmonicity and SSL

This section describes the observation to base the decision on.

1) Harmonicity: When an audio segment is detected by the front-end VAD, we divide the segment into frames and extract a feature closely related to harmonicity for each frame. It is given by

$$h(t) = \max_{T_1 \leq k \leq T_2} \frac{r_{xx}(t,k)}{r_{xx}(t,0)} \tag{1}$$

where $r_{xx}(t,k)$ is the auto-correlation sequence of the input signal at time frame t and at lag k and is computed using the method in [7] and T_1 and T_2 marks the interval of sample delays corresponding the desired frequency interval in which pitch is supposed to be. The average estimate for the segment is then obtained by

$$X = \frac{1}{T}\sum_{t=1}^{T} h(t) \tag{2}$$

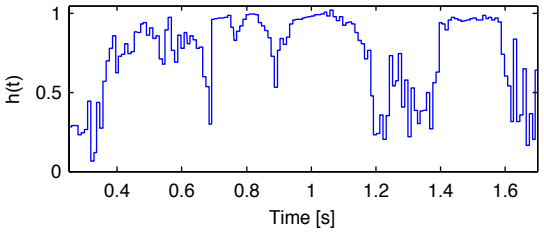
where T is the total number of frames in the segment. A frame size of 512 samples and an overlap of 70% is used based on preliminary testing, which yield a total number of frames between 101 and 205. This is believed to be enough for a robust estimate of X. It has previously been used as feature for Speech Activity Detection [8] where it shows good discriminative power.

Figure 2(a) and 2(c) show a spectrogram of a short speech segment without noise and with noise (SNR=20dB) and reverberation (T_{60}=0.3s), respectively along with the computed harmonicity in figure 2(b) and 2(d). The reverberant speech was generated using the toolbox described in [9]. It is in both cases seen that the feature attains higher value when speech is present than when it is not, allthough it is more significant in the case of clean speech.

2) Sound Source Localization: The SSL estimate is likewise calculated on each frame using the well-known SRP-PHAT method [10] given by

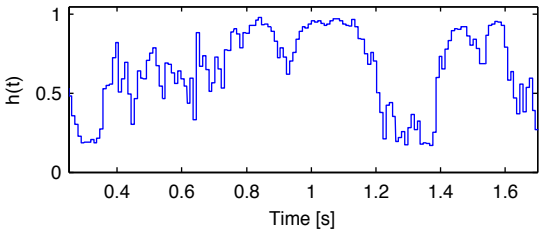
$$\left(\hat{r}_k, \hat{\phi}_k, \hat{\theta}_k\right) = \arg\max_{r,\phi,\theta} \sum_{i=1}^{M}\sum_{j=i}^{M} R^{\text{PHAT}}_{x_i,x_j}\left(\tau^{i,j}_{r,\phi,\theta}, k\right) \tag{3}$$

(a) Spectrogram of a speech segment. Note that y-axis is limited to the frequency range containing pitch.



(b) Speech feature computed using (1) in frames on same speech segment as spectrogram.

(c) Spectrogram of a speech segment. Note that y-axis is limited to the frequency range containing pitch.



(d) Speech feature computed using (1) in frames on same speech segment as spectrogram.

Fig. 2. Computed speech feature and the corresponding spectrogram for a given speech segment.

where M is the number of microphones, r, ϕ and θ are the range/distance, elevation and azimuth, respectively, $\tau^{i,j}_{r,\phi,\theta}$ is the relative time-delay between microphone i and j if the source is located at (r,ϕ,θ), $R^{\text{PHAT}}_{x_i,x_j}(\tau)$ is the PHAT weighted cross-correlation and k indicates the frame number within the entire segment. The total number of frames for SSL in a detected segment can be between $\lceil 1 \cdot 16000/512 \rceil = 32$ and $\lceil 2 \cdot 16000/512 \rceil = 63$, assuming a frame size of 512 samples and no overlap, since we have restricted the segment duration to be between 1s and 2s.

Since we only use a linear array placed in the horizontal plane, we cannot estimate the elevation and estimating the range is associated with high statistical error when the array aperture is small, so only azimuth is used. This also makes the computationally cost much lower. We then discretize the whole space of azimuth ($[0°, 360°[$) into P non-overlapping bins and compute a histogram of the azimuth estimates from (3). The overall estimate of the direction of the source is given by the bin with the highest bin-count. Only a single bin is used,

since we assume that there is only one person speaking in the short segment (max. 2s) except for noise and that this person dominates the noise in terms of loudness.

B. Decision

Based on the observations from the previous section and prior experience, the robot has to make the decision whether the audio segment was generated by a human trying to get the attention or a noise source (door slamming, radio/tv playing etc.), thus we can define two hypothesis, H_0 : Noise source and H_1 : Human speech. The posterior probability of one of the hypothesis given some observations can be stated using Bayes formula [11] as

$$p(H|Y) = \frac{p(Y|H)p(H)}{p(Y)} \tag{4}$$

where Y is the observation. We define our observation as $Y = [X\ W]$, where $X \in [0, 1]$ is the average harmonicity of the audio segment given by (2) and $W \in \{w_1,\ ..\ w_P\}$ is the global bin/direction from where the sound is estimated to come from. Due to the robot rotating, the local estimate from SSL needs to be mapped into a global bin/direction. Assuming X and W are conditionally independent given H, we can rewrite (4) as

$$p(H|X, W) = \frac{p(X|H)p(H|W)p(W)}{p(X, W)} \tag{5}$$

Based on (5) we can now find the posterior probability ratio of the two hypothesis given by

$$\frac{p(H_1|X, W)}{p(H_0|X, W)} = \frac{p(X|H_1)p(H_1|W)}{p(X|H_0)p(H_0|W)} \underset{H_0}{\overset{H_1}{\gtrless}} 1 \tag{6}$$

and make the robot act accordingly. Now the question is how to choose or find $p(X|H)$ and $p(H|W)$. In this work we set $p(X|H_0)$ and $p(X|H_1)$ to be beta distributions with some parameters, due to the fact that the support of a beta distribution matches the range of valid values for harmonicity. The values have been chosen based on initial experiments.

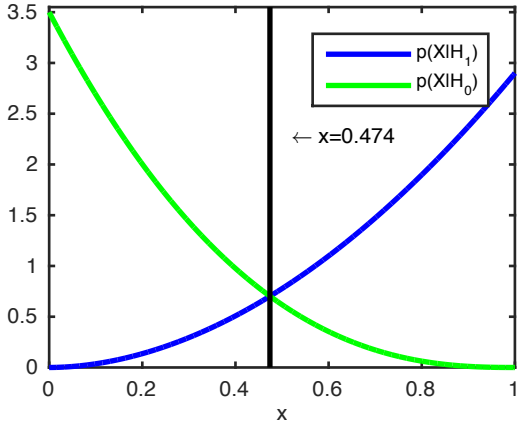


Fig. 3. Probability distributions for $p(X|H_1) = \text{Beta}(2.9, 1)$ and $p(X|H_0) = \text{Beta}(1, 3.5)$, along with the boundary between classifying as speech or noise source.

C. Updating $p(H|W)$ using Face Detection

We set $p(H_0|W) = p(H_1|W) = 1/2$ initially for all bins/directions to reflect the fact, that in the beginning the robot does not know anything about the locations of humans or noise sources. However, whenever the robot turns towards a detected sound it can use the images captured by the camera and perform face detection [12] to gain more information. Based on this principle we can define two different rules for updating $p(H|W)$ after every turn/rotation the robot does. It is assumed that after the robot has turned it attempts to acquire M consecutive images to ensure robustness and the region of the image to perform face detection is limited to the center of the image. In practice, it sometimes fails to run face detection within the time limit, thus only $K \leq M$ images with face detection are available. To clarify notation, we denote the observed value of W as $\hat{w}$.

1) Rule 1: This rule goes as follows

$$p_C(H_1|W) = \begin{cases} \beta_1 \cdot p_P(H_1|W) & \text{if face detected} \\ \beta_2 \cdot p_P(H_1|W) & \text{if } \textit{no} \text{ face detected} \end{cases} \tag{7}$$

$$\text{for } W = \hat{w}$$

where $p(H_0|W) = 1 - p(H_1|W)$ and $p_C(H_1|W)$ and $p_P(H_1|W)$ are the current and previous values of $p(H_1|W)$, respectively. $p(H|W)$ is furthermore constrained to be within $[\epsilon, 1 - \epsilon]$. A face is said to be detected if a face is detected in at least one of the M images. This has the advantage of being simple, however it does not reflect the fact that face detection is imperfect and results in false-alarms and missed detections. Furthermore, the environment of the robot might change over time thus $p(H|W)$ for some values of W might not have been updated for a long time, i.e. the robot's knowledge is based on old information and may therefore be associated with high uncertainty.

2) Rule 2: In *Rule 1* it is implicitly assumed that face detection is always correct, however this is not the case. A more natural way of updating $p(H_1|W)$ would be to incoorporate the (un)certainty connected to the given face detection estimate. We assume that for each face detected there is a scalar associated, $\rho \in [0, 1]$, where 0 means no face detected and 1 means that we are sure a face is detected. The value is computed by using the number of detections under different scalings in the face detection algorithm and taking the average value. Since K images are available, an average value is given by

$$\bar{\rho} = \frac{1}{K} \sum_{i=1}^{K} \rho_i \tag{8}$$

where ρ_i is the value of ρ for the ith image. The update rule can now be defined as

$$p_C(H_1|W = \hat{w}) = \alpha_L \cdot \bar{\rho} + (1 - \alpha_L) \cdot p_P(H_1|W = \hat{w}) \tag{9}$$

where $\alpha_L \in (0, 1)$ is a learning parameter to be determined. To account for the fact that the environment might change over time we introduce a forgetting rule as follows

$$p_{t+1}(H_1|W) = \alpha_F \cdot 0.5 + (1 - \alpha_F) \cdot p_t(H_1|W) \tag{10}$$
$$\text{for } W = \{w_1, w_2, ..., w_P\}$$

where $\alpha_F \in (0, 1)$ is a forgetting parameter to be determined and t denotes time measured in seconds. In this way, if nothing has occured for a long time (i.e. $t \to \infty$) then $p(H_1|W) = p(H_0|W) = 0.5$ for all values of W, reflecting the fact that the robot has not received feedback for a long time and is therefore very uncertain about each bin, i.e. it is reset. Equation (10) has a closed-form solution to find $p_{t+\Delta t}(H_1|W)$ given by

$$p_{t+\Delta t}(H_1|W) = 0.5 \cdot \left(1 - (1 - \alpha_F)^{\Delta t}\right) + (1 - \alpha_F)^{\Delta t} \cdot p_t(H_1|W) \quad \text{for } W = \{w_1, w_2, ..., w_P\} \tag{11}$$

such that this is only updated just in time for computing (6).

III. Experiments & Results

To evaluate the proposed method and its ability to detect and ignore noise sources, a total of four experiments were carried out. All experiments consist of two human speakers and a single loudspeaker playing noise placed as depicted in figure 6 for exps. 1 and 2 and as depicted in figure 9 for exps. 3 and 4. In all experiments the loudspeaker was constantly playing while the two speakers took turns trying to attract the attention of the robot, i.e., speaker 1 would talk until the robot was facing him and then speaker 2 would start talking after a short pause until the robot was facing him and so on. The noise types (see Tab.I for details) were chosen such that the frontend VAD would be triggered and $p(X|H_1)$ yield high likelihood, i.e., resemble speech. Experiment 1 and 2 use *Rule 1* to update $p(H|W)$ and in experiment 3 and 4 update *Rule 2* is used. In these experiments we set $P = 36$ resulting in bin widths of 10° and we use a frame size of 512 samples with no overlap for SSL. For face detection we use $M = 5$ images to perform face detection on.

A. Experiments 1 & 2

Table I shows the setting for exps. 1 and 2. Figure 4 shows the results obtained in experiment 1 where the loudspeaker was constantly playing music. Figure 4(b) shows in which direction (angle) the robot was facing at a certain time and figure 4(a) shows the values of $p(H_1|W)$ for values of W corresponding to bins in $[-25^\circ, 45^\circ]$. The two figures are aligned in time and should be interpreted in the following way: at time $t \approx 40$s the robot decides to turn towards the noise source located at 10°, then after having turned it detects no faces in any of the $K \leq M$ images and updates $p(H_1|W)$ according to (7) for the value of W corresponding to the bin $(5^\circ, 15^\circ]$ in which the noise source is located. At $t \approx 50$s the robot detects sound from speaker 1 and turns towards him, detects at least one face and increases $p(H_1|W)$ for W corresponding to bin $(-5^\circ, -15^\circ)$. This continues until $t \approx 118$s where $p(H_1|W)$ for W corresponding to the bin containing the noise source has been decreased so much, that the robot does not turn towards it anymore, thus only turning towards the two speakers.

Figure 5 show the same as figure 4 but for an experiment where the loudspeaker was playing radio (people talking). The behaviour of the robot is the same, however it ignores the noise much faster than in experiment 1. The robot does however turn towards the noise at $t \approx 142$s, but does not detect any faces thus decreasing $p(H_1|W)$ further.

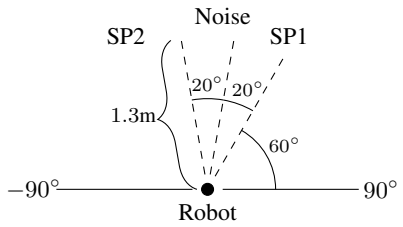


Fig. 6. Illustration of setup of exps. 1 and 2.

TABLE I. Settings for exps. 1 and 2.

	Exp. 1	Exp. 2
Noise type	Music	Radio
Avg. SPL. Noise	47 ± 4dB	51 ± 4dB
Azimuth (SP1, Noise, SP2)	$(-10^\circ, 10^\circ, 30^\circ)$	$(-10^\circ, 10^\circ, 30^\circ)$
(β_1, β_2)	$(1.2, 0.8)$	$(1.2, 0.8)$

B. Experiments 3 & 4

In experiment 3 and 4 the loudspeaker was playing radio and the values for α_L and α_F can be seen in II and were based on initial experiments. Figure 7 show the results obtained for experiment 3, where a small value of α_F is used. At time $t \approx 8$s the robot turns towards the noise source and detects no faces, i.e. $\bar{\rho} = 0$ and decreases $p(H_1|W)$ for W corresponding to the bin containing the noise source. Compared to experiments 1 and 2, the decreas is much more drastic due to using a different updating rule. At time $t \approx 18$s the robot turns towards speaker 1 (figure 7(b)) and detects a face with a high value of $\bar{\rho}$ resulting in a high increase of $p(H_1|W)$. The robot only shifts between directing attention towards the speakers and ignoring the noice source until $t \approx 98$s where it turns towards the noise source again. This is caused by introducing the forgetting rule in 10 and it is seen on figure 7(a) that the value of $p(H_1|W)$ for W corresponding to the bin containing the noise source has been slowly increasing towards 0.5 since $t \approx 8$s. The robot further turns towards the noise source at $t \approx 163$s but again decreases $p(H_1|W)$, thus ignoring it. To further investigate the ability of the proposed method to ignore noise sources, which have speech-like characteristics, a plot showing the detected audio segments detected by the VAD along with the global SSL angle is shown in figure 7(c). Green color means that the segment is classified as noise source, thus rejected, and blue color means it is classified as coming from a human. It is clear that the method is capable of ignoring/rejecting audio events from the direction of the noise source in 9 cases. To verify that the rejections are due updating $p(H|W)$ and not purely due to $p(X|H)$, the mean value and standard deviation of harmonicity, X in eq. 2, of all the rejected and accepted speech segments are calculated to be $0.54(0.065)$ and $0.64(0.069)$, respectively. The mean values for the rejected segments are above the crossing point ($X = 0.474$) in figure 3, which means that they would have been classified as human speech, if no feedback was used.

Figure 8 shows the results obtained from experiment 4. The same performance is seen as in experiment 3, however the forgetting paramter, α_F is set to a higher value resulting in the robot turning towards the noise source much more frequent compared to experiment 3. Again the detected segments from

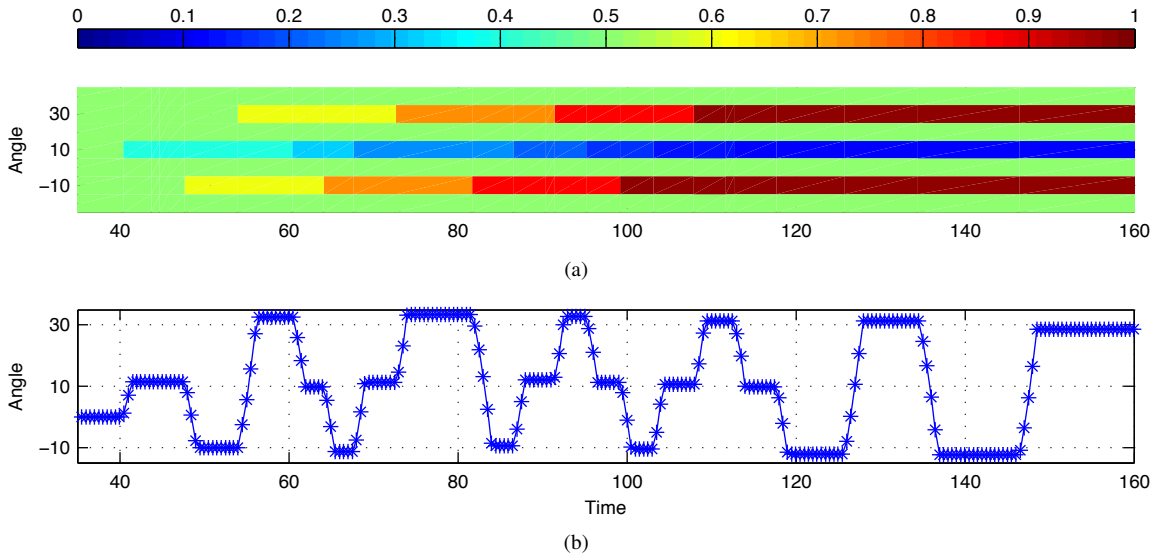


Fig. 4. Figure 4(a) shows $p(H_1|W)$ for the bins in $[-25°, 45°]$ over time, and figure 4(b) shows the direction of attention of the robot. Music was used as noise and update *rule 1* was used to update $p(H_1|W)$.

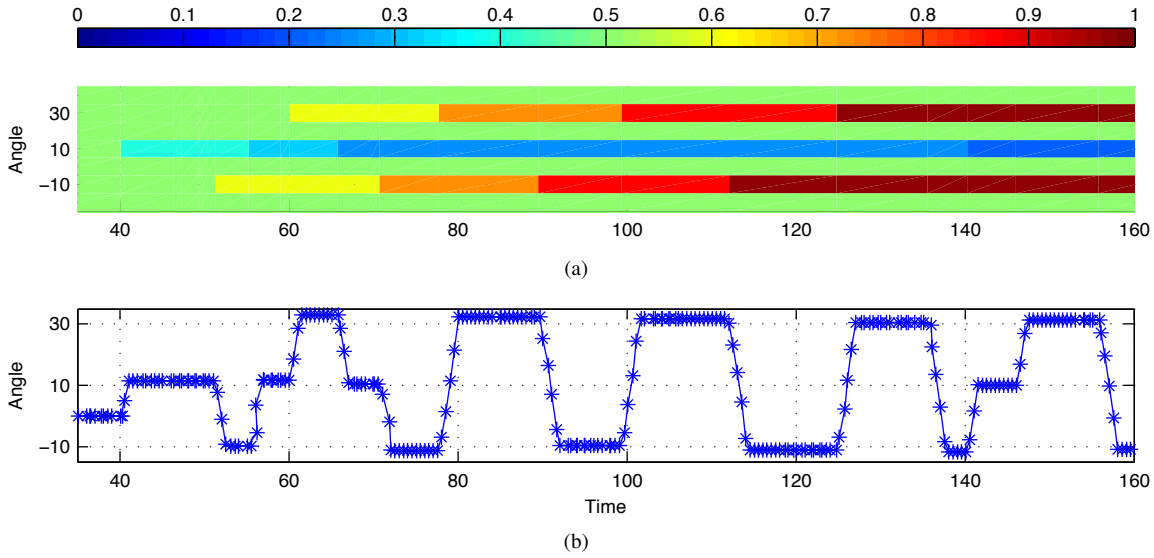


Fig. 5. Figure 5(a) shows $p(H_1|W)$ for the bins in $[-25°, 45°]$ over time, and figure 5(b) shows the direction of attention of the robot. Radio was used as noise and update *rule 1* was used to update $p(H_1|W)$.

the VAD is plotted with the corresponding angle, and again it is seen that the noise source is rejected as speech 5 times, which is not as effective as the previous experiment, but this is due to having a higher value of α_F. The mean and standard deviations for harmonicity for rejected speech segments and accepted speech segments are $0.52(0.49)$ and $0.62(0.055)$, respectively, which is again verifies that using feedback from face detection can improve classification between speech sources and noise sources.

TABLE II. SETTINGS FOR EXPS. 3 AND 4.

