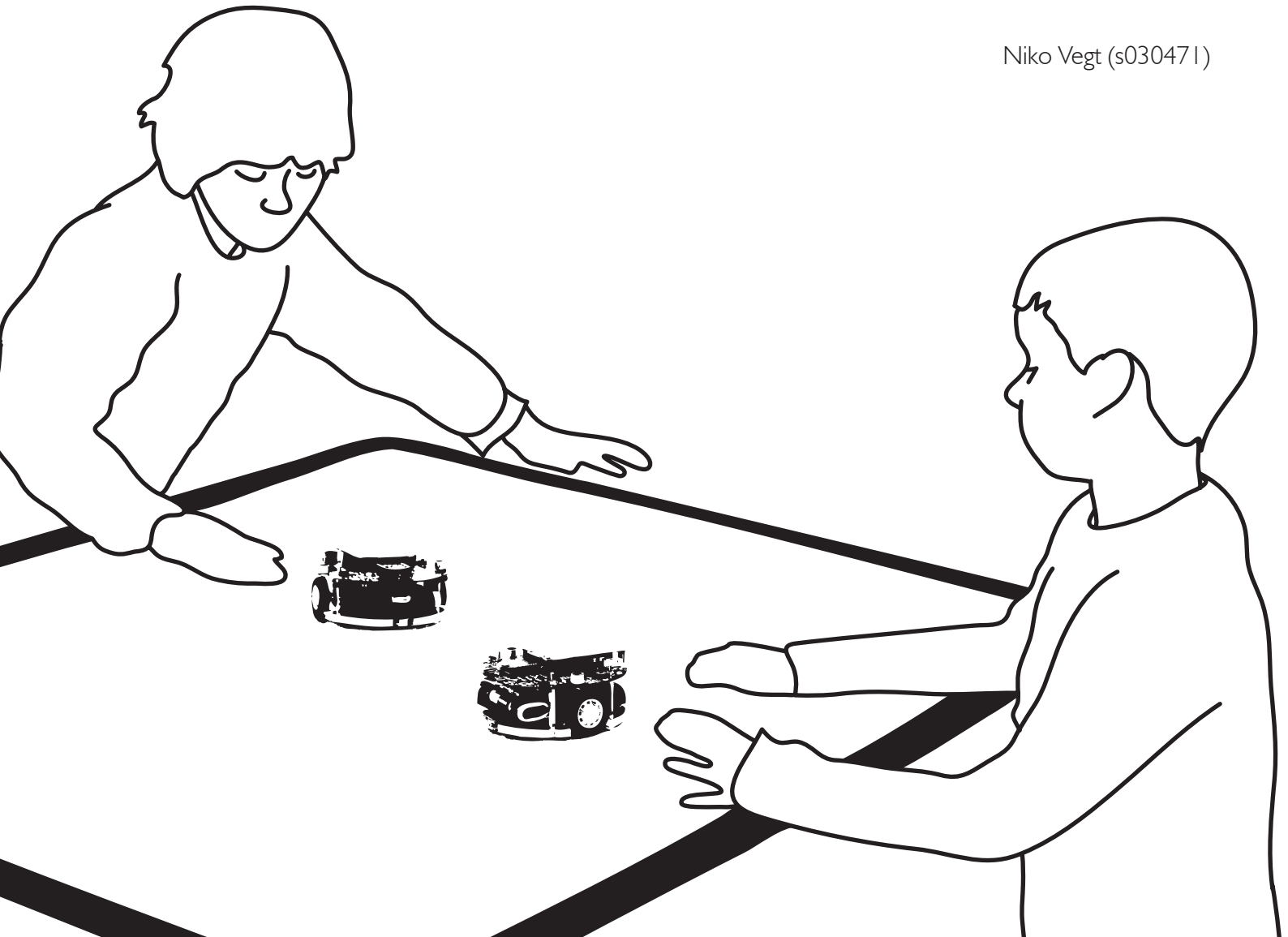
Smart interactions

Child-robot interaction through movement

Niko Vegt (s0304711)



M1.2 project
Coach: Emilia Barakova

External expertise:
OBS De Tweesprong
OBS De Hasselbraam
Sint Marie

Start: 02-09-2009
End: 08-01-2010

Content

Introduction

Children with an autistic spectrum disorder (ASD) lack development of social competence. This results in problems with initiating contact with other people and having no friends. The amount of children diagnosed with an autistic spectrum disorder is increasing and therefore the demand for solutions helping these children is growing as well.