	Exp. 3	Exp. 4
Noise type	Radio	Radio
Avg. SPL. Noise	51 ± 4dB	51 ± 4dB
Azimuth (SP2, Noise, SP1)	$(-20°, 10°, 30°)$	$(-20°, 10°, 30°)$
(α_L, α_F)	$(0.8, 0.01)$	$(0.8, 0.05)$

IV. CONCLUSION

In this paper we have presented a method for robots to ignore sound sources which are not of interest to the robot

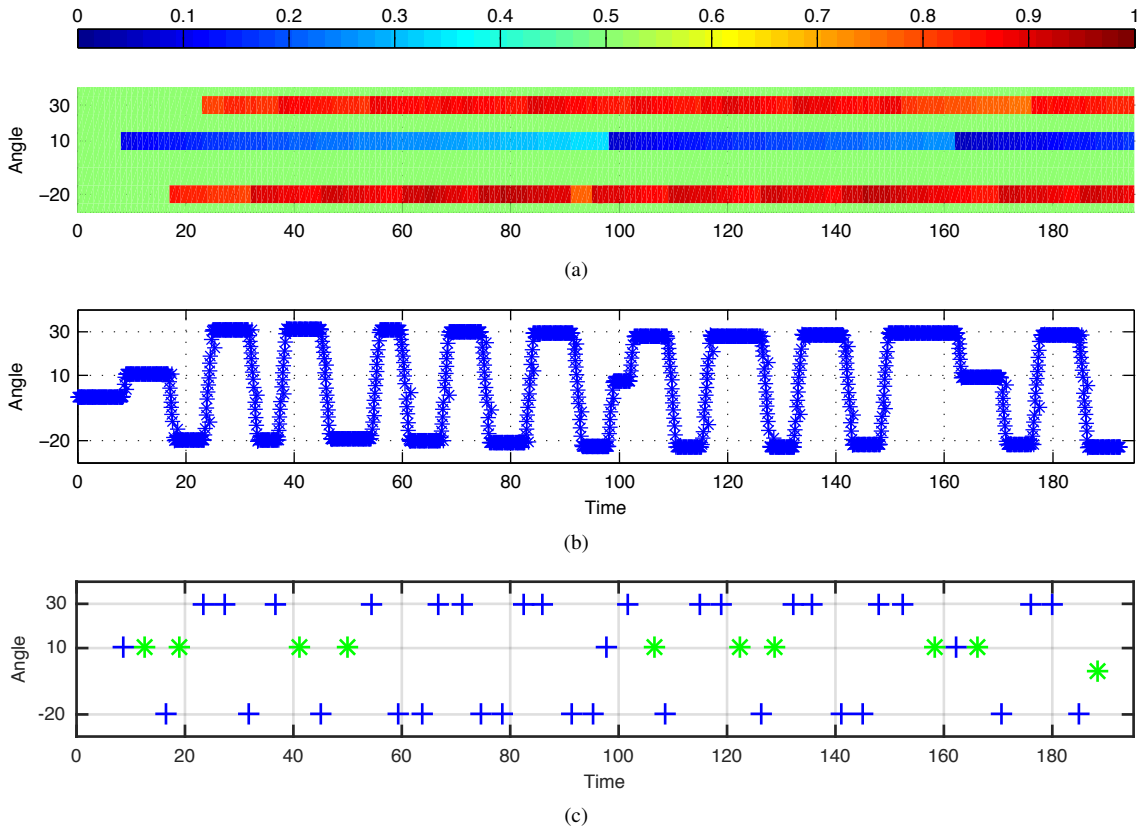


Fig. 7. Figure 7(a) shows $p(H_1|W)$ for the bins in $[-25°, 45°]$ over time, figure 7(b) shows the direction of attention of the robot and figure 7(c) shows the audio events detected by the VAD, where blue means accepted as speech and green means rejected as speech. Radio was used as noise and update *rule 1* was used to update $p(H_1|W)$.

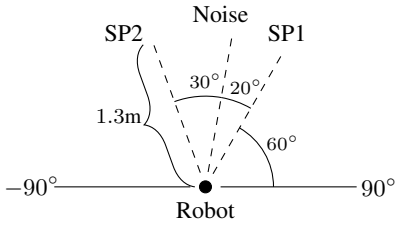


Fig. 9. Illustration of setup of exps. 3 and 4.

in a dialogue/conversation setting. The method uses feedback from face detection to gain knowledge about the relative directions of noise sources and uses this information in the future decision process. Furthermore, two different strategies for using feedback from face detection have been proposed. The method has been tested in four experiments which show the capabilities of the method to locate noise sources and afterwards ignore them, thus only turning towards the human speakers. The system is currently limited to scenarios where the robot is only turning/rotating, so future work should focus on extending the method to scenarios where the robot can move around freely. In this scenario the proposed update rule 1 will most likely not work very well, due to not forgetting or weighting old and new information as equally valid. The proposed update rule 2 will have the capability to handle this depending on the setting of the parameter α_F. It could also be interesting to use different values of α_F depending on whether the bin is believed to be occupied by a noise source or a person, i.e., if it is believed to be occupied by a person, then it makes sense to update it more aggresively due to potential movement. However if it is believed to be occupied by a noise source, this will typically not move, and thus is could make sense to have a low α_F. Another thing to consider in this scenario is when the noise source is luder than the human speaker, thus dominating SSL. In this case it is necessary to hypothesize over multiple SSL-bins, and it may also be necessary to use a SSL method which is able to account for multiple simultaneous sound sources. Another interesting direction to pursue, is updating $p(X|H_0)$ and $p(X|H_1)$ online by also making use of face detection feedback. At last more audio features than harmonicity could be incoorporated to discriminate between speech and non-speech sources more robustly.

ACKNOWLEDGMENT

The authors would like to thank Xiaodong Duan for his help in setting up experiments and contributing to parts of the robot system used.

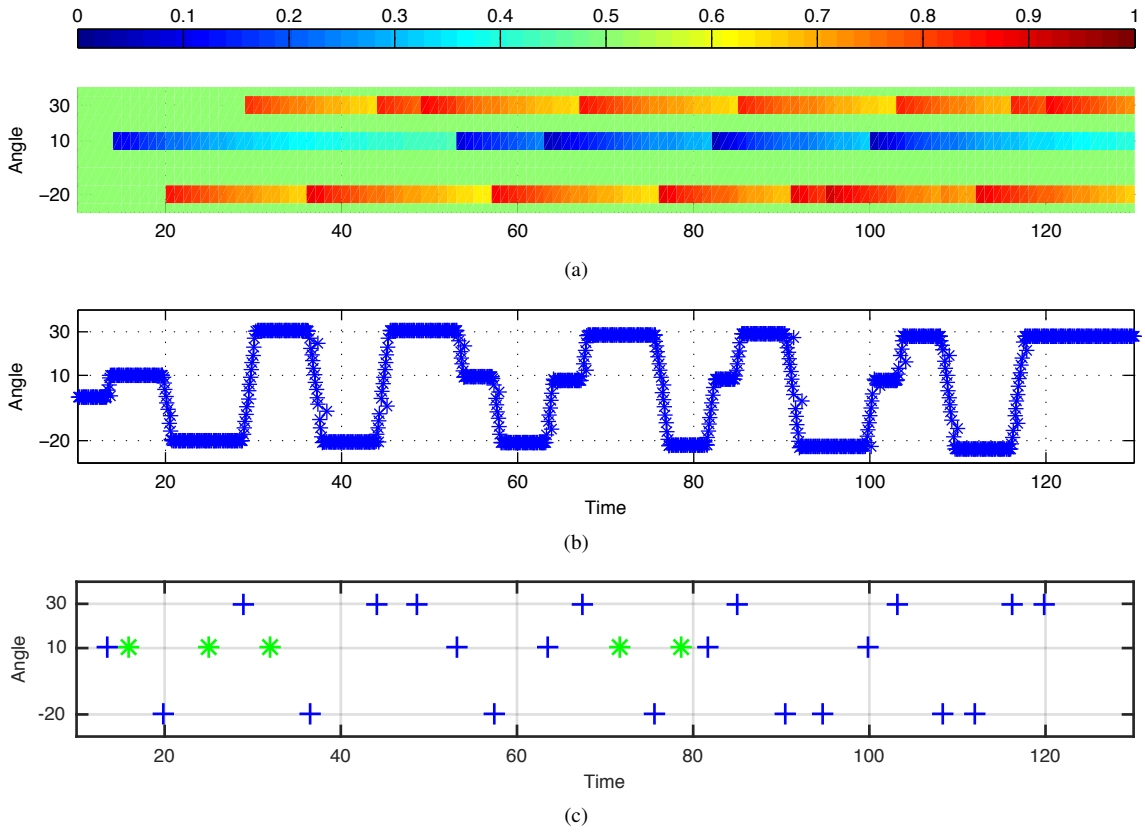


Fig. 8. Figure 8(a) shows $p(H_1|W)$ for the bins in $[-25°, 45°]$ over time, figure 8(b) shows the direction of attention of the robot and figure 8(c) shows the audio events detected by the VAD, where blue means accepted as speech and green means rejected as speech. Radio was used as noise and update *rule 1* was used to update $p(H_1|W)$.

REFERENCES

[1] J. Valin, F. Michaud, J. Rouat, and D. Letourneau, "Robust sound source localization using a microphone array on a mobile robot," in *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings*, vol. 2, Oct. 2003, pp. 1228–1233 vol.2.

[2] T. Rodemann, F. Joublin, and C. Goerick, "Audio proto objects for improved sound localization," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, Oct 2009, pp. 187–192.

[3] K. Nakamura, K. Nakadai, F. Asano, and G. Ince, "Intelligent Sound Source Localization and its application to multimodal human tracking," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2011, pp. 143–148.

[4] H.-D. Kim, K. Komatani, T. Ogata, and H. Okuno, "Auditory and visual integration based localization and tracking of humans in daily-life environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007*, Oct. 2007, pp. 2021–2027.

[5] N. Bæk Thomsen, Z.-H. Tan, B. Lindberg, and S. Holdt Jensen, "Improving robustness against environmental sounds for directing attention of social robots," in *Multimodal Analyses enabling Artificial Agents in Human-Machine Interaction (MA3HMI)*, TO APPEAR.

[6] Z.-H. Tan and B. Lindberg, "Low-complexity variable frame rate analysis for speech recognition and voice activity detection," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 5, pp. 798–807, Oct 2010.

[7] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *IFA Proceedings 17*, 1993, pp. 97–110.

[8] S. Sadjadi and J. Hansen, "Unsupervised speech activity detection using voicing measures and perceptual spectral flux," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 197–200, Mar. 2013.

[9] E. Lehmann and A. Johansson, "Diffuse reverberation model for efficient image-source simulation of room impulse responses," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 6, pp. 1429–1439, Aug 2010.

[10] J. H. Dibiase, "A high-accuracy, low-latency technique for talker localization in reverberant environments using microphone arrays," Ph.D. dissertation, BROWN UNIVERSITY, August 2000.

[11] S. M. Ross, *Introduction to Probability and Statistics for Engineers and Scientists*, 4th ed. Academic Press, 2009.

[12] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

Instrumentation of the da Vinci Robotic Surgical System

Karl Damkjær Hansen Simon Jensen Christoffer Sloth Rafael Wisniewski

***Abstract*—This paper details the AAU surgical robot, its hardware and software setup. The aim of the paper is to explain how a surgical robot has been rebuild as an open source platform for research in surgical robotics. As a result, the robot has full actuation and sensing capabilities at high sampling rate.**

We aim at exploiting the developed surgical robot for research in semi-autonomous control, and safety mechanisms in the context of robotic surgery.

I. Introduction

The development of surgical robots for minimally invasive laparoscopic surgery began in the 1980s, primarily supported by the U.S. Army in e.g. DARPA's Trauma Pod program [1]. Today's leading manufacturer of surgical robotic systems is Intuitive Surgical Inc. that is an offspring of these projects. Their robotic system called da Vinci Surgical System went into use, after being approved by the Food and Drug Administration (FDA) around year 2000, and has since then been an important tool in minimally invasive surgery requiring high accuracy.

The benefit of minimally invasive surgery is less operative trauma to patients. This results in shorter recovery time for patients, with less pain and scarring.

Since the launch of the revolutionary da Vinci Surgical System, the innovation in surgical robots has been relatively slow, among others, due to patenting by Intuitive Surgical Inc. These patents expire by 2016, opening a window of opportunity for competitors to enter the market [2]. The vast future potential in surgical robotics has spawned new robots to emerge. Among these robots is the university-developed Raven open-source surgical robot [3], and the DLR MiroSurge by the German research laboratory DLR.

The AAU surgical robot is based on a da Vinci robot that is retrofitted with hardware and software for developing an open source surgical robot similar to the Raven. The purpose of developing this system is to demonstrate the potential for autonomy in surgical robots. Currently, surgical robots possess no autonomy - the surgeon does all decision making and controls the robot manually with joysticks. We envision a semi-autonomous surgical robot, where responsibilities are shared among the surgeon and a computer system, similar to an autopilot in an airplane. To allow the da Vinci robot to be controlled by a computer, a hardware and software system has been developed. This is explained in this paper.

The paper starts by giving an overview of the physical setup of the da Vinci robot, then the developed hardware system in outlined in Section IV, and finally the software is detailed in Section V.

In this paper, we will only shortly describe the da Vinci surgical system, as this is a commercially available platform. The additional hardware and software that we have developed to control the robot, is described in details.

II. The da Vinci Surgical System

In the late 1990's, SRI International was working on a telesurgery system. The central technology in this system was an "offset remote center maniputlator" [4]. This manipulator allows robotic laparoscopic surgery by creating a fulcrum point for instruments where they intersect the patients abdominal wall. This was accomplished by using a parallel linkage system. In this way, only few stresses are applied to the incision point leading to faster recovery for the patient. See Figure 2.

Concurrently, during his doctoral studies, Akhil Madhani developed the Black Falcon [5], [6], an instrument for robotic laparoscopic surgery, which uses a system of wires and pulleys to actuate a four degrees of freedom end effector. This allows the motors to be located outside the body of the patient, see Figures 3 and 4. Combined with the SRI manipulator, this provides a 7-DOF laparoscopic manipulator; very suited for robotic telesurgery.

The focus of Madhani and his supervisor Kenneth J. Salisbury was on providing a haptic interface for the surgeons performing laparoscopic telesurgery. Consequently, they developed a telesurgery master console [7]. In this console, the surgeon has two 7-DOF joysticks that provides an intuitive interface to the manipulator discussed above. Along with the joysticks, the console has a 3D vision system and foot pedals for controlling an endoscope.

The company Intuitive Surgical Inc. acquired the patents for all these inventions to produce their da Vinci Surgical System. The original da Vinci system has been further developed into the da Vinci Si and Xi systems.

Fig. 1. The da Vinci surgery robot as it sits in the laboratory at Aalborg University. One arm is partially disassembled for easy access to the actuators and sensors.

III. Robot

The AAU surgical robot is a first generation da Vinci Surgical System, modified to allow measurement of all motor currents, velocities, and positions as well as grabbing the video from the stereo endoscope. The designed hardware allows the control of all motors of the robot's four arms, each having 7 degrees of freedom. Figure 1 shows a picture of the robot, where it is seen that there are three arms for equipping surgical instruments and one arm for the endoscope.

The arms of the robot are configured as shown in Figure 2, where the grey section of the robot is fixed during surgery, and the black offset remote center manipulator can be actuated with three degrees of freedom. The reason for fixing parts of the robot is to fix the manipulator with respect to the incisions in the patient's body. Each arm of the robot has a total of 12 degrees of freedom when including the fixed joints of the arm which are released during configuration.

The final part of the robot is the instrument that comes in several configurations. A gripper-type instrument for suturing is shown in Figure 3 and provides the last four degrees of freedom for the arm.

IV. Hardware

In the configuration intended by Intuitive Surgical Inc., the surgeon master console is directly coupled to the da Vinci patient cart. The AAU surgical robot, however, uses only the patient cart by tapping into the connection to the console. A fair amount of reverse engineering has resulted in a system that is able to measure the state of the manipulator arms and control their servos. For modularity reasons, each arm has a dedicated control system.

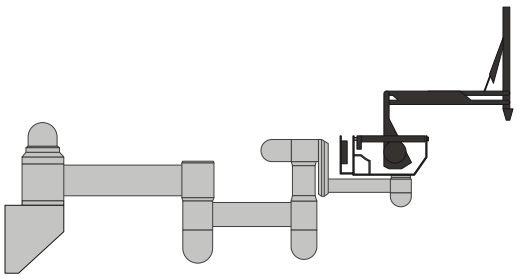


Fig. 2. Sketch of one robot arm. The black manipulator part provides a remote fulcrum point via a parallel linkage.

Fig. 3. Sketch of a needle-driver instrument.

The hardware system interfacing each arm consists of two embedded computers National Instruments single board RIO controllers, which has the two main tasks:

- Getting measurements (positions, velocities, motor currents) from the sensors on the arm and sending them to a client PC, and
- getting control reference signals from the PC to drive the motors that controls the motors.

A diagram of the system is shown in Figure 5. Beside these two tasks, the embedded computers enforce security constraints like limiting control values to protect the motors and stopping the motors when the joints reach their limits.

A surgical robot is a safety critical system. Therefore, one of the embedded computers is in charge of handling the signals from sensors and actuators, while the other acts as the supervisor; handling error situations and dis-/enabling the motor drivers.

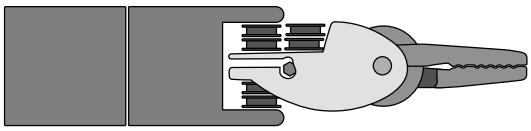


Fig. 4. Sketch of the tip of the needle-driver instrument.

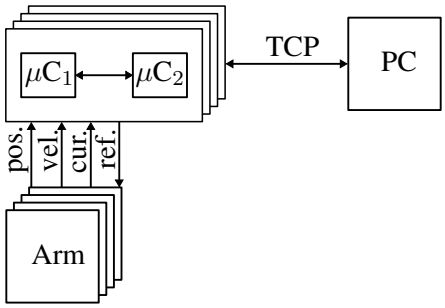


Fig. 5. System diagram of the robotic system connected to a client PC.

The client PC and the robot are connected via a TCP/IP network. This interface is very convenient, as it is ever present on most computers. However, this may introduce timing problems, as the network may lose packages that must be retransmitted or the network may simply present a delay. The timing problems can be reduced by directly connecting the client and the robot without any other computers on the network. But in contrast, the system will natively support remote surgery, where the client PC may be located far away from the robot.

Each joint is fitted with both encoders and potentiometers, thus providing both precise relative and absolute measurements of position and velocity. Each single board RIO has an onboard FPGA, which has been configured to interface to the sensors.

The control system also hosts a bank of ESCON motor controllers, one controller per servo in the arm. These controllers are set and tuned to speed control with inner current control loop of the servos and they also provide a measurement of the motor current to the embedded computers.

V. Software

Low level functions are handled in the embedded computers of the control system. The low level processes all run in real-time and include the speed and current control loops and system error checking. All the higher level functions are handled in the client computer. These functions are characterized as processes which are not possible to run in real-time, require significant computing power, and/or have loose timing constraints. These functions are path planning, user interaction, and position control loops.

The software on the client PC is built on the ROS robotic middleware, which provides visualization of the robot, inverse kinematics computations, image processing, and path planning. The interface between the embedded computers and the client is based on TCP/IP connections with a JSON based serialization protocol. This interface is implemented in C++ in a ROS node called `davinci_driver`. This node implements the `ros_control` C++ interface `hardware_interface::RobotHW`. This allows the system to use the plug-in controller structure of the `ros_control` package and employ e.g. community contributed general purpose PID controllers, but also to write custom controllers using the C++ interface `controller_interface::Controller`.

The JSON based protocol is a compromise between a packet size optimization and a protocol that is easy to implement and debug. The JSON serialization [8] provides a human readable format which is self-descriptive but rather compact compared to e.g. XML. The default message rate from the control system to the client PC is 100 Hz, which is a compromise between being fast and avoiding bugs that are bandwidth related and difficult to identify while developing. In comparison, the da Vinci surgeon console uses an update rate of 1300 Hz [9]. The theoretical limit of the JSON protocol is in the range of 25000 Hz, as the system is using 100BASE-T Ethernet capable of transferring 12,500,000 characters per second and each status update message is around 500 characters long. But as each arm is equipped with two embedded computers (eight in total), all reporting updates, the update rate may rather be in the range of 3000 Hz, not including overhead and leaving no margins.

A physics simulation of the robot is being developed to run in the Gazebo robot simulator. This will allow development and testing of methods and software with out having access to the physical robot.

All the software for the system is open-sourced and available on GitHub [10].

VI. Control

The designed control system must enable the robot to follow a trajectory given by either a surgeon or a computer. Therefore, a position controller is designed for each joint of the robot. It is chosen to implement the controllers with the cascade structure shown in Figure 6. The controller Ctrl 1 is a current controller implemented at the motor driver, and controller Ctrl 2 is a velocity controller implemented at the motor driver as well. Controller Ctrl 3 is a position controller implemented as a PI-controller in the PC; this controller must ensure that the setpoint is attained. The position controllers can either run entirely on the client PC, or they can be run in

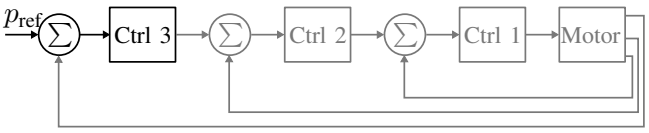


Fig. 6. The cascaded control structure of a single joint. The gray blocks of the diagram are implemented on hardware, and the black box is implemented on a PC.

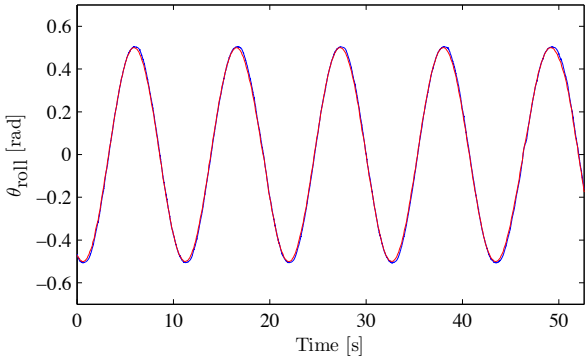


Fig. 7. Roll angle of the robot arm (blue line) and reference for the roll angle (red line).

the embedded computers if an increased sampling rate is desired.

It is chosen to use a cascade structure for the controllers to attenuate disturbances, and minimize friction effects. Implementing the inner loops of the controller in the motor controller with high sampling rate also implies that the position controller can have a relatively large bandwidth. A large bandwidth is crucial when humans are operating the robot via a joystick. In this scenario a delay exceeding 40 ms is unacceptable; however, the designed controller complies with this requirement.

The performance of the controller for the roll angle of the robot is illustrated in Figure 7, with a sinusoidal reference signal. This dynamics of the roll angle is the slowest of the robot, even though two motors are driving the joint, as this movement involves the largest moment of inertia.

The current focus with regards to control is on implementing the safety feature detailed in [11] in the control system. This will ensure that constraints on the robot configuration are not violated.

VII. Conclusion

This paper has presented an overview of software and hardware designed to rebuild a da Vinci Surgical System into an open-source platform for research in surgical robotics. Custom hardware has been designed to control the motors of the robot, and a high-level software based interface has been developed in ROS. All software for the system is open-source and available on GitHub.

We currently aim at exploiting the developed robotic system for developing semi-autonomous control for robotic surgery, and to include safety features such as virtual fixtures. Also we aim at including feedback from the stereo endoscope in the control.

Acknowledgment

This work is supported by the Spar Nord Fonden.

References

[1] J. Rosen and B. Hannaford, "Doc at a distance," *IEEE Spectrum*, pp. 34–39, October 2006.

[2] A. N. Hoffman, "Intuitive surgical, inc.: How long can their monopoly last?" Rotterdam School of Management, Erasmus University and Bentley University, Tech. Rep. 310-106-1, 2010.

[3] B. Hannaford, J. Rosen, D. W. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. N. Kosari, and L. White, "Raven-II: An open platform for surgical robotics research," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 4, pp. 954–959, April 2013.

[4] J. F. Jensen, "Remote center positioning device with flexible drive," Oct. 6 1998, US Patent 5,817,084.

[5] A. J. Madhani, G. Niemeyer, and J. K. Salisbury, "The black falcon: A teleoperated surgical instrument for minimally invasive surgery," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, Oct 1998, pp. 936–944 vol.2.

[6] A. Madhani and J. Salisbury, "Articulated surgical instrument for performing minimally invasive surgery with enhanced dexterity and sensitivity," Aug. 11 1998, US Patent 5,792,135.

[7] J. K. Salisbury, A. J. Madhani, G. S. Guthart, G. D. Niemeyer, and E. F. Duval, "Master having redundant degrees of freedom," Jan. 27 2004, US Patent 6,684,129.

[8] *The JSON Data Interchange Format*, ECMA International Std. 404, 2013.

[9] R. Faust, *Robotics in Surgery: History, Current and Future Applications*. Nova Science Publishers, 2007.

[10] K. D. Hansen, S. Jensen, and C. Sloth. (2013) Aalborg university-robotic surgery group software repository. [Online]. Available: https://github.com/AalborgUniversity-RoboticSurgeryGroup

[11] C. Sloth and R. Wisniewski, "Towards safe robotic surgical systems," in *Proceedings of the IFToMM Symposium on Mechanism Design for Robotics*, Aalborg, Denmark, 2015, pp. 165–175.

Real-Time Augmented Reality for Robotic-Assisted Surgery

Martin Kibsgaard and Martin Kraus
Department of Architecture, Design and Media Technology
Aalborg University, Denmark
{kibsgaard, martin}@create.aau.dk

***Abstract*—Training in robotic-assisted minimally invasive surgery is crucial, but the training with actual surgery robots is relatively expensive. Therefore, improving the efficiency of this training is of great interest in robotic surgical education. One of the current limitations of this training is the limited visual communication between the instructor and the trainee. As the trainee's view is limited to that of the surgery robot's camera, even a simple task such as pointing is difficult. We present a compact system to overlay the video streams of the *da Vinci* surgery systems with interactive three-dimensional computer graphics in real time. Our system makes it possible to easily deploy new user interfaces for robotic-assisted surgery training. The system has been positively evaluated by two experienced instructors in robot-assisted surgery.**

I. Introduction

The motivation for this work is to improve training efficiency for surgeons and medical students who are learning to use the *da Vinci* robotic surgery system for minimally invasive surgery. The training in robotic-assisted surgery is considered expensive, but it is also crucial for improving the outcome of operations [1]. A *da Vinci* system allows a surgeon to perform surgery inside a patient by controlling robotic arms through small incisions. A stereoscopic endoscope (camera) is also inserted to allow the surgeon to see the operating field. The basic setup of a *da Vinci* system is illustrated in Figure 1.

By observing training sessions and interviewing the instructors at Aalborg University Hospital (AUH), it became apparent that the limitation of the communication between the trainer and trainee is a significant problem in the training with

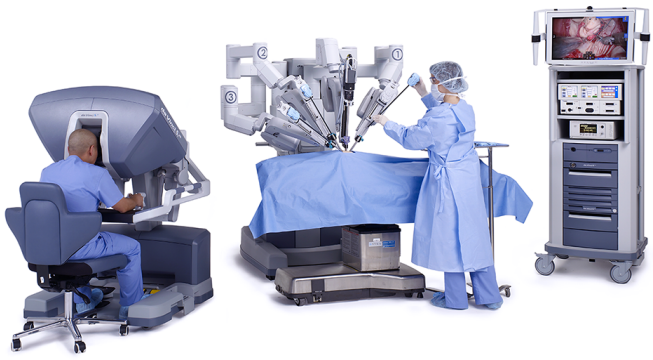


Fig. 1. *da Vinci* surgery system. Left: surgeon operating the surgeon console. Center: assisting nurse operating the patient cart. Right: vision cart. Copyright 2015 Intuitive Surgical, Inc. [2]

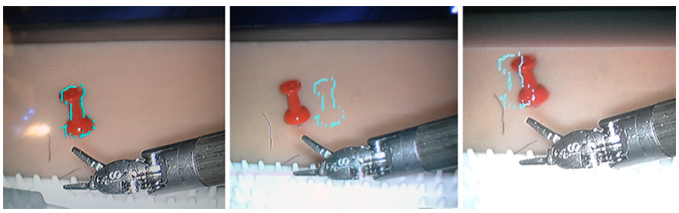


Fig. 2. Telestration. Left: drawing around a pen on the touch screen. Center: the drawing as displayed in the surgeon console's left display. Right: the drawing in the right display. The drawing gets offset for both eyes, making it appear at a different depth than the background. For some users this creates double vision and thus makes precise pointing impossible.

the *da Vinci* system. The source of this problem is that the vision of the trainee sitting at the surgeon console is limited to that of the stereoscopic endoscope, which makes it difficult for the instructor to visually communicate with the trainee. Even a simple task such as pointing to a place in the operational field is currently difficult and anything more advanced is nearly impossible. If the trainee leans back from the console to gain vision of the instructor and/or operating room, the surgical system locks and the trainee loses vision of the surgical field. Often it is necessary for the trainee to leave the console for the instructor to take over to, for example, identify anatomy or demonstrate a skill.

Currently, the *da Vinci* system offers two methods of visual communication without disrupting operation of the robot: drawing on a touch screen (telestration) or showing additional video signals side-by-side with the endoscopic view (a feature called TilePro, see Figure 3). However, neither are being used during training at AUH. The 2D drawings from the telestration do not translate well to the stereoscopic display in the surgeon console (see Figure 2) [3], [4]. The drawings are displayed at a fixed depth in the stereoscopic view, which often results in double vision and, therefore, greatly limits the situations in which they are usable. The TilePro feature is not being utilised during training sessions as it reduces the size and resolution of the endoscopic view.

In this work, we present a system that uses hardware keying to overlay the stereoscopic video signal with computer graphics with minimal latency. The system allows the instructors to communicate more precisely with the trainee without interrupting the trainee's operation of the robot. We describe how our system improves on previous work in Section II. In Section III we describe the video hardware of the *da Vinci* systems and our system — including input and output devices. Evaluation of the system with two experienced instructors

is described in Section IV. The system creates a foundation for multiple new user interfaces that we propose and discuss further in Section VI.

II. Previous Work

To solve the visual communication limitation, Galsgaard et al. [5] developed a system that overlays the stereoscopic video displayed in the surgeon console with 3D computer graphics. However, the 164 ms latency induced by the system made it difficult to operate the *da Vinci* robot and difficult to validate the benefits of the system. Ali et al. [3] created a similar system where they overlaid the stereoscopic video to create and evaluate 3D telestration. Like Galsgaard et al. [5] and other similar work [6], [7] (we assume add 400 ms delay), their overlay approach also involves data transfer with a CPU and multiple format conversions, which presumably introduces a noticeable delay of the video signal.

The previous works most similar to our system are [8] and [9]. Hattori et al. [8] employ an OCTANE MXE graphics workstation that can directly overlay computer graphics on an analog S-Video signal. It is, however, no longer in production and only the first generation *da Vinci* system uses S-Video for the stereoscopic video signal. Figl et al. [9] use two external video mixers to overlay graphics with minimal latency — also on the S-Video signal of the first generation system. Newer video mixers that support the digital HD video format of the newer *da Vinci* systems are available, but they are often bulky and expensive, making them less ideal for the scenario at AUH. An exception to this, is the Blackmagic Design ATEM Television Studio switcher; however, to reduce the delay to less than one frame (if possible at all) would require a more costly and bulky solution than the system we describe in Section III. It would require two switchers, SDI outputs from a computer and synchronization of the signals. A delay of one frame might appear very small, but at 60 Hz, it corresponds to more than 16 ms, while a 10 ms delay can already make a statistically significant difference in user performance [10].

Most other works that augment the video signal of the *da Vinci* systems use the TilePro feature to display both the original (undelayed) and the augmented video signal at the same time [11], [12], [13]. In this way, they avoid that the additional latency affects the control of the robot. However, as can be seen in Figure 3, this significantly reduces the size and resolution of both video signals. From [11] it appears that surgeons then tend to switch off the augmented video signal whenever possible. Thus, the TilePro setup is not ideal in a training situation.

None of the cited works appear to support the digital HD video format of the newest generation *da Vinci* systems, which require much faster processing because of the higher data rate (0.3 Gbit/s vs. 1.5 Gbit/s).

III. Materials and Methods

A. Video Hardware

Initially, we investigated the video hardware of various *da Vinci* systems that were available to us (*da Vinci*, *da Vinci S HD*, and *da Vinci Si*). We examined the video signals using a Blackmagic Design DeckLink Duo card and a custom

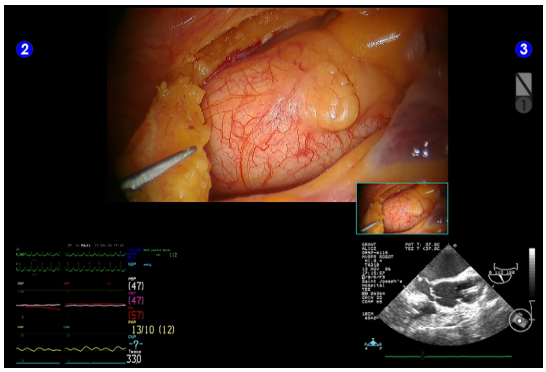


Fig. 3. TilePro feature of the *da Vinci S HD* system seen from the surgeons perspective. Here the feature is used to display physological information (left) and ultra sound video (right). Copyright 2015 Intuitive Surgical, Inc. [2]

made format detection application. The pixel format of all the examined systems is YCbCr 4:2:2, which is a chroma sub-sampling format commonly used in live TV productions. The video components of the systems are connected by RG-59 coaxial cables with BNC connectors and are in most cases directly accessible. The video formats of each generation of the system are listed in Table I.

1) First generation da Vinci: The first generation *da Vinci* system uses S-Video (analog) and SD-SDI (standard definition serial digital interface). The SD-SDI video signal can be intercepted directly between the vision cart and the surgeon console. To overlay graphics on the first generation *da Vinci* system, a single DeckLink Duo card is sufficient, as this PCIe card supports internal keying on two SD-SDI signals at the same time. The stereoscopic video signal consists of two separate SDI signals: one for each eye. The setup for a first generation *da Vinci* system is illustrated in Figure 4.

2) da Vinci S HD and da Vinci Si: The newer *da Vinci S HD* and *da Vinci Si* systems use HD-SDI (high definition SDI) with a 1080i video format. On those systems, the video signal can be intercepted between the camera controller(s) and the synchronizer/CORE unit in the vision cart as illustrated in Figure 5. Additionally, the *da Vinci S HD* system has a redundant HD-SDI output on each camera controller.

For these newer systems, a more advanced video device is required to allow for keying in HD, for example, one DeckLink HD Extreme card for each video signal. We acquired two DeckLink HD Extreme cards (second generation) to support stereoscopic keying on a *da Vinci S HD* system. Recently, Blackmagic Design has released a less costly video card, DeckLink SDI 4K, that also supports internal HD keying.

TABLE I. Video formats of the da Vinci surgical systems. *i* indicates an interlaced format.

	Video Format	Refresh Rate	Year
da Vinci	PAL / NTSC	50i / 60i	1999
da Vinci S	PAL / NTSC	50i / 60i	2006
da Vinci S HD	1080 HD	59.94i	2006
da Vinci Si	1080 HD	59.94i	2009
da Vinci Xi	(1080 HD)	(59.94i)	2014

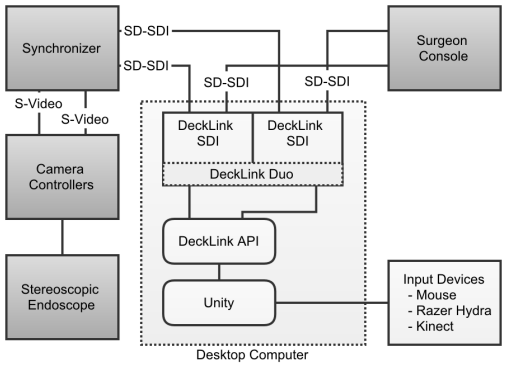


Fig. 4. System diagram of the setup used to overlay 3D graphics on the stereoscopic video of the first generation *da Vinci* system. Dark grey indicates *da Vinci* components.

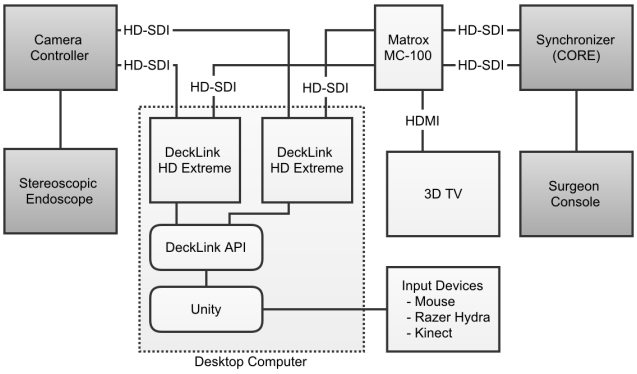


Fig. 5. System diagram of the setup used to overlay 3D graphics on the stereoscopic video of the *da Vinci S HD* and newer systems. Dark grey indicates *da Vinci* components.

B. Computer Graphics

For future applications and to make it possible to overlay three-dimensional (3D) graphics, we implemented a system in the free version of the game engine Unity, which we used to render 3D graphics that are output through the DeckLink API to the DeckLink card(s). We have created a custom graphical interface that allows developers to choose input, output or keying mode for each DeckLink device installed in the system. Furthermore, we implemented virtual instruments resembling those used on the *da Vinci* systems, which can be seen in Figure 6. Currently, the bottleneck of the system is the time it takes to transfer the rendered images from the graphics card to the system RAM, where the DeckLink cards can access them. A workstation graphics card, i.e. NVIDIA Quadro or AMD FirePro, could possibly improve the transfer rate as they have dedicated processing units for exactly that. We emphasize, however, that this bottleneck only delays the computer graphics, and it does not affect the stereo video signal of the endoscope.

C. Input and Output

For the trainer to be able to control the 3D graphics, we employed Razer Hydra game controllers. They support three translational degrees of freedom (DOF), three rotational DOFs and an analog trigger (one DOF); therefore, they are similar to the 7-DOF controls of the *da Vinci* systems [14]. Additionally, we implemented control with the Kinect for Windows v2, which supports skeleton tracking and hand gestures. To display the stereoscopic video signal to the trainer, we used a Matrox MC-100 converter, which converts two HD-SDI signals to 3D HDMI (see Figure 5). The latter signal is displayed by a 3D TV with passive stereo. To minimize latency for the 3D TV, it is important to turn off any processing performed by the TV (often achieved by switching the TV to *game mode*)

To compensate for any inaccuracies in the camera optics, the *da Vinci* systems have a calibration feature, where the two video streams are aligned by offsetting the images both horizontally and vertically. The offsets are introduced after we intercept the video signal and it is therefore necessary to adjust the signal for the 3D TV as well. Fortunately, the MC-100 has an on screen menu where it is possible to offset the images accordingly. Similarly, the virtual cameras have to be adjusted to match the offset, but in the opposite direction, for the computer graphics to be properly aligned.

D. System Setup

The core of our system is a regular desktop computer with the DeckLink cards installed. It has the following specifications: Intel Core i3-3240 3.4GHz CPU, Geforce 660 GTX GPU, 8 GB 1600 MHz RAM, 2×DeckLink HD Extremes (second generation), Microsoft Windows 8. The complete setup installed and tested at AUH is depicted in Figure 5.

To make the use of the *da Vinci* system less dependent on our system, we attached the redundant outputs of the camera controllers to loss-of-signal switchers. This has two benefits: it makes it possible to still use the robot without our system, and it makes it possible to immediately switch to the original video streams in case there are any problems with our system – simply by switching off our system.

IV. Results

The proposed system is able to overlay the video streams of the *da Vinci* system with computer graphics. Like other work it supports the older generation *da Vinci* systems, and it is also

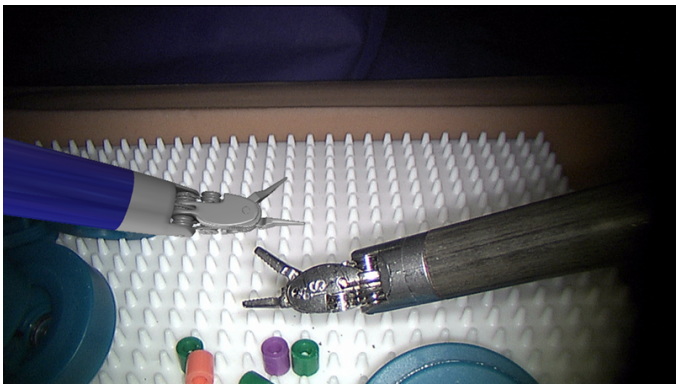


Fig. 6. Endoscopic video signal overlaid with a virtual instrument (blue) in real-time. This composition is shown to the left eye. A similar image is shown to the right eye, but with the real and virtual camera offset to the right.

compatible with the newer generations that utilize HD video signals. Our system is able to produce high quality computer graphics and overlay them in stereo at a rate matching the 1080i59.94 format. The overlaying induces less than 1 ms delay [15] on the original video signal. The computer graphics (e.g. virtual tools) are slightly delayed (less than 100 ms) compared to outputting on a regular computer monitor.

We evaluated our system with the help of two experienced robotic surgery instructors. Our system has twice been connected to the *da Vinci S HD* system normally used for training at Aalborg University Hospital. The tests showed that the system can improve visual communication between the instructor and a trainee. Especially, the overlaid virtual instruments were considered very useful by the experienced surgeon for showing advanced tasks to the trainees, while a simpler cursor was preferable for the assisting nurse.

Regarding user input devices, the Razer Hydra is useful for experienced surgeons to demonstrate certain skills to a trainee. The Kinect sensor has less DOFs, but is sufficient for pointing or drawing, which is often what is needed for the assisting surgeon or nurse. It is also possible to use the Kinect sensor in a sterile environment as it does not require any touching. However, because of the limited space in the training room at AUH, it is not convenient to use the Kinect sensor and a wireless gyroscopic mouse has since proved to be superior.

During the test we noticed that the perceived depth of the 3D graphics (virtual robotic instruments) did not entirely match the depth of the real environment. Furthermore, we learned that a clutch (allowing to move the controls without moving the instruments) is necessary to sufficiently simulate the real instruments. Since the test, the cameras have been adjusted to match properly with the endoscope used at AUH and clutching of the tools has been implemented. They have since been positively evaluated by several of the instructors at AUH.

The stability and usability of the system has since been tested for extended periods of time at multiple eight hour sessions. The only technical problem during the sessions was the 3D TV turning off, presumably, due to power fluctuations caused by the cauterization instruments. It did not affect the overlay system and the 3D TV was simply switched back on.

V. Conclusion

We have created a compact system that can be the foundation for multiple new types of user interfaces directly visible in the surgeon console during training for robotic-assisted surgery. Our implementation allows overlaying of stereo graphics in 1080 HD at 59.94i with less than 1 ms delay.

The developed software for keying graphics from Unity is available at homes.create.aau.dk/kibsgaard/.

VI. Future Work

Our system was developed to support future work and is currently used by students without requiring knowledge of the underlying APIs; i.e., the DeckLink API and OpenGL. Future work could further evaluate the use of overlaying virtual objects, images, videos, webcams, task lists, etc. during training with the *da Vinci* systems.

Acknowledgment

We thank Jane Petersson and Johan Poulsen from Aalborg University Hospital for all their help and sharing of expertise during the development and evaluation of our system.

References

[1] Health Quality Ontario, "Robotic-assisted minimally invasive surgery for gynecologic and urologic oncology," *Ontario Health Technology Assessment Series*, vol. 10, no. 27, pp. 1–118, Dec. 2010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3382308/

[2] Intuitive Surgical. (2014) Intuitive Surgical - Image Gallery - da Vinci Si System. http://www.intuitivesurgical.com/company/media/images/davinci_si_images.html.

[3] M. R. Ali, J. P. Loggins, W. D. Fuller, B. E. Miller, C. J. Hasser, P. Yellowlees, T. J. Vidovszky, J. J. Rasmussen, and J. Pierce, "3-d telestration: a teaching tool for robotic surgery," *Journal of Laparoendoscopic & Advanced Surgical Techniques. Vol. 18, Issue 1*, pp. 107–112, 2008.

[4] C. Hasser, D. Larkin, B. Miller, G. Zhang, and W. Nowlan, "Medical robotic system providing three-dimensional telestration," Patent EP1 965 699 A2, 2008, US Patent EP1965699 A2. [Online]. Available: http://www.google.com/patents/EP1965699A2

[5] B. Galsgaard, M. M. Jensen, F.-O. Matu, M. Thgersen, and M. Kraus, "Stereoscopic augmented reality system for supervised training on minimal invasive surgery robots," *Proceedings of Virtual Reality International Conference (VRIC 2014)*, 2014.

[6] L.-M. Su, B. P. Vagvolgyi, R. Agarwal, C. E. Reiley, R. H. Taylor, and G. D. Hager, "Augmented reality during robot-assisted laparoscopic partial nephrectomy: Toward real-time 3d-ct to stereoscopic video registration," *Urology*, vol. 73, no. 4, pp. 896 – 900, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S009042950801947X

[7] B. Vagvolgyi, L.-M. Su, R. Taylor, and G. D. Hager, "Video to ct registration for image overlay on solid organs," *Proc. Augmented Reality in Medical Imaging and Augmented Reality in Computer-Aided Surgery (AMIARCS)*, pp. 78–86, 2008.

[8] A. Hattori, N. Suzuki, M. Hashizume, T. Akahoshi, K. Konishi, S. Yamaguchi, M. Shimada, and M. Hayashibe, "A robotic surgery system (da vinci) with image guided function–system architecture and cholecystectomy application," *Studies in Health Technology and Informatics*, vol. 94, pp. 110–116, 2003.

[9] M. Figl, D. Rueckert, D. Hawkes, R. Casula, M. Hu, O. Pedro, D. P. Zhang, G. Penney, F. Bello, and P. Edwards, "Image guidance for robotic minimally invasive coronary artery bypass," *Computerized Medical Imaging and Graphics: The Official Journal of the Computerized Medical Imaging Society*, vol. 34, no. 1, pp. 61–68, Jan. 2010.

[10] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Comput. Graph. Appl.*, vol. 21, no. 6, pp. 34–47, Nov. 2001. [Online]. Available: http://dx.doi.org/10.1109/38.963459

[11] F. Volont, N. C. Buchs, F. Pugin, J. Spaltenstein, B. Schiltz, M. Jung, M. Hagen, O. Ratib, and P. Morel, "Augmented reality to the rescue of the minimally invasive surgeon. the usefulness of the interposition of stereoscopic images in the da vinci robotic console: Augmented reality in robotic surgery," vol. 9, no. 3, pp. e34–e38. [Online]. Available: http://europepmc.org/abstract/med/23239589

[12] O. Ukimura, M. Aron, M. Nakamoto, S. Shoji, A. L. d. C. Abreu, T. Matsugasumi, A. Berger, M. Desai, and I. S. Gill, "Three-dimensional surgical navigation model with TilePro display during robot-assisted radical prostatectomy," vol. 28, no. 6, pp. 625–630. [Online]. Available: http://online.liebertpub.com/doi/abs/10.1089/end.2013.0749

[13] A. Hughes-Hallett, P. Pratt, E. Mayer, S. Martin, A. Darzi, and J. Vale, "Image guidance for allTilePro display of 3-dimensionally reconstructed images in robotic partial nephrectomy," vol. 84, no. 1, pp. 237–243. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0090429514003409

[14] K. Grande, R. S. Jensen, M. Kraus, and M. Kibsgaard, "Low-cost simulation of robotic surgery," in *Proceedings of the Virtual Reality International Conference: Laval Virtual*, ser. VRIC '13. ACM, 2013,

pp. 6:1–6:4. [Online]. Available: http://doi.acm.org/10.1145/2466816.2466823

[15] Yoshan, Developer Support, Blackmagic Design, Personal Communication, 2015.

Formation Control of AAUSHIP

Nick Østergaard
Dep. of Electronic Systems
Aalborg University
Denmark

Jeppe Dam
Dep. of Electronic Systems
Aalborg University
Denmark

Jesper A. Larsen
Dep. of Electronic Systems
Aalborg University
Denmark
Email: jal@es.aau.dk

***Abstract*—Many maritime mapping tasks are today carried out by large research ships, which are very costly to operate. As a way to overcome this, a number of small surveying vessels have been developed called AAUSHIP. In order to efficiently map the an area with such smaller vessels, it is important that several vessels are able to corporate on the task at hand. In this paper, the developed formation control strategy for the AAUSHIP series of vessels is presented, along with simulation results, which confirms, that the algorithm works as intended.**

I. Introduction

The background for the formation control subject of this project originate in a collaboration with Port of Aalborg, who has a vision to make their harbour an intelligent harbour. This will, among other things, include autonomous piloting of cargo ships bringing cargo to and from Aalborg. For the cargo ships to enter the harbour it is important that the seabed is deep enough and the sand has not build up larger bars or moved the channel unexpectedly. Currently bathymetry surveys are performed manually with a small manned survey boat equipped with a multi beam sonar, scanning some area of interest, which is a smaller fraction of the whole Limfjord.

This is done with a period between three months up to three years, depending on how active the seabed is. If the level is too shallow, such that the cargo ships cannot enter, it is the Port of Aalborg that needs to clear the area and ensure safe travel for their customers.

The work within this project is carried out to assist the Port of Aalborg with their survey task. The development and implementation of the AAUSHIP project will fit very well into this environment and be of good aid for the Port of Aalborg.

The first focus point of the project is to model and test the prototype of the AAUSHIP and then extend the fleet with duplicates of the first AAUSHIP. The ship needs to follow a trajectory and thereby sail within a predetermined location of interest. The second focus point is to implement formation control of a fleet of AAUSHIP's and test this at the location of interest. An area of the harbour has been given as a use case to test the results against.

Previously research on using an autonomous surface vehicles (ASV) for bathymetry measurement has been reported in Vasilijevic et al. (2015); Ferreira et al. (2009); Bourgeois et al. (1999), however, these were all operating independantly. Some preliminary studies in formation control of surface and sub-surface vehicles have been reported in Simetti et al. (2015) where mechanically linked cooperative sub-surface vessels are considered, in Ghommem et al. (2007) where a non-linear approach to formation keeping of a group of surface vehicles is considered and in Ihle et al. (2005), where an inter-vessel force is introduced to keep the formation in place, which has certain parallels to the potential field approach, which is discussed in section III. However, this is the first time that the formation control problem for ASVs is applied to bathymetry surveying.

II. Method

The AAUSHIP is modelled by a 5 degree of freedom (DOF) model, which differs from a 3 DOF model by including the pitch and roll also. These are taken into account due to the fact that the AAUSHIP runs with single beam sonars and therefore it is important to know the relative pitch and roll angles.

A. Simulation Model

In order to simulate the ships behaviour in water, an accurate simulation model has been developed and verified agains the real ship. The hydro-dynamic model used to simulate the ships is given as:

$$M_A\dot{\nu}_r + C_A(\nu_r)\nu_r + D(\nu_r)\nu_r + g(\eta_r) = \tau \quad (1)$$

where M_A is the added mass matrix from the system, C_A is the added mass matrix due to the Coriolis force, $D(\nu)$ is a combination of the potential and viscous damping matrices, $g(\eta)$ is the restoring forces, which is dependent on the position of the vessel, τ is control and propulsion forces and ν is the velocities of the vessel in all directions and moments.

The rigid body is used to model the physics of the vessel, as it is assumed that the vessel is sufficiently stiff to neglect bending dynamics.

Translational motion and rotational motion can be derived by analysis of this, and by (SNAME, 1950) and (Fossen, 2011, sec. (3.3.1)) written in component form as: $f_b^b = [X, Y, Z]^T$ and $m_b^b = [K, M, N]^T$, force through o_b and moments about o_b expressed in $\{b\}$, $v_{b/n}^b = [u, v, w]^T$ is the linear velocity of o_b relative o_n expressed in $\{b\}$, $\omega_{b/n}^b = [p, q, r]^T$ is the angular velocity of b relative to $\{n\}$ expressed in $\{b\}$ and $r_g^b = [x_g, y_g, z_g]^T$ is the vector from o_b to CG expressed in $\{b\}$

The rigid body forces are written as:

$$M_{RB}\dot{\nu}_r + C_{RB}(\nu_r)\nu_r = \tau_{RB} \tag{2}$$

where M_{RB} is the system inertia matrix, C_{RB} is the coriolis-centriopedal matrix, τ_{RB} is a lumped force combined of τ_{hyd} + τ_{hs}+ τ_{wind} + τ_{wave}.

Combined, this gives the following full vessel model:

$$\underbrace{M_{RB}\dot{\nu}_r + C_{RB}(\nu_r)\nu_r}_{\text{rigid-body forces}} + \underbrace{M_A\dot{\nu}_r + C_A(\nu_r)\nu_r + D(\nu_r)\nu_r + g(\eta_r)}_{\text{hydrodynamic forces}} = \tau + \tau_{RB} \tag{3}$$

Since the vessel within this project is of smaller scale, the C_A and C_{RB} from 1 and 2 are neglected (Wondergem, 2005, eq. (2.23)).

III. Formation control

In the following approach a potential field is generated for each agent including obstacles, formation span, desired, and actual position. It will be a combination of virtual leader and potential field. The principle generates a potential field to keep the formation and that field is moved around as a virtual leader. When the virtual leader is moved around it results in a deflection of the desired position and causes the affected agents to get back into position. The positions of the agents in the field is given individually to the specific agents relative to the virtual leader. The approach generates a single resulting vector for each agent which is used to guide the agent. The potential field for each agent is generated from four components:

$$\tilde{\mathbf{F}}_i^{tot} = \mathbf{F}_{vl} + \mathbf{F}_{ij}^{tot} + \mathbf{F}_{ca}^{tot} + \mathbf{F}_{oa}^{tot} \tag{4}$$

where:

$\mathbf{F}_{vl}$	virtual leader force
$\mathbf{F}_{ij}^{tot}$	inter-agent forces
$\mathbf{F}_{ca}^{tot}$	agent-agent collision avoidance forces
$\mathbf{F}_{oa}^{tot}$	agent-obstacle collision avoidance forces

1) Virtual Leader, $\mathbf{F}_{vl}$: The virtual leader is an anchor of each formation, the Formation Reference Point (FRP), and controls the movement of this. This movement can be given as a full trajectory of as a set of way points. The local virtual leader's contribution to the field is defined as:

$$\mathbf{F}_{vl} = K_{vl}(p_{vl}^n - p_i^n - [p_{vl}^n - p_{i0}^n]) \tag{5}$$
$$= K_{vl}(\mathbf{d}_i - \mathbf{d}_{i0}) \tag{6}$$

K_{vl} is a tuning parameter. p_{vl} is position of the virtual leader, p_i is position of agent i, p_{i0} is desired position of agent i and the d is a shorter notation for the distances in between. The virtual leader component guides the agents directly to their desired positions relative to the virtual leader.

2) Inter Vehicle Influence, $\mathbf{F}_{ij}$: This is the contribution of other vehicles to the potential field, which is expressed as:

$$\mathbf{F}_{ij} = K_{ij}(p_j^n - p_i^n - [p_{j0}^n - p_{i0}^n]) \tag{7}$$
$$= K_{ij}(\mathbf{d}_{ij} - \mathbf{d}_{ij0}) \tag{8}$$

Similar to previously the ps are positions, K_{ij} is a tuning parameter and d is a shorter notation for the distances in between. This component preserves the formation by affecting the agents to keep their respective desired distances among themselves. The weighting on each goal can be adjusted by K_{vl} and K_{ij}, hence this weighting is a weighting that causes the agents to either follow the virtual leader or to preserve their desired formation. In a swarm of N agents the total field for agent i given by:

$$\mathbf{F}_{ij}^{tot} = \sum_{j=1}^{N} \mathbf{F}_{ij}(i,j) \text{ for } j \neq i \tag{9}$$

3) Collision Avoidance, $\mathbf{F}_{ca}$: The collision avoidance takes effect when the agents get closer than a pre defined distance of each other. It generates an additional field component for the vehicle i which points away from the entering agent causing the agents to move away from each other. To ensure the avoidance the component converges towards infinity in the centre of the i'th agent. The $\mathbf{F}_{ca}$ is expressed as:

$$\mathbf{F}_{ca}^{ij} = \begin{cases} \left(\frac{K_{ca}\, r}{||\mathbf{d}_{ij}||} - K_{ca}\right) \frac{\mathbf{d}_{ij}}{||\mathbf{d}_{ij}||}, & \text{for } ||\mathbf{d}_{ij}|| < r \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where K_{ca} is a tuning parameter. r is the safety radius for collision and $\mathbf{d}_{ij}$ is the distance between the individual agents. The collision avoidance can

be expressed in a total term of the collision avoidance:

$$\mathbf{F}_{ca}^{tot} = \sum_{j=1}^{N} \mathbf{F}_{ca}^{ij} \text{ for } i \neq j \tag{11}$$

4) Obstacle Avoidance, $\mathbf{F}_{oa}$*:* The same principle as for collision avoidance can be applied to obstacle avoidance. Now each obstacle needs to be handled as an agent, which will make the same result, but the reference is a little different:

$$\mathbf{F}_{oa}^{ik} = \begin{cases} \left(\frac{K_{oa}}{||\mathbf{d}_{ki}||} - \frac{K_{oa}}{r} \right) \frac{\mathbf{d}_{ki}}{||\mathbf{d}_{ki}||}, & \text{for } ||\mathbf{d}_{ki}|| < r \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

where k denotes the counter for obstacles instead of other agents. K_{oa} is also a tuning parameter for the obstacle avoidance. $\mathbf{d}_{ki}$ is the vector between an agent and the obstacle, which in a total term is summed up as:

$$\mathbf{F}_{oa}^{tot} = \sum_{k=1}^{M} \mathbf{F}_{oa}^{ik} \text{ for } i \neq k \tag{13}$$

Here d_{ki} represents one of the M place vectors which has the effect of a detected obstacle.

It can be noticed that there is a difference between $\mathbf{F}_{ca}$ and $\mathbf{F}_{oa}$. The two forces applies the same directions of forces, making the agents repulse from another agent or an obstacle, but for equal parameters the $\mathbf{F}_{ca}$ generates a larger force, meaning that collision is punished harder than obstacles.

The distance r can be determined dynamically depending on the velocity of the agent:

$$r = r^{min} + K_r ||\dot{\mathbf{p}^n}|| \tag{14}$$

Still will ensure that the agents have the possibility to decelerate from their absolute velocity in the safety radius such that they can turn away from each other.

5) Potential field: The forces are summed together to get $\tilde{\mathbf{F}}_i^{tot}$, which is an intermediate vector which gives the magnitude and direction of the potential field for vehicle i at its current position.

$$\mathbf{F}_i^{tot} = \min\{ \ ||\tilde{\mathbf{F}}_i^{tot}|| \ , \ F_{max} \ \} \frac{\tilde{\mathbf{F}}_i^{tot}}{||\tilde{\mathbf{F}}_i^{tot}||} \tag{15}$$

The $\tilde{\mathbf{F}}_i^{tot}$ denotes that it is a middle variable and not the final value of the potential calculation, thus not the one used by the controller yet. As the potential field does not need to expand to infinity it is reasonable to define a maximum amplitude for the vector, while still keeping its direction, F_{max}. This will be a limitation of the agents' speed. As a start in the simulation phase is F_{max} chosen as a constant, but in the fully implemented system it can be of benefit to adjust this maximum speed dynamically, for instance as in equation 16, as an example from (Paul et al., 2008).

$$F_{max} = F_{min} + K_{vl} ||\dot{p}^n|| \tag{16}$$

where the F_{min} is a minimum value for the upper limit and then with an applied gain of the speed. The reference trajectory is used by the controller to calculate the agent's control input which can be based on the desired movement in the NED frame as:

$$\mathbf{p}_{i,r}^n = \mathbf{p}_i^n + \mathbf{F}_i^{\text{tot}} \tag{17}$$

where their positions are added together with the potential field, such that the position and the potential field becomes linked.

IV. The Potential Field Strategy

The potential field is generated for each individual agent at every update step to make the formation move and converge to a specified formation and position. The field is generated based on forces acting in an overlying potential field structure where one force converges the agent to a desired position, a force attracting the agent to obtain the desired formation along the trajectory, a force repelling the agent from other agents if their distance is too small and finally a force repelling the agents from static objects. The latter two can seem the same, but the repelling force will be larger for the agent-agent force due to the fact that two agents could have course directly toward each other and a more aggressive avoidance can be needed.

To be able to generate and simulate the potential field the implementation needs to be generic. First it was developed with one agent that needs to converge to a desired position and afterwards were other agents added as obstacles and some static objects were added in extend. From these obstacles it can be seen that a single agent is able to converge to a position which makes it possible to expand such that more agents can converge into formation with reference from either a virtual leader or from each other. This will solve the formation coordination task, where the following task will be the group coordination task. The group coordination task has the goal to move the formation around, which here will be done by making the virtual leader, or an actual leader of the formation, follow a specified trajectory. This

will make the other agents follow this leader and keep their formation on the trajectory.

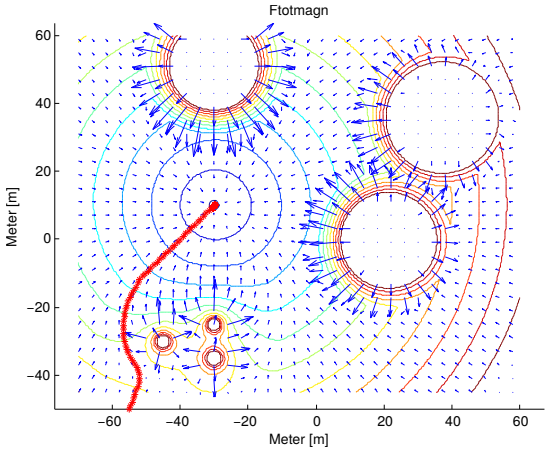


Fig. 1. Plot of one agent's trajectory with a desired position with obstacles to avoid

A plot for a total potential reference field for a single agent can be seen on figure 1. The red line made of crosses is the trajectory that one single agent will follow, if the obstacles to avoid in the plot are static. In the plot every object, either another agent or an object, are kept static. So it shows how the trajectory will be in one single time step. This will change dynamically in the next time step if the other agents also move in the potential field. The agent avoids obstacles on the way, where it can be seen that it does not get into the safety radius of the obstacles. In this specific plot is a safety radius (r) of 20m chosen, such that the distance from agent p_i (red trajectory) to any obstacle always will be larger than 20m.

The same algorithm is applied where agent i avoids other agents, agent j, $j+1$. This can be seen on figure 2.

Agent i takes a direct course toward the virtual leader but meets another agent as an obstacle. Agent i moves on the boarder of agent j with the defined safety radius and afterwards diverges from agent j towards the virtual leader. This is all done by following the lowest gradient at all times.

The gain of K_{ij} is not to be interpret from figure 1. K_{ij} is the gain to the force that attracts the agents together by minimizing the distance in between them. By doing this the agents will get faster into the desired formation. The gain K_{vl} does at some point the opposite. This gain adjusts the weighting of how fixated the agents should be to converge to the desired position. If this gain is relatively larger than K_{ij} then the agents will converge directly to their position around the

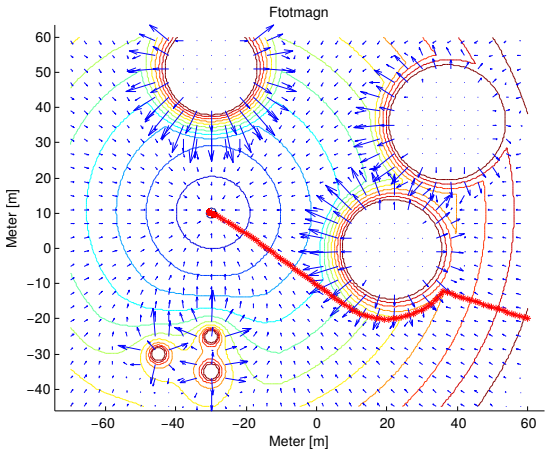


Fig. 2. Agent i avoids agent j and converges to the minima at the virtual leader

virtual leader and not converge to the desired formation on the way. This implies that the scaling between K_{vl} and K_{ij} controls if the formation should converge to the desired formation on the way to the desired position, or if agent i should only have the desired position in focus.

A. Numerical Solution

The grid in which the potential field is generated are limited with a certain resolution while simulating the agents movement. This reduces the directions of where the agents can move, which will not arise a problem on the same level when implemented in reality. In the simulation environment it reduces the resolution such that a single field in the grid contains one value of magnitude of the potential field, which makes the basis of a certain gradient to the field. The agents are following the implementation of the steepest decent. This generates a gradient towards the steepest decent, which the agent tracks. The analogy can be seen as a bowl, or sphere in this case, where a ball will converge towards the lowest point in the direction of the minimum gradient.

The method of applying the grid with magnitude of the potential field rises a problem with resolution, and therefore also a problem that makes the 'corners' of the grid around the agent to have the steepest decent. This is seen as if the agent is placed in the middle of a 3-by-3 matrix, and have eight placements around it. The placements around the agent will then be checked. The magnitude of the vector from the agent and outgoing will therefore be biggest in the corners since the distance to those are greater than the distances to the sides, up and down. This problem has been expanded

with a solution such that a certain radius in the potential field around an agent will be checked. The value at the radius around the agent can be checked, and due to the newly equal distance to every point, these will be weighted equally with respect to their value. This makes in principle the possibility to make the agent go in all directions which will be closer to the reality. When testing the two methods against each other it is clear that the first proposed with the grid structure did not have the same mobility thus not preferable in simulations though it is simpler. The first made the agents move only in the diagonals of their local placement, where the latter makes the agents able to move in a number of directions specified in the algorithm.

B. *Local Minima Problem*

A problem that can become crucial arises when two agents or two objects are within the radius of each other. This will result in a local minima in the potential field between those objects. This will create a local minima in between these agents or objects. If an agent converges toward this minima they cannot get out again. The problem can be seen on figure 3. The gains here are chosen exactly the same as in figure 1.

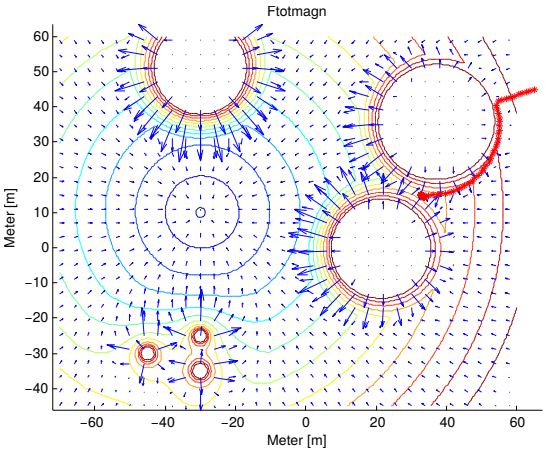


Fig. 3. An agent gets stuck due to a local minima between two other agents. The agent cannot get out of this minima unless the other two agents makes the space for the agent to pass through

The scenario on figure 3 has the following steps. The agent i moves in the direction of the steepest decent. Then it gets to the border of another agent where it cannot go through thus starts to go around this agent. The problem arises when agent i reaches another agent on the way where it now has reached a local minima. Now the steepest gradient will point at the position where the agent already is thus making it think it has reached the end point. Solutions to this problem can be formulated in different ways.

One solution could be to cluster the two objects together and instead of making their potential field individually, then combine those together and make an ellipsoid or even a circle formed obstacle of those objects. This will ensure that the local minima disappears thus not making an agent get stuck between those objects.

Another solution is to make an exception handler that can tell if agent i has reached the desired position. If it has not reached its end point, and the position is constant on the same placement, it perturbs the desired position of the agent until the direction of the steepest decent changes more than a predefined value. This will mean that the agent is out of the local minima and can continue on the trajectory. The solution of clustering the objects, that are too close, can be seen on figure 4.

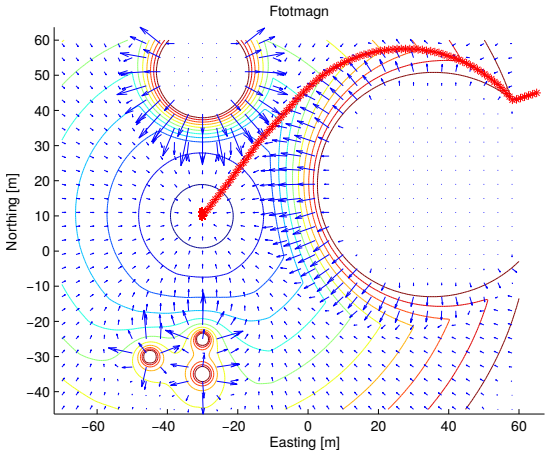


Fig. 4. An agent that before was stuck now does not get into a local minima close to the agents, as it now sees the two other agents as one larger agent.

Here the first solution is applied where the two agents, that were too close to each other, have been clustered into one, seen from the i'th agent. Now the local minima between the agents have been neglected and the i'th agent can generate its trajectory around the agents and continue to the endpoint of the potential field. The algorithm checks if the distances between the agents are lower than two times the radius r. If this is the case it means that the i'th agent cannot generate a trajectory in between these agents, which can lead to a local minima. Therefore is the agents that are too close combined into one by generating the middle point between their positions and generating a new radius. This makes a larger circle where

the two agents are in the subset. This circle will be larger depending on the wanted safety radius thus rises the need to recheck the potential field again after have generated a new combined agent. If the radius of the new agent places it close to one of the single agents, these also might need to cluster. Thus the algorithm needs to run until no distances between agents are less than two times the radius.

```
Data: clustering of agents
initialization;
if ||p_i, p_j|| < 2 · r then
    p_{j,new} ← mid point between p_i and p_j
    r_new ← calc new r for p_{j,new}
    delete p_i and p_j with p_{j,new}
end
```

Algorithm 1: This pseudo code describes how agents that are too close to each other are getting clustered and seen as one. The algorithm can also be applied for obstacles in the potential field.

The algorithm generating this combined agent can be seen in pseudo code in algorithm 1. The algorithm works by first checking every distance between the agents to see if it is lower than $2 \cdot r$. If the distance is lower, a new coordinate set needs to be calculated. The coordinates for the $\mathbf{p}_{j,new}$ is generated to the middle value of the two points

$$\mathbf{p}_{j,new} = \frac{\mathbf{p}_i + \mathbf{p}_j}{2} \tag{18}$$

and afterwards the new radius for $\mathbf{p}_{j,new}$ can be found from

$$r_{new} = \frac{||\mathbf{p}_i, \mathbf{p}_j||}{2} + r \tag{19}$$

In the end this results in that the every agent needs a magnitude and a direction of which they should move. This will be given depending on the total environment where the agents are manoeuvring, and will be assigned by the gradient vector. When applying this formation strategy a collision free movement is guaranteed which is one of the more critical criteria to be fulfilled.

V. Results

The potential field control system consists of multiple elements as seen with the flow which is illustrated on the block diagram on figure 5. The first block is the potential field generator. This is the potential field calculation to compute the magnitude of the global potential field. This information is passed to a trajectory generator, which generates the reference trajectory. This reference

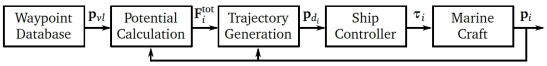


Fig. 5. Block diagram showing the iteration process of using the potential fields for computation of the input vector

trajectory is where the i'th agent needs to move. This is passed to the controller of the vessel, which then computes the input to the actuators on the vessels, which in this case is done via a linear controller, which has been developed based on a linearized version of the vessel around its nominal operating point. The position of the vessels are then fed back, both to the potential field calculation, the trajectory generation and the controller. The potential field needs to be calculated from the vessels relative position, the trajectory generation needs the position for the intermediate reference position and the controller will need it for i.e. error calculations.

Part of this flow can be computed by the i'th ship themselves, but the overlying trajectory generation needs to be handled by the virtual leader, or one leader in the formation.

The Guidance Navigation and Control (GNC) works by an array of mission specific waypoints given, usually computed from a desired area used to create a lawnmower pattern or similar area coverage algorihm. This trajectory becomes the area of interest, and is the one overlying trajectory that the virtual leader has to follow. The other ships will need to maintain their individual positions at all time steps respective to the virtual leader. Dependent of how the position is formed, the ships needs to go into formation before or during the trajectory tracking phase.

For the inner loop, a heading based LOS method can still be used, but this should be calculated for every ship, with each their reference position $\mathbf{p}_{i,r}$.

The algorithm 2 on the next page describes how the potential field strategy can be simulated, were each iteration of the while loop is a time step, which means that the control will continue until the formation has reached the way point acceptance radius of the track. This is to ensure that the formation anchor do not move forward if the ships are not properly in formation. This is analogous with the group coordination task as defined by (Thorvaldsen, 2011).

A simulation of the algorithm with the dynamics of the AAUSHIP can be seen on figure 6. The red crosses are waypoints that have been targeted as the next waypoint to reach. The line connecting

Data: track as global mission trajectory as way points
initialization;
while $m <=$ *length of track* **do**
  for *every* i*-th boat* **do**
    if *formation is ok* **then**
      if $\mathbf{p}_{vl}$ *is inside the way point acceptance radius of the track* **then**
        $m \leftarrow m + 1$;
        $\mathbf{p}_{vl} \leftarrow$ *LOS*(p_{vl} , track(m));
      end
    end
    ($\mathbf{p}_{d,i}$, $\mathbf{F}_{\text{tot},i}$) $\leftarrow$ *pathgen*(p_i , $p0_i$);
    $\mathbf{p}_{r,i} \leftarrow \mathbf{p}_i + \mathbf{F}_{\text{tot},i}$;
    $\psi_{d,i} \leftarrow$ heading from $\mathbf{p}_i$ to $\mathbf{p}_d$;
    $u_i \leftarrow$ *controller*(ψ_d);
    send input u *to ship*;
    $\mathbf{x}_i \leftarrow$ *sense ship states*;
    $\mathbf{p}_i \leftarrow$ position of $\mathbf{x}_i$;
  end
end

Algorithm 2: This pseudo code describes how the potential field is used for each boat to calculate the reference for the inner controller for every boat at every time step. Every iteration in the while loop is a time step.

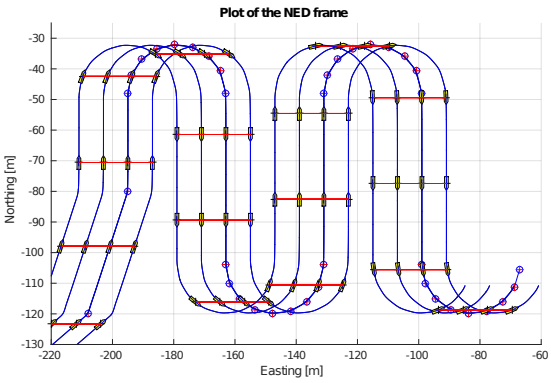


Fig. 6. Four agents are placed relative to the middle point of the formation, the virtual leader. This leader moves at the trajectory with the waypoints, but only changes to the next waypoint if the agents are in formation. The blue circles with the red plus signs are the waypoints of the virtual leader. The other paths are computed from this and the FRP.

those waypoints are the virtual leader movement, which changes position from waypoint to waypoint to generate the straight line segments for the formation to follow. Every of the four agents have a relative position placement to the virtual leader, the agents positions are p_{ij} and given as p^n_{vl}+offset and the position of the virtual leader is given as p^n_{vl}. The ships are shown as yellow ships and the ships in formation is connected with a red line.

The formation have started at position $[-250, -150]$ and has the first waypoint in $[-208, -120]$. When the agents are close to reach the waypoint at $[-208, -120]$, they ensure that every agent are in formation by waiting for the last to catch up, if needed. Then all of them are in formation with respect to the virtual leader and this changes waypoint such that the agents needs to go toward the next waypoint. This waypoint shifting continues until no more waypoints are available.

It can be seen that there is a little divergence of the ships to the line segments which is mainly due to the dynamics of the AAUSHIP. Their respective line segments are not shown on the figure, while this would make the figure confusing. Though the agents will follow a line segment from their position in the formation to the new position in the formation. This is due to the movement of the virtual leader where this only moves in straight lines, thus the following agents pursue to do the same. The trajectory the ships follow is plotted beneath the third vessel from the left from where it can be seen that this vessel is placed on top of the virtual leader, and almost makes this vessel serve as the leader of the formation. Due to the formation setup the following ships will follow the same trajectory as the leader but only in this case shifted in the easting position.

Running the same setup as used in figure 7 on the following page, but with the addition of a few objects in the way of the paths it can be seen how the ships tries to evade the obstacles. The obstacles potential field is drawn as black circles. It shall be noted the plotted boundary is not a hard boundary, but only shows the radius whereby the potential field of the obstacles can affect other objects. On figure 7 on the next page there are placed two obstacles near each other on the first north going leg of the path. Here it is seen that two ships tries to gently diverge from the paths following near the boundary to the object, while it is seen that there is a ship going “through” the two objects influence zone. As describe previous, this is due to that the penalty for going around the entire set of obstacles is larger than crossing in between the objects. Because the two objects are offset a bit and about the same distance from the path it is clearly seen that the ship follows the minimum potential field through the set.

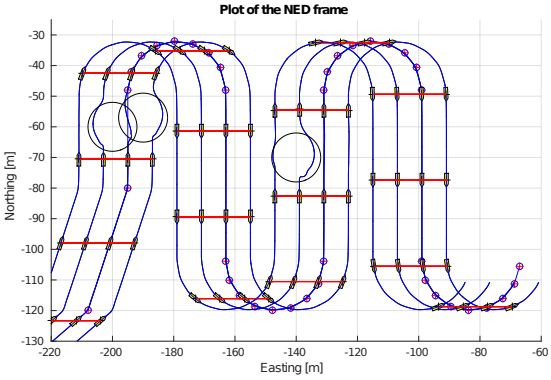


Fig. 7. Same setups as in figure 6 on the preceding page, but with obstacles positioned on the pathways.

VI. Discussion

The model and simulation has proved to work when implemented on the AAUSHIP which results in a vessel that can track a predetermined trajectory. This makes the basis for the further work of expanding the fleet of AAUSHIPS and implement the investigated control strategies on the fleet, ultimately ending up with a successful digital mapping of the seabed in the Limfjord in Denmark.

In this work the obstacles was regarded as point masses. An obvious extension of this work would be to expand the method to allow for polyhedral obstacles and then add the potential field of influence to this. This will immediately allow for better modelling of e.g. piers and other larger structures.

References

Bourgeois, B., Martinez, A., Alleman, P., Cheramie, J., and Gravley, J. (1999). Autonomous bathymetry survey system. *Oceanic Engineering, IEEE Journal of*, 24(4):414–423.

Ferreira, H., Almeida, C., Martins, A., Almeida, J., Dias, N., Dias, A., and Silva, E. (2009). Autonomous bathymetry for risk assessment with roaz robotic surface vehicle. In *OCEANS 2009 - EUROPE*, pages 1–6.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley.

Ghommem, J., Mnif, F., Poisson, G., and Derbel, N. (2007). Nonlinear formation control of a group of underactuated ships. In *OCEANS 2007 - Europe*, pages 1–8.

Ihle, I.-A. F., Jouffroy, J., and Fossen, T. I. (2005). Formation control of marine surface craft using lagrange multipliers. In *44th IEEE Conference on Decision and Control*, pages 752–758.

Paul, T., Krogstad, T., and Gravdahl, J. (2008). UAV formation flight using 3D potential field. In *16th Mediterranean Conference on Control and Automation*, pages 1240–1245.

Simetti, E., Casalino, G., Manerikar, N., Sperinde, A., Torelli, S., and Wanderlingh, F. (2015). Cooperation between autonomous underwater vehicle manipulations systems with minimal information exchange. In *OCEANS 2015 - Genova*, pages 1–6.

SNAME (1950). Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. Technical and Research Bulletin No. 1-5. Technical report, The Society of Naval Architects and Marine Engineers.

Thorvaldsen, C. (2011). Formation control of marine vessels. Master's thesis, Norwegian University of Science and Technology.

Vasilijevic, A., Buxton, B., Sharvit, J., Stilinovic, N., Nad, D., Miskovic, N., Planer, D., Hale, J., and Vukic, Z. (2015). An asv for coastal underwater archaeology: The pladypos survey of caesarea maritima, israel. In *OCEANS 2015 - Genova*, pages 1–7.

Wondergem, M. (July, 2005). Output feedback tracking of a fully actuated ship. Master's thesis, Technische Universiteit Eindhoven. [Online; accessed 3-June-2014]. DCT 2005.104. http://alexandria.tue.nl/repository/books/598447.pdf.

Robots and Art:

Interactive Art and Robotics Education Program in the Humanities

Elizabeth Ann Jochum
Department of Communication and Psychology
Aalborg University (AAU)
Aalborg, Denmark
jochum@hum.aau.dk

Lance Putnam
Department of Architecture, Design and Media Technology
Aalborg University (AAU)
Aalborg, Denmark
lp@create.aau.dk

***Abstract*—We describe the design of an undergraduate course in art and robotics that aims to integrate basic concepts of computer science, robotics and art installation for undergraduate students within the problem-based learning model. Our methodology aims to bridge the gap that separates humanities from computer science and engineering education to prepare students to address real world problems in robotics, including human-robotic interaction and HCI. Given the proliferation of interactive, systems-based art works and the continued interest in human-centered factors in robotics research (such as aesthetics, culture and perception), we believe this is an important area for education and research.**

Keywords—robotics; interactive art; education; pedagogy; human-robot interaction; HCI

I. INTRODUCTION

The problem-based learning (PBL) model at Aalborg University (AAU) provides a unique framework for developing a transdisciplinary course that prepares students to combine critical thinking and problem-solving skills with hands-on experiments and practice. The Art and Technology (ArT) undergraduate program at AAU is unique in that it offers a cross-disciplinary background in both new media art and pertinent technical subjects such as electronics, programming and rapid prototyping. Unlike traditional fine arts programs, ArT provides students with the competencies required to deeply engage with the latest technologies and to translate creative theories and approaches into practical results.

We were motivated to develop a course that exemplifies the educational vision of ArT and to provide a clear progression and integration of previous ArT courses. Multimedia Programming-Autonomous Art (MMP-AA) integrates art and technology by focusing specifically on the theoretical and practical aspects of robotic art. The course places equal emphasis on both aesthetic and technical concerns so that students may develop competencies in the creation of aesthetically engaging autonomous art works.

While there do exist some undergraduate programs that combine art and computer science education [1], many of these curriculums are centered on electronic and/or digital media arts and do not directly incorporate robotics. Robotics education remains out of reach for many students not enrolled in traditional computer science or engineering programs. Therefore, a second aim of MMP-AA was to generate interest and enthusiasm for robotics research among humanities students. The semester theme of "Narratives and Interaction" provided the context for the course and we encouraged students to incorporate this theme in their projects, for example by combining robotic art with storytelling and interactivity. Students were asked to develop original research projects that combined mobile robotics with the creation of an original artifact, performance or installation.

We incorporated several modes of evaluation into the course, including student entrance and exit surveys, video recording, and project documentation. Here, we present the data including course development and curriculum, analysis of content and student projects, evaluation, and plans for future research.

II. THE ART AND TECHNOLOGY CONTEXT

A. *Transdisciplinarity*

One of the pedagogical challenges at ArT is to select subjects that apply to a wide range of media and at the same time have direct applications within those media. In other words, students should not only be familiar with the languages and research methods across disciplines but should also be able to integrate diverse fields of knowledge to solve real world problems. The boundaries that have traditionally separated humanities research from computational research are increasingly blurred, and students should be prepared for this new landscape. Transdisciplinary thinking requires one to make abstractions on top of domain-specific knowledge. A significant obstacle to transdisciplinary teaching is to choose both the most effective abstractions and the clearest language to communicate such concepts. This is the challenge we address in MMP-AA.

B. *Multiple Languages*

The ArT curriculum involves a unique mix of theory and practice, and students take courses across a wide range of subjects including art history, sculpture, dynamic art, interactive systems programming, play and event, and

entrepreneurship. While the program offers a wide range of courses across topics, few courses formally combine aesthetic theory and practice with programming and technology. The languages of art and programming do not readily translate across disciplines, and it is usually left to the students to combine these two fields in their individual projects. The course theme of "Languages of Motion" was an effort to bring computational motion and aesthetic motion into closer alignment. Furthermore, many of the technologies taught in ArT involve software programs for screen-based media. By focusing on robotic art, we hoped to expand the students' awareness to include physics-based scenarios, encouraging students to experiment with new approaches to automated motion such as choreography.

C. Demographics

Thirteen students enrolled in MMP-AA. There were five male students and eight female students, ranging in age from 22-29 years old. All of the students were third year (fifth semester) ArT bachelor students, and had varying levels of programming experience outside of the ArT curriculum. Most students had basic programming skills including C++, HTML/CSS, and Java. Prior to MMP-AA, students had completed courses in basic electronics, materials (including structure and composition), and digital representations (including laser cutting and 3D printing). One student had some prior experience working with Lego Mindstorms, but beyond that none of the students had previous experience working with robots. Most of the students had seen some robotic art works, but their familiarity with historical and contemporary robotic art and robotics research was limited.

III. PEDADOGICAL APPROACH

This section outlines our pedagogical approach, including the teaching formats, curriculum and learning objectives.

A. Co-Teaching

Our teaching is based on a collaborative model where teaching, assignments and evaluations are developed and implemented collaboratively at the faculty level. Our appointments are at the Faculty of Humanities and the Faculty of Engineering and Science respectively, which further strengthens the transdisciplinary foundation of the course and demonstrates to the students ways to foster fruitful collaboration between humanities, computer science and engineering. We also decided that both professors should be present at all lessons in order to facilitate communication across disciplines.

B. Lecture/Workshop Format

We deliberately structured the course to balance the time between lectures and hands-on workshops as well as aesthetic and technical considerations. The course consisted of eight lessons in total with each lesson lasting two hours. Lessons typically followed the structure of a lecture on a specific topic followed immediately by a hands-on learning exercise or discussion. Wherever possible, individual lessons incorporated both technical and aesthetic topics.

C. Curriculum

The overall curriculum aimed to teach students how to design, program and execute a computer-controlled work of art based on computational models and theories in robotic art. The lesson topics were

- Origin and development of robotic art
- Robot communications
- Languages of motion I (periodic motion and random walks)
- Languages of motion II (kinesics, flocking, emergent behavior)
- Markov chains and "Acting for Robots"
- Workshop on designing and constructing robot bodies and mechanisms with a visiting robot artist

The remaining two lessons were in-class presentations of the midterms and final projects, where the students were asked to present their functioning prototypes and answer questions about their projects.

Students were provided with robots to experiment with (the mobile Arduino robot and Sphero mobile robot), but were also given the opportunity to develop their own design or robotic prototypes. As the course is an upper-level undergraduate course, a prerequisite for enrollment is imperative and object-oriented programming (e.g., C++ or Java).

D. Project-Based Assignments

Following the AAU PBL model, we used project-based assignments to encourage students to engage in open-ended, play-based experimentation and inquiry. We deliberately refrained from placing too many constraints on the midterm and final projects, but rather encouraged students to be guided by their own curiosity given the constraints of the relatively simple robotic platforms. Our hope was that this would result in works that were relevant to the students' experience, skill level and general artistic interest.

A necessary component for developing interactive, robotic art works is to investigate, anticipate or understand human response and reactions to the exhibited art works [2]. These topics are relevant to the study of robot design and HRI research [3]. In their projects, students were expected to apply theoretical foundations from art and performance and to explore the aesthetic potential of motions. From this we can learn how to approach concepts such as autonomy and interactivity on an experiential and aesthetic level.

There were two assignments in the course: 1) a midterm sketch/study and one-page written summary outlining the research project and 2) the completion of a group-based mini-project incorporating computer-controlled robotics. The mini-project was presented in class, and the functioning prototype was accompanied by a written report and oral presentation summarizing the project, method, approach and conclusions.

Students were allowed to form their own project groups and we made no effort to balance the groups in terms of skill sets, e.g., programming. Rather, we preferred the students form groups based on shared interest in a topic or specific robotic platform. We speculate that this approach worked well because of multiple factors: all students had basic programming skills, the incorporation of group-based class exercises, and students were from the same study program (ArT). If the student population was more diverse, it may have been necessary to structure the group formation more. However, the group-based class exercises may have the additional benefit of getting students acquainted with one another.

IV. Implementation

In this section, we present a detailed description of how certain aspects of the course were implemented. We discuss the selected teaching platforms, programming languages, and algorithms and describe the robot artist workshop and student projects.

A. Theme: Languages of Motion

The underlying theme of MMP-AA was "Languages of Motion." Our goal was to introduce elementary algorithms of motion and how those motions, individually or combined into choreography, can create an aesthetic response in the viewer. One goal of robotic or interactive systems-based art is to evoke emotional responses through planned motions executed by otherwise inanimate objects. Other media such as computer animation and sound have their own specific languages of motion, and we believe there are general principles within languages that can be applied directly to robotics. For example, random paths are useful for exploration and for masking artificial movements and periodic functions can be composed to synthesize complex, structured patterns. Understanding the link between the languages of motion and aesthetic and cognitive responses may open up new ways of thinking about design and interactivity for robotics research [4].

B. Platforms

As the intention of the course was to focus on robot communications and languages of motion, we wanted to use ready-made mobile robots that supported open, standard protocols. We ruled out use of drones and other 3D-navigable robots early on as we felt these would present too many technical challenges as well as safety issues for an introductory course. The robot platforms chosen were the Orbotix Sphero 2.0 [5] (Figure 1) and Arduino Robot [6] (Figure 2). The Sphero is a Bluetooth-controlled, hermetically-sealed ball that can be commanded to move and turn relative to an initial reference frame. It also has a controllable colored light and can stream numerous sensor data back to the client including its position, accelerometer, gyro and IMU readings. The Arduino Robot is essentially a programmable two-wheeled cart. It also provides buttons and a dial for user input and a display screen and speaker for feedback. A major advantage of the Arduino Robot over the Sphero is that additional sensors, such as a video camera, can be easily attached to the robot. The Sphero, on the other hand, can be used on a wider variety of terrains, including water, and may be easier to use as an actuator.

Figure 1. The Orbotix Sphero 2.0 remote-controlled mobile robot. Shown are its outer appearance (left) and inner workings (right) [7].

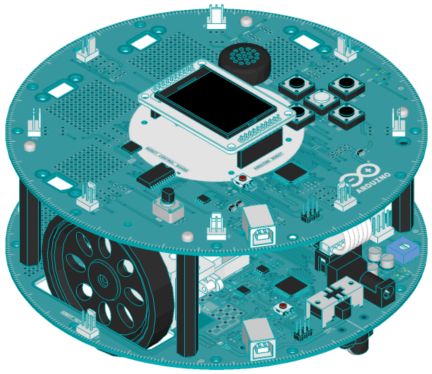


Figure 2. Illustration of the Arduino Robot [8]: a programmable two-wheeled cart robot.

C. Programming

It was important to use the same programming language for both robot platforms to maximize the possibility of code reuse. We selected C++ as the Arduino platform and much of the robotics community use it. For the Sphero, we used the AlloSystem multimedia toolkit [9] and custom-made Bluetooth client and Sphero packet generating and parsing classes. We considered using the Robot Operating System (ROS) [10] but at the moment only Linux/Ubuntu is officially supported and most of our students use either Windows or OS X.

The algorithms we selected were meant to teach the students about elementary types of motions and how to combine those motions. For the elementary motions, we chose circular motion, random walks, and flocking [11] as the basis for periodic, random, and force-based motions, respectively. To combine motions two main principles were introduced: motion

superposition, i.e. weighted vector summation of elementary motions, and time-based sequencing. Sequencing, in turn, was divided into two main approaches: linear and random. Linear sequencing consisted of a series of control commands separated by delta times while random sequencing was implemented via Markov chains with fixed delta times between state transitions.

In-class assignments were conducted using the Sphero and generally followed a progression from simple to more complex tasks. For example, in one lesson, the first task was to change the color of the Sphero's light. This progressed to moving the Sphero in a line and then stopping, and then onto circular and random motions. The last exercise was to simultaneously control the motion and color of the Sphero according to some pattern.

D. Workshop with Guest Artist

One special feature of the course was a one-day workshop taught by a visiting robot artist. The artist also gave a corresponding lecture on his own art works and design methodology. In the workshop, students constructed simple mechanisms (gears, belts, and pulleys) with flexible and cheap materials such as cardboard boxes, straws and rubber bands. In the second part of the workshop students were taught how to combine the mechanisms with Arduino controllers. Unfortunately, we did not allow enough time for the students to implement more advanced mechanisms, but each student succeeded at designing and constructing a functioning prototype. After working with rigid bodies with limited degrees of freedom and movement primitives (i.e., the Sphero and Arduino robots), the students enjoyed the chance to work with flexible materials and to design their own robots. Unfortunately, the workshop came too late in the semester for students to incorporate this knowledge into their projects. In hindsight, we believe the students would have benefited from the workshop earlier in the semester before they commenced work on their final projects. This would have introduced the possibility of building their own robots or combining the Arduino or Sphero with custom-built robots or parts.

E. Project Descriptions

Thirteen students worked in five groups. The projects reflected a wide range of topics, some more scientific and others more artistic in their approach and methodology.

1. *Color/Gesture Mapping*. This project aimed to map color changes to the Sphero based on human-robot interaction, using the robot's orientation and altered motion trajectories (human input) to control the color of the Sphero.

2. *Sphero Dancers*. This project abstracted choreographic structures from human dancers to generate motion trajectories for three Sphero "dancers" to create dramatic tension and evoke an emotional response in the viewers.

3. *FlirtyBot*. Using an Arduino Robot, this project built a social robot with a distinct personality, using simple interactions such as dialogue, sound, and simple movement patterns in response to input from the human user.
4. *Mind-controlled Sphero*. This project combined the Sphero platform with the Emotiv EPOC biopotential neuron headset [12] to simulate telekinesis. The custom software (authored by the students) enabled users to control the motion of the Sphero in real time using only facial expressions.
5. *"Kiwi" Interactive Narrative*. This robotic performance combined preprogrammed and tele-operated Spheros with live shadow puppetry, a musical score and narration based on the Kiwi bird, a flightless bird from New Zealand. The students developed a storyboard and applied principles of narratives and interaction to generate an original performance (Figure 3).

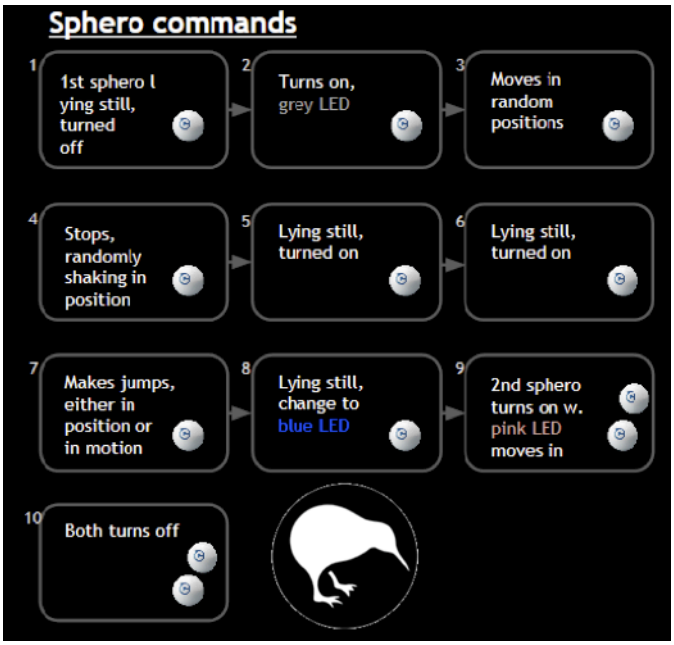


Figure 3. Storyboard of the *"Kiwi" Interactive Narrative* student project (© Daniel K. Kittow and Martin L. Nielsen). This was a live performance that combined preprogrammed and tele-operated Spheros with live shadow puppetry, a musical score, and narration based on the flightless bird from New Zealand.

V. Analysis and Reflections

This section presents analysis and reflections of the course based on our observations and experiences in the classroom, evaluation of student projects, and student feedback.

A. Entrance and Exit Surveys

At the beginning of the course, students were asked to complete an entrance survey that included questions about prior programming experience and experience with robots and

robotic art, as well as their expectations for the course. All thirteen students completed the intake survey. On the last day students were asked to complete an exit survey, which included questions on the course content, format, projects and challenges they faced. Eight students completed the exit survey.

Based on the entrance surveys, many students were excited about the possibility of moving from screen-based media to physical systems ("reaching beyond the screen"), and some thought that working with robots would deepen their understanding and knowledge of programming. Nearly all of the students expressed enthusiasm about the "hands-on" nature of the course, and were eager to apply their skills to more advanced, interactive art works that involved human-robot interaction.

The exit surveys show that students' initial expectations were largely met, and the students generally agreed that the course struck a good balance between theory and practice. The students appreciated the flexibility to develop their own projects as well as the opportunity to apply theory to practice in the in-class workshops and assignments. Students responded positively to the challenge of working in the physical world, including discovering the limits and challenges of working in physics-based scenarios.

Working with physical systems challenged the students in new and unexpected ways:

> "Robots sometimes react to the physical world in unpredictable ways. They might also be preprogrammed with a behavior that clashes with something we want to make them do."

> "A lot of the stuff we have been programming have had screen-based outputs. It was nice and quite interesting to be able to move into the physical world, and discover the limits thereof. Suddenly we had to deal with a whole new set of issues and problems along the way."

> "Considering physics, robots can be quite unreliable."

Many of the groups made an effort to apply theories of narrative and interaction to their projects:

> "In conclusion of the design process, many lessons have been learnt, both practical and aesthetic in creating robotic based art. The group has discovered the power of motion in narrative storytelling and how robots can seem more lifelike and organic with focus on movement rather than on form. […] This movement can be implemented like an actor in a theatrical performance."

B. Platforms

Out of the five student projects, four projects involved use of the Sphero and only one project used the Arduino Robot. One explanation for this preference may be because the Sphero was used as the primary teaching platform, but also possibly due to its simplistic, yet open-ended nature.

One of the problems we encountered with the Sphero is that it is difficult to get it to move accurately along a predefined trajectory over time. This stems partially from its lack of ability to report its absolute position and orientation. Our solution was to run a calibration program before each performance and align the Spheros through visual inspection. Other factors such as slipping, changes in momentum, or dropped control packets can also introduce errors in the motion trajectory and cannot be prevented through calibration. A workaround to these problems, which we did not test, may be to use some form of feedback control.

C. Algorithms

Overall, the students responded well to the algorithms that were taught. Most notably, several students expressed a deep interest in the theory of Markov chains and requested a more in-depth study of them.

The application of the presented algorithms to the project work varied. We found that students tended to pick one algorithm and explore its possibilities rather than combining multiple algorithms. For example, one project exclusively used circular motions, another used primarily random walks, and a third utilized a Markov chain consisting of move/turn commands. A possible explanation for this, supported by the project reports, is that the students encountered unexpected technical challenges along way and simply did not have time to progress beyond the basic algorithms. None of the projects made use of flocking which could be due either to the small number of project groups or the demand of the algorithm for more advanced control of multiple robots.

D. Co-teaching

Students were generally enthusiastic about the co-teaching model, and suggested that it was helpful to have both professors on-hand to answer questions and to maintain the balance between artistic and technical discourses. Personally, we feel that our distinct areas of expertise were complementary and that our mutual interest and engagement in the topic of autonomous art was a good model for the students.

VI. FUTURE WORK

Overall the students and professors were satisfied with the course, and many students expressed interest in continuing work on their projects. Given their interest and the strong artistic and technical merit of the projects, we plan to develop a second course that builds on this framework. Our objective is to develop the skills and competencies the students acquired in MMP-AA by expanding on the knowledge and practice of robotic art.

In the next course, we will introduce basic feedback control using video or other sensor data to address some of the problems with obtaining precise trajectories. We plan to introduce new algorithms that can be used to generate motion such as gradient fields, cellular automata, and chaotic or other non-linear equations. We are also interested in conducting a hands-on workshop in 3D-printed bodies and mechanisms that will build on the students' previous coursework in rapid prototyping. Overall, we hope that students will be able to refine their previous projects and at the same time experiment with designing their own robots and news forms of expressive motion.

ACKNOWLEDGMENT

We thank Mikael Vetner, Falk Heinrich, and Ståle Stenslie for their commitment to this course and to research-based teaching. We thank the members of the RELATE research group for their early advice and support of this project.

References

[1] H. J. Kim et al., "Artbotics: Community-based collaborative art and technology education," in *ACM SIGGRAPH 2007 Educators Program*, ACM, New York, NY, USA, 2007.

[2] O. Bown et al., "The Machine as Autonomous Performer," in *Interactive Experience in the Digital Age*. L Candy and S. Ferguson, Eds. Springer, 2014, pp. 75-90.

[3] K. Dautenhahn, "Methodology & Themes of Human-Robot Interaction: A Growing Research Field," *Int. Journal of Advanced Robotic Systems*, vol. 4, no. 1, pp. 103-108, 2007.

[4] G. Hoffman and W. Ju, "Designing Robots with Movement in Mind," *Journal of Human-Robot Interaction*, vol. 3, no. 1, pp. 89-122, 2007.

[5] Sphero 2.0, (2014). [Online]. Available: http://www.sphero.com/sphero

[6] Arduino Robot, (2014). [Online]. Available: http://arduino.cc/en/Main/Robot

[7] Sphero 2.0, (2014). [Image]. Available: http://www.sphero.com/sphero

[8] Arduino Robot, (2014). [Image]. Available: http://arduino.cc/en/Main/Robot

[9] Git Hub, Inc. (2014). AlloSphere-Research-Group/AlloSystem. [Online]. Available: https://github.com/AlloSphere-Research-Group/AlloSystem

[10] M. Quigley et al., "ROS: An open-source Robot Operating System," *IRCA Workshop on Open Source Software*, vol. 3, no. 3.2, 2009.

[11] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphics*, vol. 21, no. 4, pp. 25-34, 1987.

[12] Emotiv, (2014). Emotiv EPOC/EPOC+ [Online]. Available: https://emotiv.com/epoc.php

Safety Critical Java for Robotics Programming

Bent Thomsen*, Kasper Søe Luckow*[1], Thomas Bøgholm*, Lone Leth Thomsen*,
Stephan Erbs Korsholm†
*Department of Computer Science, Aalborg University
{bt,luckow,boegholm,lone}@cs.aau.dk
†VIA University College
sek@viauc.dk

***Abstract*—This paper introduces Safety Critical Java (SCJ) and argues its readiness for robotics programming. We give an overview of the work done at Aalborg University and elsewhere on SCJl, some of its implementations in the form of the JOP, FijiVM and HVM and some of the tools, especially WCA, TetaSARTS tool suite and SymRT, allowing programmers to analyze their SCJ applications for correct time behaviour.**

Since its inception in 1995, Java has become one of the most popular programming languages. With its *write once - run anywhere* approach, it was rapidly propelled into mainstream computing. The robotics community quickly started to appreciate the language, and today Java is very popular in robotics programming with several academic and commercial frameworks [48], [50], and a very active community [21].

It is well known that Java, in its standard edition, is unsuited for hard real-time applications, mainly due to insufficient thread semantics and the use of unpredictable garbage collection algorithms in the implementation of the Java Virtual Machine (JVM). Clearly some areas of robotics have hard real-time requirements. Here ad-hoc approaches or approaches based on external code, often written in C running on real-time operating systems such as RT_PREEMPT and accessed via the Java Native Interface (JNI), have prevailed. Although impressive systems, such as the programming of Humanoid Robots [57], have been implemented this way, it requires the programmer to juggle two programming languages and run-time environments with very different and sometimes conflicting semantics.

To accommodate hard real-time applications in Java, much research has been devoted to developing appropriate programming models in Java facilitating real-time systems development. The first such approach was conducted under the very first Java Community Process (JSR001) and led to the Real-Time Specification for Java (RTSJ) [13]. RTSJ introduces high resolution clocks, the notion of *NoHeapRealTimeThread* and scoped memory, allowing application developers to program threads that have no interaction with the heap to avoid problems with the garbage collector during real-time execution. As reported in [58] RTSJ has been used in the M2V2 humanoid robot [46] and there are several examples of RTSJ being used for industrial robotics [60], [49].

However, RTSJ is rather resource demanding and the original implementation from SUN required a high end Sparc processor running a Solaris 10 operating system, though newer versions from Oracle, as well as commercial versions of RTSJ from other vendors, have reduced the resource demands considerably [1], [3]. RTSJ is fairly dynamic in nature, leaving checks of consistent use of scoped memory and real-time properties to run-time analysis. Thus RTSJ is not suitable for static verification of hard real-time properties, and in many cases it is not suitable for small embedded systems. Therefore Java has so far been absent in the area of robotics on small embedded platforms used for hard real-time systems requiring certification of time properties.

In recent years the Java community, through JSR 302, has made tremendous progress, and an important step has been taken with the upcoming Safety Critical Java (SCJ) standard [36] making Java a viable choice for development of embedded hard real-time systems. The SCJ standard is an extended subset of RTSJ; it similarly contains high-resolution clocks and timers, the idea of *NoHeapRealTimeThread* and (a simplified version of) the scoped memory model to remove the need for a garbage collector. In addition, SCJ has a sufficiently tight thread semantics and a programming model based on tasks grouped in missions, contributing to facilitating the task of verifying real-time properties. A mission encapsulates a specific functionality or phase in the lifetime of the real-time system as a set of schedulable entities. For instance, a flight-control system may be composed of take-off, cruising, and landing, each of which can be assigned a dedicated mission. A schedulable entity handles a specific functionality and has release parameters describing the release pattern and temporal scope in terms of release time, deadline, etc. The release pattern is either periodic or aperiodic following a classic control system structure [15].

There are now a number of JVM implementations supporting the SCJ programming model including FijiVM [43], The Hardware-near Virtual Machine (HVM) [59], [32], [38] and the Java Optimized Processor (JOP) [51]. FijiVM has sufficiently low memory demands and is applicable for embedded systems. It does, however, require a POSIX-like OS. HVM is a lean JVM implementation directed towards use in resource-constrained embedded devices with as low as 256 KB ROM and 20 KB RAM. It features both iterative interpretation, Java-to-C compilation, and a hybrid of the two [32]. The HVM is self-contained and does not rely on an OS. It runs on popular robotics platforms such as Atmel AVR ATmega2560 microcontroller, Arduino and Lego EV3. The JOP is a JVM implemented in hardware using an FPGA. Both JOP and the HVM support the notion of Hardware Objects [33], an object-

[1]Now at Carnegie Mellon University, Silicon Valley, USA.

oriented abstraction of low-level hardware devices, such as I/O registers and interrupts, which can be handled directly from Java space.

Rigorous verification is essential for safety critical embedded hard real-time systems needing to comply with tight timing constraints, especially for systems needing to comply with standards such as DO-178C, ISO-26262, IEC-61508 and EN-50128 (applying to systems operating in safety-critical domains such as avionics and automotive). Of special interest is the verification of the system being *schedulable* i.e. verifying that all real-time tasks, under the conditions of the employed scheduling policy, finish before their respective deadlines in all circumstances. Response time analysis [15] is a traditional approach for concluding on schedulability; the response times of the real-time tasks are calculated using Worst Case Execution Time (WCET) and blocking times, and the system is schedulable if the response times are less than the task deadlines. Traditionally WCET analysis has been done by measurement, which provides an unsafe estimation approach. In recent years, a number of tools for timing analyses of systems written in Java have emerged; WCA [54] for WCET analysis on JOP, TetaJ [27] for WCET analysis on HVM, SARTS [12] for schedulability analysis on JOP, and TETASARTS [41] for schedulability and other time analyses on JOP and HVM. All the mentioned tools follow a strategy of translating SCJ programs into networks of Timed Automata, and timing analysis is then formulated in terms of an appropriate logic, e.g. Timed Computation Tree Logic (TCTL), and subjected to model checking using UPPAAL [5]. The SARTS and TETASARTS tools take the interactions between tasks into account during schedulability analysis, therefore systems deemed unschedulable using traditional response time analysis may be deemed schedulable using model checking as demonstrated in [12]. TETASARTS allows the programmer to write the program in a platform independent way, using the SCJ profile, and then analyse whether it can be scheduled on the particular platform. Hence, this enables a *write once - run wherever possible* development approach. The tool also facilitates energy savings on processors, such as JOP and Atmel's AVR, since TETASARTS allows the clock frequency of the target hardware to be set prior to the analysis. By adjusting this parameter, the lowest clock frequency, at which the system is still schedulable, can be found as demonstrated in [37].

Both JOP and HVM have been used in prototypical control systems such as the Minepump control system [27], the Real-Time Sorting Machine (RTSM) [12] and many student projects. The JOP has been used in industrial applications such as the Kippfahrleitung system for the Austrian Railways controlling up to 15 independent actuators [52].

In our opinion Safety Critical Java is now ready for more widespread use in robotics. It is supported by several academic and commercial implementations, and a number a tools can assist in the development of applications ensuring that they operate in time predictable manners.

The rest of this paper is organized as follows. In section I we describe the SCJ programming model. In section II we give an overview of three execution platforms for SCJ programs. In section III we describe a number of tools which can be used for analyzing timing properties of SCJ applications. Finally in section IV we draw conclusions and elaborate on future work.

I. THE SCJ REAL-TIME PROGRAMMING MODEL

Safety critical applications have different complexity levels. To cater for this the SCJ programming model is based on tasks grouped in missions, where a mission encapsulates a specific functionality or phase in the lifetime of the real-time system as a set of schedulable entities. The SCJ specification lets developers tailor the capabilities of the platform to the needs of the application through three compliance levels. The first level, Level 0, provides a simple, frame-based cyclic executive model which is single threaded with a single mission. Level 1 extends this model with multi-threading via periodic and aperiodic event handlers, multiple missions, and a fixed-priority preemptive scheduler (FPS). Level 2 lifts restrictions on threads and supports nested missions. The development of SCJ applications at Level 0 is well described in [44]. In the remainder of this section we will focus on Level 1.

A. Missions

A mission encapsulates a specific functionality or phase in the lifetime of the real-time system as a set of schedulable entities. For instance, a flight-control system may be composed of take-off, cruising, and landing each of which can be assigned a dedicated mission. A schedulable entity handles a specific functionality and has release parameters describing the release pattern and temporal scope e.g. release time and deadline. The release pattern is either periodic or aperiodic.

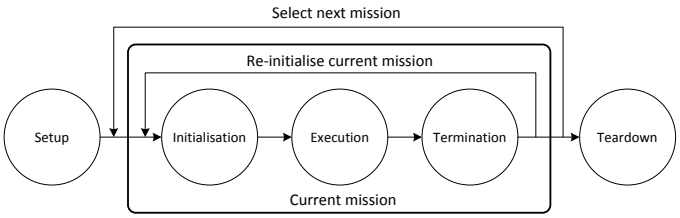


Fig. 1: Overview of the mission concept [38].

The mission concept is depicted in Figure 1 and contains five phases;

Setup where the mission objects are allocated. This is done during start-up of the system and is not considered time-critical.

Initialisation where all object allocations related to the mission or to the entire applications are performed.

Execution during which all application logic is executed and schedulable entities are set for execution according to a pre-emptive priority scheduler. This phase is time-critical.

Cleanup is entered if the mission terminates and is used for completing the execution of all schedulable entities as well as performing cleanup-related functionality. After this phase, the same mission may be restarted, a new is selected, or the Teardown phase is entered.

Teardown is the final phase in the lifetime of the application and comprises deallocation of objects and release of locks etc. This phase is not time-critical.

A *mission sequencer* is used for governing the order of the mission objects and can be customised to the application.

B. Memory Model

SCJ introduces a memory model based on the concept of *scoped memory* from the RTSJ, which circumvents the use of a garbage collected heap to ease verification of SCJ systems. The SCJ memory model is shown in Figure 2 and introduces three levels of memories;

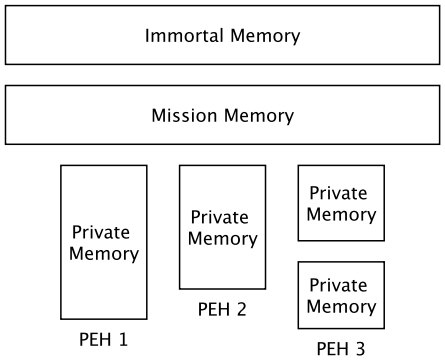


Fig. 2: The memory model in SCJ [38].

Private memory which is associated with each real-time event handler, which can be periodic (PEH) or aperiodic (APEH). The private memory exists for the entire duration of the handler. Upon task finish, the memory area is reset.

Mission memory is associated with every mission of the system and as such manages the memories of all real-time handlers part of the mission as well as objects that are shared among the handlers. When the mission completes execution, the mission memory is reset.

Immortal memory is the memory area that exists for the lifetime of the system.

Dynamic class loading is outside the scope of the SCJ specification. Hence, it is not necessary to reason about classes potentially being loaded over a network which would complicate timing analysis significantly. Furthermore, finalizers will not be executed and we make the assumption that Java Bytecode verification of class files has been done prior to the time-critical phase. The Predictable Java profile [10], being an alternative Java profile for hard real-time systems development, does allow the use of finalizers and [11] has demonstrated that timing analysis is possible. This may be important for systems written in a mix of Java and C++.

C. An SCJ Application

In this subsection we describe a hard real-time system implemented in SCJ based on a reduced version of the classical text-book control system example of a mine pump [16]. The purpose of the mine pump is to monitor a number of environmental properties in a mine to safely remove excess water using a water pump. It consists of two environmental properties being monitored: the water level in the mine and the methane level. When the water level rises to a predetermined level, the water pump is started, and when the water level drops to another predetermined level, the water pump is stopped. The water pump must not run if the methane levels exceed safe levels. These functionalities have temporal requirements stating the reaction times of the system required for safe operation such as timely stopping the water pump whenever a critical level of methane is reached.

Listing 1 shows the periodic event handler adhering to the SCJ profile.

```
1 PeriodicMethaneDetection
2   methaneDetection =
3     new PeriodicMethaneDetection(
4       new PriorityParameters(METHANE_DETECTION_PRIORITY
            ),
5       new PeriodicParameters(
6           new RelativeTime(0, 0),
7           new RelativeTime(PERIODIC_GAS_PERIOD, 0)),
8       new StorageParameters(
9           SCOPED_MEMORY_BACKING_STORE_SIZE,
10          NATIVE_STACK_SIZE,
11          JAVA_STACK_SIZE),
12      methaneSensor,
13      waterpumpActuator);
14
15 methaneDetection.register();
```

Listing 1: An SCJ handler for methane level [8].

An SCJ periodic event handler has a number of parameters: since the SCJ profile level 1 uses an FPS scheduler, evidently a priority must be specified. Furthermore, a release parameter specifies the start time, the relative initial time for the first release of the handler, and a further relative time gives the period. An instance of `StorageParameters` expresses memory-related constraints for the handler. The objects `methaneSensor` and `waterpumpActuator` are interfaces to a sensor and an actuator. The sensor observes the current methane level and the actuator starts and stops the water pump. When a handler instance has been created, it is set for being scheduled when the `register()` method is invoked.

```
1 public void handleEvent() {
2   waterpumpActuator.emergencyStop(
3     methaneSensor.isCriticalMethaneLevelReached()
4   );
5 }
```

Listing 2: Detecting the methane level [8].

Listing 2 shows the event handling method of the periodic event handler `PeriodicMethaneDetection`.

The actual prototype consists of two parts: the physical plant and the control software. Lego is used to construct the physical plant together with Lego NXT sensors and actuators connected to a JOP board or a board with an AVR ATmega2560. The control software comprises two periodic and two sporadic real-time tasks written in Java. The periodic tasks are responsible for monitoring the methane and water levels. The sporadic tasks are released whenever either the low or the high level has been reached. More details can be found in [8].

II. Real-Time Execution Platforms

The SCJ programming model provides a structuring framework for applications with hard-real-time requirements. Next such applications need an execution platform. For applications written in C this is usually a hardware processor. However, Java applications are typically translated into Java Bytecodes which are then either interpreted or further translated into native code before execution, also called ahead-of-time (AOT) execution or during, also called just-in-time (JIT) execution. This approach entails a time predictable implementation of each Java Bytecode. In this section we will describe the Java Optimized Processor (JOP) [51], the FijiVM ahead-of-time compiler for Java Bytecode programs [43] and The Hardware near Virtual Machine (HVM) [59], [32]. There are commercial implementations of the JVM supporting the SCJ programming model. These include the FijiVM, JamaicaVM [1] and PicoPERC [42].

A. Java Optimized Processor

The simplest way to ensure a time predictable execution of each Jave Bytecode is to implement the JVM in hardware. This is the approach taken by the JOP [51]. The JOP is implemented on an FPGA (Altera Cyclone EP1C6Q240 or EP1C12Q240). The JOP has its own micro code instruction set with most Java Bytecodes having a one-to-one mapping. However, some Java Bytecodes are more complex and are implemented as a sequences of JOP micro codes, some are even implemented in Java. However, the end result is that for each Java Bytecode its execution can be bound and its WCET be determined.

Important for WCET analysis of programs executing on the JOP is that the JOP does not feature data caches, but it features a method cash and its use must be taken into account for tight bounds of WCET.

The JOP is usually hosted on a board which comes in two configurations; The Baseio, which provides a complete Java Processor system with Internet connection for JOP with network connection via a CS8900 Ethernet controller with RJ 45 connector. Java sources for CS8900 driver and a simple TCP/IP stack are available for JOP. The Simpexp, which is a cheap IO extension to get started with JOP, contains a linear 3.3V regulator and serial connector. The JOP can interface to sensors and motors of the LEGO Mindstorms series and thus the JOP can substitute the LEGO RCX.

B. FijiVM

FijiVM is an ahead-of-time compiler for Java Bytecode programs [43]. FijiVM has sufficiently low memory demands and is applicable for embedded systems. The FijiVM parses Bytecodes in the Java 1.6 or earlier format and generates ANSI C code. The generated C code is then automatically passed to a C compiler, typically GCC, for the target platform. The FijiVM compiler does extensive high-level optimizations prior to generating C, as the C compiler cannot perform some high-level optimizations as effectively on C code generated from Java as a Java-optimized compiler could. Such optimizations include control flow optimizations, such as Intra-procedural and whole-program type propagation; Devirtualization, turning virtual calls into direct calls and Virtualization, turning interface calls into virtual calls; exception optimizations; Null pointer check elimination; Array bounds check removal; locking and memory optimizations; Inlining; Copy propagation; Constant folding and Tail duplication.

The FijiVM generates a stand-alone executable which, however, is dependent on libraries that are standard on POSIX-like OSs, such as Linux (libc, libpthread, and libm). The FijiVM has support for embedded processors such as ARM and ERC32 or more powerful processors such as PowerPC and x86/x86_64.

C. The Hardware near Virtual Machine

The HVM [59], [32] is a lean JVM implementation intended for use in resource-constrained embedded devices with as low as 256 KB ROM and 20 KB RAM. It features both iterative interpretation, Java-to-C compilation (AOT), and a hybrid of the two.

The HVM employs *JVM specialisation*; a JVM is produced specifically for hosting the Java Bytecode program of a given application. This is done using the ICECAP-TOOLS Eclipse-plugin, which analyzes the Java Bytecode program and produces an executable for the target platform. The analyzes and transformations can be extended, and it incorporates a number of static analyzes for improving performance of the JVM and for reducing its size. This includes receiver-type analysis for potentially devirtualising method calls and intelligent class linking which computes a conservative set of classes and methods that are used in the application. Only this set will be embedded in the final HVM executable. It also conservatively estimates the set of Java Bytecodes that will actually be used. Those that are not, are omitted from the final executable.

The HVM is self-contained and does not rely on the presence of an OS or a C standard library. The overall structure of an SCJ application running on top of the HVM using the accompanying SCJ implementation (HVM-SCJ [59]) as well as the ICECAP-TOOLS SDK is shown in Figure 3. Porting the HVM to new target platforms (including SCJ support) is a matter of implementing the *HW Interface* and the *VM Interface*.

The HVM implements SCJ level 0, 1 and 2 [62]. It has support for multicore[1] and Hardware Objects [33], an object-oriented abstraction of low-level hardware devices such as interrupts and I/O registers which can be handled from Java space. A related feature is *native variables*, which allow for access to certain variables in the JVM from Java space.

The HVM has been ported to Atmel AVR ATmega2560 microcontroller, Arduino and Lego EV3.

III. Timing Analysis Tools

Rigorous verification is essential for safety critical embedded hard real-time systems needing to comply with tight timing constraints. Of special interest, is the verification of the system being *schedulable*, i.e. verifying that all real-time tasks under the conditions of the employed scheduling policy, finish before their respective deadlines in all circumstances. The Worst and Best Case Execution Time (WCET and BCET) often play an integral role in this relation – especially the former,

[1] https://github.com/zs673/Multiprocessor-icecap-SCJ-RTE

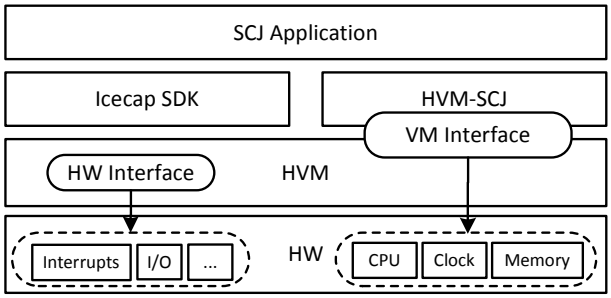


Fig. 3: Constituents of an SCJ application on HVM [38].

which has applications in traditional methods for verification of schedulability such as response time analysis [15].

For systems written in C or rather a suitable subset of C, there are many such analysis tools, both academic and commercial [4], [34], [20], [45], [17], [23], [29], [47], [26]. These tools vary in the platforms they support, in the way they analyze programs and which restrictions they place on the analyzed programs, but as stated in [61] "To avoid having to solve the halting problem, all programs under analysis must be known to terminate. Loops need bounded iteration counts and recursion needs bounded depth". The amount of required annotations is reduced by analysis, such as automatic loop-bound and array-call recognition.

Analyzing timing properties for Java programs is challenging primarily due to the fact that Java is usually translated to Java Bytecode, which is then interpreted by a JVM or further translated into native code, sometimes via a compilation to C. This level of indirection complicates formal analysis as both program and JVM have to be taken into account for a given hardware platform; some of this complexity can, however, be reduced by a hardware implementation of the JVM such as JOP.

Recently a number of analysis tools have been developed including WCA [54], SARTS [12], TetaJ [27], TETASARTS [41] and SymRT [40]. In this section we give an overview of three tools; WCA, TETASARTS and SymRT.

A. WCA

WCET Analyzer (WCA) [55], [30] is a static code analysis tool for conducting WCET analysis of Java Bytecode executed on the JOP. As described earlier the JOP is a hardware implementation of the JVM which facilitates known execution times of each Java Bytecode. The relative simplicity and predictability of the JOP architecture and, in particular, the use of a method cache instead of more general cache disciplines, makes it relatively easy to perform precise WCET analysis. WCA employs two distinct strategies for WCET analysis; one is the Implicit Path Enumeration Technique (IPET) [35] and the other models the real-time application using timed automata in the verification tool UPPAAL [5]. The rationale behind supporting two different strategies is that the two represent a trade off between estimation time and precision. In WCA, the IPET strategy yields WCET estimates relatively fast, while the model-based strategy results in more precise estimates at the cost of a relatively long verification time. The precise WCET estimate is a consequence of the model representing the detailed behaviour of the system, especially the cache model. Common to both WCET estimation strategies is the Control-Flow Graph (CFG) of the application which is constructed by consulting the Java class files using the Byte Code Engineering Library. For the IPET strategy, WCA transforms the CFG into an integer linear programming problem which is solved using the linear programming solver lp_solve [7] resulting in a WCET estimate. In the modelbased strategy, the CFG is directly transformed into timed automata models for UPPAAL. Currently, WCET estimates using the model-based strategy are computed by making an initial guess of WCET, which can be based on the estimate derived using IPET. Afterwards, UPPAAL verifies whether the timed automata are verifiable within the guessed time and, afterwards, the estimate is gradually refined using a binary search tactic. For unbounded loops, WCA introduces comment-based annotations of source code which make explicit the iteration count of the particular loop. Alternatively, WCA provides the option of using data-flow analysis for extracting these. Obviously not all bounds can be extracted as part of static code analysis and in such cases the programmer needs to insert annotations. Furthermore, WCA performs receiver type analysis to increase the precision of the WCETs in case of dynamic method dispatch. Besides printing the resulting WCET estimate to standard output, WCA conveniently generates a detailed HTML report containing a visual representation of the CFG and timings of individual methods including their cache misses.

B. TETASARTS

TETASARTS started as an amalgamation of the SARTS [12] and TetaJ [27] tools and is today an open-source collection of timing analysis tools which operate on a timing model amenable to model checking using UPPAAL. Figure 4 shows the major components of the toolchain and their interactions.

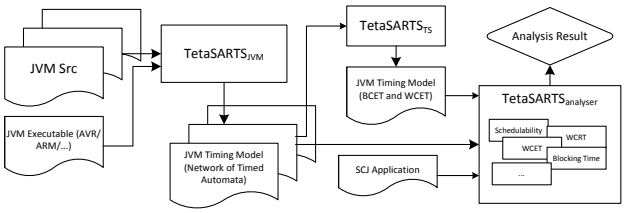


Fig. 4: Overview of the TETASARTS toolchain [38].

Reasoning about the timeliness of the system or any other real-time aspect, requires both the real-time application as well as the underlying execution environment to be modeled. Generating an appropriate model of the latter is the purpose of the TETASARTS$_{JVM}$[2] tool.

TETASARTS$_{JVM}$ generates a timing model as a Network of Timed Automata (NTA), the modeling formalism of the UPPAAL model checker. A Timed Automaton (TA) is a finite state machine extended with real-valued clocks. An NTA is the parallel composition of a number of TAs sharing clocks and actions (for details of the semantics, see [6]).

[2]TETASARTS$_{JVM}$, HVM$_{TP}$, and generated models are available on the project website: http://people.cs.aau.dk/~luckow/hvmtp/

TETASARTS$_{JVM}$ builds a TA that captures the control-flow for each of the supported Java Bytecodes. The tool processes the JVM executable such that compiler optimisations, transformations etc. are accounted for when reconstructing the control-flow. TETASARTS$_{JVM}$ conducts loop identification analysis and expects loop bounds to either be provided interactively at construction time or as comment-style annotations in the source code of the JVM. The tool allows specifying code regions constituting Java Bytecode implementations. The regions are created by embracing the Java Bytecode implementations with macros: BEGIN_JBC(X) and END_JBC(X) mark the beginning and end, respectively, of Java Bytecode X. The macros generate instrumentation code in the binary, which is used by TETASARTS$_{JVM}$ for reconstructing the control-flow. Listing 3 shows the implementation of the i2l Java Bytecode and the region specification.

```
1 case I2L_OPCODE: {
2 #if defined(INSTRUMENT)
3   BEGIN_JBC(I2L_OP);
4 #endif
5   int32 lsb = *(--sp);
6   if(lsb < 0) {
7     *sp++ = -1;
8   } else {
9     *sp++ = 0x0;
10  }
11  *sp++ = lsb;
12  method_code++;
13 #if defined(INSTRUMENT)
14  END_JBC(I2L_OP);
15 #endif
16 }
```

Listing 3: i2l [38].

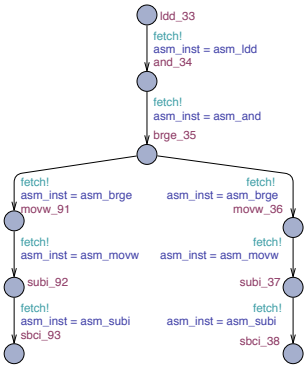
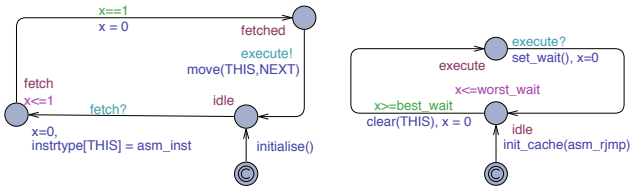


Fig. 5: TA excerpt of i2l [38].

Figure 5 shows an excerpt of the corresponding TA generated by TETASARTS$_{JVM}$. As an example, note how the location labelled *brge_35* branches to *movw_91* and *movw_36*. This captures the if-statement in line 6 of Listing 3. Each transition simulates the timing behavior of executing the instruction assigned to the UPPAAL variable *asm_inst*. In general, this explicit modeling of system behavior does not scale to large systems due to the inherent problem of state space explosion. However, it is important to note that in our case, the behavior of each Java Bytecode in isolation is sufficiently small to allow model checking to be feasible. The Java Bytecode implementations we are analyzing, range from just a few lines of C code in the simplest case, to a few hundred lines of C code producing ∼200 and ∼9000 machine instructions, respectively.

We denote the composition of the generated TAs, which yield the NTA, as the *JVM NTA*. The timing behavior depends on the hardware used; in this example the AVR ATmega2560 microcontroller whose behavior is captured by the TAs shown in Figure 6 collectively referred to as the *HW NTA*. In addition TETASARTS$_{JVM}$ supports the ARM7 and ARM9 models from METAMOC [23] and can be extended to support different hardware by providing TAs modeling the hardware. The *fetch* channel is used for hand-shake synchronisation between the JVM NTA and the pipeline fetch stage in Figure 6a; *asm_inst* communicates the instruction to be simulated in the HW NTA. Similarly, the *execute* channel establishes the communication between the fetch stage TA and the execute stage TA in Figure 6b. In both TAs, x is a clock variable simulating the



(a) Pipeline fetch stage. (b) Pipeline execute stage.

Fig. 6: Hardware TA models from METAMOC [23].

instruction processing time in the respective pipeline stage. In the execute stage TA, *worst_wait* and *best_wait* are set according to the worst and best case clock cycle execution times of the simulated instruction. Note the call *init_cache(asm_rjmp)*; it initializes the pipeline with the temporal behavior of the *rjmp* instruction. This is necessary to ensure a safe timing model, since the pipeline will be filled with this instruction prior to executing the first instruction of any of the Java Bytecodes.

Synthesizing the JVM NTA and the HW NTA yields a complete *JVM Timing Model* (see Figure 4) that simulates the timing behavior of the JVM. The JVM Timing Model can subsequently be used directly by further synthesising it with a model of an SCJ application on which various timing related analyses can be applied such as schedulability analysis and WCRT analysis using the TETASARTS$_{ANALYSER}$ tool (see [37], [41], [39] for more details).

The JVM Timing Model can also be used for representing the timing behavior of the JVM more abstractly using the TETASARTS$_{TS}$ tool (see Figure 4). For all Java Bytecode TAs, the tool determines the WCET and BCET using the sup- and inf-query extensions of the UPPAAL model checker. Sup- and inf-queries perform a state space exploration, and outputs the maximum and minimum values, respectively, of the specified clock-variables.

TETASARTS takes the interactions between tasks into account during schedulability analysis, thus systems deemed unschedulable using traditional response time analysis may be deemed schedulable using model checking as demonstrated in [12]. TETASARTS also allows the clock frequency of the target hardware to be set prior to the analysis, thus making it possible to determine the lowest clock frequency at which the system is still schedulable as demonstrated in [37].

C. SymRT

SymRT [40] is a tool based on a combination of symbolic execution [31], [19] and real-time model checking that generates a precise control-flow model from the symbolic execution trees obtained with a symbolic execution of the program. Each tree characterizes the set of *feasible* execution paths (up to some bound) of the analyzed task and yields a precise timing model.

Symbolic execution is used for generating a safe and tight timing model of the analyzed system capturing the feasible execution paths. This timing model is combined with execution environment models capturing the timing behavior of the target host platform including the JVM and complex hardware

features such as caching. The complete timing model is a NTA and directly facilitates safe estimates of Worst (and Best) Case Execution Time to be determined using the UPPAAL model checker. Furthermore, the integration of these techniques into the TETASARTS tool facilitates reasoning about additional timing properties such as the schedulability of periodically and sporadically released Java real-time tasks (under specific scheduling policies), Worst Case Response Time, and more.

The program model is built *modularly*. The timing behavior of the execution environment is obtained from *environment models* capturing the timing behavior of the JVM and hardware of the target platform. This technique generates the complete timing model, based on a configuration, as an NTA amenable for model checking using the UPPAAL [5] model checker. The NTA model can be used for estimating WCET and BCET. Furthermore, it generalizes to verification of properties expressible in Timed Computation Tree Logic (TCTL). In contrast to previous tools [27], [41], which provide limited feedback, SymRT can also generate *witness traces* that expose the reported behaviour, useful for debugging and program understanding.

Both symbolic execution and model checking have issues with scalability due to the large number of paths respectively states to explore. SymRT addresses this by using a "per task" symbolic execution, leveraging the SCJ programming model, that groups code into missions consisting of relatively short tasks. Furthermore, when using timing models of the target execution environment, the generated TA of the program is at basic block level, which significantly reduces the state space size.

Clearly there are systems for which analysis is intractable. This is for instance the case when attempting to do a full schedulability analysis for the Real-Time Sorting Machine (RTSM) [12] on HVM on the AVR processor. This application consists of 1 kloc. However, it is not the number of lines of code that limits the analysis, but its complexity, especially the number of branch points in each task and the number of tasks in the application, as the size of the UPPAAL model grows exponentially with the number of components in the NTA. However, since schedulability is viewed as a reachability problem, it may be possible to translate it into the subset of the UPPAAL modeling language supported by the opaal+LTSmin system [24]. In [22] opaal+LTSmin demonstrates a speedup of 40 on a 48 core machine compared to UPPAAL. Future work will investigate this direction.

D. Comparison of tools

This section summarizes a comparison of the WCET (and BCET) estimates obtained from WCA, TETASARTS and SYMRT, reported in more details in [40].The comparison uses as examples the Java implementations (obtained from the JOP distribution[3]) of a subset of the algorithms from the Mälardalen WCET benchmark suite [28]: Bubble Sort, Quick Sort, Insertion Sort and Binary Search. For the sorting algorithms, the array is initialized with symbolic values. For Binary Search, the search key is symbolic. Note that there was no need to provide loop bound annotations in any of the examples nor did they reach the default search depth (corresponding to 100 branches) during analysis.

For the analysis two configurations of the execution environment was used; (1) JOP and (2) HVM[4] [59] running on the AVR ATmega2560. The same configuration of the execution environment is used across the tools e.g. WCA and SYMRT have been configured with read and write wait cycles set to 1 and 2 (and cache configuration is the same).The analysis also compared measured BCET and WCET obtained by using inputs yielding the best and worst case behavior (e.g. for Bubble Sort a sorted and unsorted list were used). Furthermore, the JOP simulator was used to to read the cycle count before the first instruction is executed of the target method and after the return instruction. The measurements for HVM+AVR have been obtained in a similar way by using the debugging facilities of Atmel Studio 6. For this set of experiments, a laptop with an Intel Core i7-2620M CPU @ 2.70GHz with 8 GB of RAM was used. The peak memory consumption for symbolic execution is 500-700 MB for all examples. UPPAAL peaks at 50-200 MB during model checking. Table I shows the results for the comparison on JOP.

First note that all estimates are *safe* i.e. ($\text{BCET}_{symrt} \leq \text{BCET}_{\text{m}}$ and $\text{WCET}_{symrt} \geq \text{WCET}_{\text{m}}$) and that the precision of SYMRT is better (and in one case equally as good) as the other tools. The major contributor to the pessimistic results of TETASARTS and WCA are that they over-approximate the iterations of nested loops with interdependencies. The analysis time using SYMRT is however longer, which is due to symbolic execution. Also note that for e.g. Quick Sort, it is relatively difficult to exercise and measure the path yielding the worst case behavior since it depends on the pivot element selection.

Table II shows the comparison when using the HVM and an AVR ATmega2560.

Again all estimates produced by SYMRT are safe and more precise than TETASARTS. For this first set of experiments (including the results obtained for JOP), the analysis time is largely attributed symbolic execution. In all cases, model checking using UPPAAL takes less than a second.

We compare the schedulability analysis of SYMRT with TETASARTS using the Minepump control system [15], [27], the Real-Time Sorting Machine (RTSM) [12] and a variant of MD5SCJ [41]. For this set of experiments we used an application server with an Intel Xeon X5670 @ 2.93GHz CPU and 32 GB of RAM. The results are shown in Table III. We also conducted the analysis on a version of the Lift real-time system from Jembench [53] with 18 tasks. TETASARTS has not been able to construct the models for this. The *TD* subscript denotes that a Timing Scheme with fixed execution times for all the Java Bytecodes has been used instead of modeling their behavior as an NTA.

In all cases, the systems have been deemed schedulable, and the results show that the analysis times and memory consumptions are lower when using SYMRT. We also tried e.g. RTSM with HVM+AVR, but the complexity of the resulting models regardless of the tool used, is too big, which can

[3]Available for download at http://www.jopdesign.com/

[4]Available for download at http://icelab.dk/

System	SYMRT (JPF-SYMBC-RT)			TETASARTS		WCA		Measured	
	BCET [cycles]	WCET [cycles]	An. Time [seconds]	WCET [cycles]	An. Time [seconds]	WCET [cycles]	An. Time [seconds]	$BCET_m$ [cycles]	$WCET_m$ [cycles]
Binary Search	136	818	1	927	1	818	1	138	722
Bubble Sort	653	1,253	51	1,770	2	1,553	1	653	1,253
Quick Sort	1,425	2,638	510	18,749	5,375	20,275	1	1,425	1,895
Insertion Sort	774	2,586	21	4,600	1	4,296	1	774	2,586

TABLE I: Comparison of SYMRT, TETASARTS, and WCA [40].

System	SYMRT (JPF-SYMBC-RT)			TETASARTS		Measured	
	BCET [cycles]	WCET [cycles]	An. Time [seconds]	WCET [cycles]	An. Time [seconds]	BCET [cycles]	WCET [cycles]
Binary Search	3,991	65,046	2	70,153	2	4,140	23,262
Bubble Sort	19,514	93,380	50	287,526	31	19,754	37,388
Quick Sort	42,651	151,784	589	133,134	228	43,251	50,437
Insertion Sort	20,351	182,099	21	244,680	4	22,625	70,028

TABLE II: Comparison of SYMRT and TETASARTS for systems running on the HVM and AVR [40].

System	Exec. Env.	Analysis time		Memory	
		SYMRT	TETASARTS	SYMRT	TETASARTS
Minepump	HVM+AVR	14h 12m	15h 25m	16274 MB	17933 MB
Minepump	HVM+AVR_{TD}	$<$ 1s	2s	8 MB	11 MB
Minepump	JOP	$<$ 1s	1s	5 MB	11 MB
RTSM	HVM+AVR_{TD}	$<$ 1s	1m 2s	7 MB	17 MB
RTSM	JOP	$<$ 1s	5s	6 MB	15 MB
MD5SCJ	HVM+AVR_{TD}	$<$ 1s	8s	7 MB	17 MB
MD5SCJ	JOP	$<$ 1s	1m 23s	5 MB	47 MB
Lift	HVM+AVR_{TD}	33m 6s	–	5897 MB	–
Lift	JOP	15m 43s	–	6037 MB	–

TABLE III: Comparison of TETASARTS and SYMRT [40].

be attributed the JVM NTA, which largely dominates the complexity. In this case, UPPAAL runs out of memory.

IV. CONCLUSION

In this paper we have presented the Safety Critical Java programming model, some of its implementations in the form of the JOP, FijiVM and HVM and some of the tools, especially WCA, TetaSARTS tool suite and SymRT, allowing programmers to analyze their SCJ applications for correct time behaviour. Furthermore, we have argued the suitability of SCJ for use in robotics applications with hard-real-time constraints.

Small, but realistic SCJ applications, such as the Minepump control system, the Real-Time Sorting Machine (RTSM) and a variant of MD5SCJ, as well as a number of applications from the Mälardalen WCET Benchmarks, have been implemented and analyzed using the above mentioned tools. A number of student projects have used either the JOP or the HVM on either Arduino or Lego EV3 to learn about robotics programming using robots build in Lego Mindstorm. The JOP has been used in industrial applications such as the Kippfahrleitung system for the Austrian Railways controlling up to 15 independent motors [52]. Although not written SCJ, larger systems with real-time constraints have been developed in Java. [2] presents the first use of Real-time Java in avionics in the context of control software for a ScanEagle Unmanned Aerial Vehicle, and [56] presents the Use of PERC Pico in the AIDA Avionics Platform. Our own work include the performance analysis of different components of a NASA tactical layer solution for planes, T-TSAFE, currently focusing on the conflict detection and conflict resolution algorithms.

Clearly SCJ is not the solution for all robotics applications. The programming model of tasks and missions, and especially the scoped memory model, is rather restrictive. All the mentioned JVM implementations have real-time garbage collection implementations available, and it would thus be possible to dispense with the scoped memory model. To the best of our knowledge, at present, none of these garbage collectors have been analyzed for time predictability and thus cannot be used in systems needing to comply with standards such as DO-178C, ISO-26262, IEC-61508 and EN-50128. However, we expect this to be just a matter of time and hard work.

Many robotics systems will have components that are not time critical. Currently SCJ does not cater well for such mixed criticality systems. RTSJ caters for such mixed criticality systems, however, at the expense of analyzability. We envision that recent developments in compositional schedulability analysis [14] could be integrated with the SCJ programming model, basically allowing different missions to have different scheduling policies and using the SCJ Level 2 notion of nested missions to implement hierarchies of components with different criticality levels. This would allow components with no real-time requirements to execute in their own missions as long as there is time budget for the time critical components to execute their tasks. The HVM facilitates a tight integration with (legacy) code in C, i.e. handlers in Java can directly be called from handlers in C and visa versa, clearly at the expense of more complex analysis, however, some some systems it is not possible to port all parts of the code to Java. We envision that the techniques of I/O automatas [25] used in the ECDAR tool, analysis of C code using METAMOC [23] and the notion of schedulability abstraction [9] could be combined to provide

a framework for analysis of such mixed applications.

Many advanced robotic systems are programmed using rule-based systems. Another, direction for future work, would be to cater for such robotics systems by implementing a time predictable rule match algorithm in SCJ, e.g. implementing a variant of the rete-algorithm [18] used in some advanced robotics applications.

REFERENCES

[1] Aicas. *JamaicaVM User Manual: Java Technology for Critical Embedded Systems*, 2010.

[2] Austin Armbruster, Jason Baker, Antonio Cunei, Chapman Flack, David Holmes, Filip Pizlo, Edward Pla, Marek Prochazka, and Jan Vitek. A real-time Java virtual machine with applications in avionics. *ACM Trans. Embed. Comput. Syst.*, 7(1):5:1–5:49, December 2007.

[3] Atego. Atego Home, 2013. http://atego.com/.

[4] Clément Ballabriga, Hugues Cassé, Christine Rochange, and Pascal Sainrat. OTAWA: An Open Toolbox for Adaptive WCET Analysis. In *Software Technologies for Embedded and Ubiquitous Systems*, pages 35–46. Springer, 2011.

[5] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. UPPAAL – a Tool Suite for Automatic Verification of Real-Time Systems. In *Hybrid Systems*. 1996.

[6] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, Lecture Notes in Computer Science. 2004.

[7] M Berkelaar. lp_solve reference guide.[online].", 2014.

[8] Thomas Bøgholm, Christian Frost, RenéRydhof Hansen, CasperSvenning Jensen, KasperSøe Luckow, AndersP. Ravn, Hans Søndergaard, and Bent Thomsen. Towards harnessing theories through tool support for hard real-time Java programming. *Innovations in Systems and Software Engineering*, 9(1):17–28, 2013.

[9] T. Bogholm, B. Thomsen, K.G. Larsen, and A. Mycroft. Schedulability analysis abstractions for Safety Critical Java. In *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2012 IEEE 15th International Symposium on*, pages 71 –78, april 2012.

[10] Thomas Bøgholm, René R. Hansen, Anders P. Ravn, Bent Thomsen, and Hans Søndergaard. A predictable Java profile: Rationale and implementations. In *Proceedings of the 7th International Workshop on Java Technologies for Real-Time and Embedded Systems*, JTRES '09, pages 150–159, New York, NY, USA, 2009. ACM.

[11] Thomas Bøgholm, René R. Hansen, Anders P. Ravn, Bent Thomsen, and Hans Søndergaard. Schedulability analysis for Java finalizers. In *Proceedings of the 8th International Workshop on Java Technologies for Real-Time and Embedded Systems*, JTRES '10, pages 1–7, New York, NY, USA, 2010. ACM.

[12] Thomas Bøgholm, Henrik Kragh-Hansen, Petur Olsen, Bent Thomsen, and Kim G. Larsen. Model-based Schedulability Analysis of Safety Critical Hard Real-time Java Programs. In *Proceedings of the 6th International Workshop on Java Technologies for Real-time and Embedded Systems*, JTRES '08, pages 106–114, New York, NY, USA, 2008. ACM.

[13] Gregory Bollella and James Gosling. The real-time specification for java. *IEEE Comp.*, 2000.

[14] Jalil Boudjadar, Kim Guldstrand Larsen, Jin Hyun Kim, and Ulrik Nyman. *Compositional Schedulability Analysis of An Avionics System Using UPPAAL*. 2014.

[15] Alan Burns and Andy Wellings. *Real-Time Systems and Programming Languages: Ada 95, Real-Time Java, and Real-Time POSIX*. 2009.

[16] Alan Burns and Andy Wellings. *Real-Time Systems and Programming Languages: ADA 95, Real-Time Java, and Real-Time POSIX*. Addison-Wesley Educational Publishers Inc., Boston, MA, USA, 4th edition, 2009.

[17] Mälardalen University Real-Time Research Center. SWEET (SWEdish Execution Time tool). http://www.mrtc.mdh.se/projects/wcet/sweet/.

[18] Yanik Kim Challand. A real time expert system: Adapting match algorithms and implementing a tailored rule language. Master's thesis, Aalborg University, June 2011.

[19] Lori A. Clarke. A System to Generate Test Data and Symbolically Execute Programs. *IEEE Trans. Software Eng.*, 1976.

[20] Antoine Colin and Isabelle Puaut. A modular and retargetable framework for tree-based WCET analysis. In *Real-Time Systems, 13th Euromicro Conference on, 2001.*, pages 37–44. IEEE, 2001.

[21] Java Robotics Community. Java robotics community. https://community.java.net/community/robotics.

[22] Andreas E Dalsgaard, Alfons Laarman, Kim G Larsen, Mads Chr Olesen, and Jaco Van De Pol. Multi-core reachability for timed automata. In *Formal Modeling and Analysis of Timed Systems*, pages 91–106. Springer, 2012.

[23] Andreas E. Dalsgaard, Mads Chr. Olesen, Martin Toft, René Rydhof Hansen, and Kim Guldstrand Larsen. METAMOC: Modular Execution Time Analysis using Model Checking. In *10th International Workshop on Worst-Case Execution Time Analysis*, 2010.

[24] AndreasEngelbredt Dalsgaard, RenéRydhof Hansen, KennethYrke Jørgensen, KimGulstrand Larsen, MadsChr. Olesen, Petur Olsen, and Jiří Srba. opaal: A Lattice Model Checker. In Mihaela Bobaru, Klaus Havelund, GerardJ. Holzmann, and Rajeev Joshi, editors, *NASA Formal Methods*, volume 6617 of *Lecture Notes in Computer Science*, pages 487–493. Springer Berlin Heidelberg.

[25] Alexandre David, Kim G. Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. Timed i/o automata: A complete specification theory for real-time systems. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '10, pages 91–100, New York, NY, USA, 2010. ACM.

[26] Christian Ferdinand, Reinhold Heckmann, and Bärbel Franzen. Static memory and timing analysis of embedded systems code. In Perry Groot, editor, *Proceedings of VVSS2007 - 3rd European Symposium on Verification and Validation of Software Systems, 23rd of March 2007, Eindhoven*, number TUE Computer Science Reports 07-04, 2007.

[27] Christian Frost, Casper Svenning Jensen, Kasper Søe Luckow, and Bent Thomsen. WCET analysis of Java bytecode featuring common execution environments. In *9th International Workshop on Java Technologies for Real-Time and Embedded Systems*, 2011.

[28] Jan Gustafsson, Adam Betts, Andreas Ermedahl, and Björn Lisper. The Mälardalen WCET Benchmarks - Past, Present and Future. In *Proceedings of the 10th International Workshop on Worst-Case Execution Time Analysis*, July 2010.

[29] Niklas Holsti and Sami Saarinen. Status of the Bound-T WCET tool. *Space Systems Finland Ltd*, 2002.

[30] Benedikt Huber and Martin Schoeberl. Comparison of implicit path enumeration and model checking based wcet analysis. In *OASIcs-OpenAccess Series in Informatics*, volume 10. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[31] James C. King. Symbolic Execution and Program Testing. *Commun. ACM*, 1976.

[32] Stephan Korsholm. Hvm, 2013. http://www.icelab.dk.

[33] Stephan Korsholm, Anders P. Ravn, Christian Thalinger, and Martin Schoeberl. Hardware objects for Java. In *Proceedings of the 11th IEEE International Symposium on Object/component/serviceoriented Real-time distributed Computing (ISORC 2008*, 2008.

[34] Xianfeng Li, Yun Liang, Tulika Mitra, and Abhik Roychoudhury. Chronos: A Timing Analyzer for Embedded Software. *Science of Computer Programming*, 69(1):56–67, 2007.

[35] Yau-Tsun Steven Li and Sharad Malik. Performance Analysis of Embedded Software Using Implicit Path Enumeration. In *Proceedings of the 32Nd Annual ACM/IEEE Design Automation Conference*, DAC '95, pages 456–461, New York, NY, USA, 1995. ACM.

[36] Doug Locke, B. Scott Andersen, Ben Brosgol, Mike Fulton, Thomas Henties, James J. Hunt, Johan Olmütz Nielsen, Kelvin Nilsen, Martin Schoeberl, Joyce Tokar, Jan Vitek, and Andy Wellings. Safety-Critical Java Technology Specification, Public Draft. 2011.

[37] K. S. Luckow, T. Bøgholm, and B. Thomsen. Supporting Development of Energy-Optimised Java Real-Time Systems using TetaSARTS. In *WiP Proceedings of the 19th Real-Time and Embedded Technology and Application Symposium*, 2013.

[38] K. S. Luckow, B. Thomsen, and S. E. Korsholm. HVM-TP: A Time Predictable and Portable Java Virtual Machine for Hard Real-

Time Embedded Systems. In *12th International Workshop on Java Technologies for Real-Time and Embedded Systems*, 2014. To appear.

[39] K. Søe Luckow, Thomas Bøgholm, Bent Thomsen, and Kim Guldstrand Larsen. TetaSARTS: Modular Timing and Performance Analysis of Safety Critical Java Systems. *Concurrency and Computation: Practice and Experience*, 2014. In Submission.

[40] Kasper Søe Luckow. *Platforms and Model-Based Analyses for Real-Time Java*. PhD thesis, Department of Computer Science, Aalborg University, 2014, http://people.cs.aau.dk/~luckow/thesis.pdf.

[41] Kasper Søe Luckow, Thomas Bøgholm, Bent Thomsen, and Kim Guldstrand Larsen. Tetasarts: A tool for modular timing analysis of safety critical Java systems. In *Proceedings of the 11th International Workshop on Java Technologies for Real-time and Embedded Systems*, JTRES '13, pages 11–20, New York, NY, USA, 2013. ACM.

[42] Kelvin Nilsen. Differentiating features of the perc virtual machine, 2009.

[43] Filip Pizlo, Lukasz Ziarek, and Jan Vitek. Real Time Java on Resource-constrained Platforms with Fiji VM. In *Proceedings of the 7th International Workshop on Java Technologies for Real-Time and Embedded Systems*, JTRES '09, pages 110–119, New York, NY, USA, 2009. ACM.

[44] Ales Plsek, Lei Zhao, Veysel H. Sahin, Daniel Tang, Tomas Kalibera, and Jan Vitek. Developing safety critical Java applications with oscj/l0. In *Proceedings of the 8th International Workshop on Java Technologies for Real-Time and Embedded Systems*, JTRES '10, pages 95–101, New York, NY, USA, 2010. ACM.

[45] Adrian Prantl, Markus Schordan, and Jens Knoop. TuBound - A Conceptually New Tool for Worst-Case Execution Time Analysis. In Raimund Kirner, editor, *8th International Workshop on Worst-Case Execution Time Analysis (WCET'08)*, volume 8 of *OpenAccess Series in Informatics (OASIcs)*, Dagstuhl, Germany, 2008. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. also published in print by Austrian Computer Society (OCG) with ISBN 978-3-85403-237-3.

[46] Jerry Pratt and Ben Krupp. Design of a bipedal walking robot. In *SPIE Defense and Security Symposium*, pages 69621F–69621F. International Society for Optics and Photonics, 2008.

[47] RapiTime. Rapitime wcet tool homepage. http://www.rapitasystems.com.

[48] LCC RidgeSoft. Robojde java-enabled robotics software development environment, 2014. http://www.ridgesoft.com/robojde/robojde.htm.

[49] Sven Gestegård Robertz, Roger Henriksson, Klas Nilsson, Anders Blomdell, and Ivan Tarasov. Using real-time java for industrial robot control. In *Proceedings of the 5th international workshop on Java technologies for real-time and embedded systems*, pages 104–110. ACM, 2007.

[50] Andreas Schierl, Andreas Angerer, Alwin Hoffmann, Michael Vistein, Wolfgang Reif, and I Vision. Using Java for real-time critical industrial robot programming. 2012.

[51] Martin Schoeberl. JOP: A Java Optimized Processor. In *Proceedings of Java Technologies for Real-Time and Embedded Systems*, 2003.

[52] Martin Schoeberl. Using a java optimized processor in a real world application. In *WISES*, pages 165–176, 2003.

[53] Martin Schoeberl, Thomas B. Preusser, and Sascha Uhrig. The Embedded Java Benchmark Suite JemBench. In *Proceedings of the 8th International Workshop on Java Technologies for Real-Time and Embedded Systems*, JTRES '10, pages 120–127, New York, NY, USA, 2010. ACM.

[54] Martin Schoeberl, Wolfgang Puffitsch, Rasmus Ulslev Pedersen, and Benedikt Huber. Worst-Case Execution Time Analysis for a Java Processor. *Softw.: Prac. and Exp.*, 2010.

[55] Martin Schoeberl, Wolfgang Puffitsch, Rasmus Ulslev Pedersen, and Benedikt Huber. Worst-case Execution Time Analysis for a Java Processor. *Software: Prac. and Exp.*, 40(6):507–542, 2010.

[56] Tobias Schoofs, Eric Jenn, Stéphane Leriche, Kelvin Nilsen, Ludovic Gauthier, and Marc Richard-Foy. Use of perc pico in the aida avionics platform. In *Proceedings of the 7th International Workshop on Java Technologies for Real-Time and Embedded Systems*, JTRES '09, pages 169–178, New York, NY, USA, 2009. ACM.

[57] Jesper Smith, Douglas Stephen, Alex Lesman, and Jerry Pratt. Real-time control of humanoid robots using openjdk. In *Proceedings of the 12th International Workshop on Java Technologies for Real-time and Embedded Systems*, JTRES '14, pages 29:29–29:36, New York, NY, USA, 2014. ACM.

[58] JP Smith. *Online swing leg trajectory optimization for a force controllable humanoid robot*. PhD thesis, TU Delft, Delft University of Technology, 2012.

[59] Hans Søndergaard, Stephan E. Korsholm, and Anders P. Ravn. Safety-critical Java for low-end embedded platforms. In *Proceedings of the 10th International Workshop on Java Technologies for Real-time and Embedded Systems*, JTRES '12, 2012.

[60] Kleanthis Thramboulidis and Alkiviadis Zoupas. Real-time Java in control and automation: a model driven development approach. In *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, volume 1, pages 8–pp. IEEE, 2005.

[61] Reinhard Wilhelm and Daniel Grund. Computation Takes Time, but How Much? *Commun. ACM*, 57(2):94–103, February 2014.

[62] Shuai Zhao. Implementing level 2 of Safety-Critical Java, 2014.