In terms of participation to society autistic children benefit from intensive education in social behaviour [1]. The use of robots for educative applications is increasing. Especially autistic children respond positively on interaction with robots and technology in general. "Autistic children are fond of technological toys" [2]. So deploying robots in educating children with an autistic spectrum disorder seems to be an obvious solution. Teaching social competences through robots is promising but to achieve this, the effect of human robot interaction on social behaviour needs to be investigated.

In [3]it is argued that "interaction with objects is a developmental stage of social behaviour, and that shortcomings in the motor level of interaction can result in impaired social behaviour." Body movement appears to be a valuable denominator for human emotions. Autistic children tend to have difficulties recognizing emotions through body language. Therefore interactions specifically focused on movements that elicit emotion are valuable for social interaction education. Interaction through movement with products or robots is a fairly unexplored field though. Early tests with synthesizing emotions with robots through movement show promising results. "A control group of 42 typically developing children were tested to observe the robots emotional behaviours. The outcome of the tests showed a good recognition of several basic emotions" [2].

Another topic in the field of technological solutions for autistic children is multi-agent systems (MAS). Tests with autistic children show that playing with a multi-agent system of coloured light blocks [4] encourages explorative play, rather than repetitive play. Multi-agent systems become more and more used to control or analyse complex systems. They might also be used to simulate the

complexity of social interaction. Parameters of emotional movements are known and neural network algorithms provide the opportunity to filter and interpret emotion from complex human movement patterns [5]. In relation to communication through motion multi-agent systems haven't been investigated though.

Collaborative games with the blocks have been tested. It became clear that autistic children can be encouraged to make social contact [6]. This is also shown in the research by Legoff on using Lego© in therapy sessions [1]. Autistic children increased their social skills significantly after playing in couples with Lego©.

Study objectives

Introduction

Numerous publications seem to lead to a relation between the development of motor skills and social skills. The implementation of interaction through movement is therefore an interesting topic to investigate. Making robots to encourage movement appeals to the motor skills and might therefore stimulate social activity. The relation between emotions and motion agrees to this.

The synthesized emotions through movement [2] seem not suitable for human-robot interaction yet though. These kinds of movement patterns are promising but not yet controllable enough for valid implementation in interaction scenarios. So in this test simplified movement patterns are used.

In [3] it is mentioned that three types of interaction can be implemented: "Robots that imitate, enhance or counteract an emotional state of a person". Interesting to know is the relation between the interaction behaviours and social interaction.

This can only be tested if the context of the test supports social interaction. At least two children have to participate in interacting with robots through movement. Giving them a collaborative task will support social interaction even more and the implementation of a multi-agent system should arouse a more open (explorative) attitude and encourage social interaction as well [4]. These three contextual elements should provide enough reason for social interaction and enable a test on the different interaction behaviours.

Problem statement

Main research question

Will counteracting, imitating or enhancing two children's hand movements by a multi-agent system of robots encourage social interaction?

Sub research questions

1. Is teaching movement patterns an engaging task?
2. To which extend will the children interpret the behaviour of the MAS of moving robots?
3. How do the children react on the different interaction behaviours?
4. Does interaction through movement enhance social interaction?
5. Is there a difference in social interaction between a MAS communicating through movement and a MAS communicating through coloured lighting?
6. Is there a relation between the explorative character of multi-agent games and the enhancement of social interaction by the multi-agent system?

Objectives

Main objectives

- To compare the amount of social interaction between two children when performing a collaborative task by interacting with a multi-agent system that imitates, counteracts or enhances hand movement patterns.
- To quantify the relation between the multi-agent system interaction behaviour and the amount of social interaction occurring during a collaboration task.

Sub objectives

- To propose guidelines for suitable interactive behaviour of multi-agent systems to enhance social interaction in a collaboration task.
- To generate a vision on the use of multi-agent systems for educating social skills to groups of children.

Interaction scenario

Introduction

Several interaction scenarios that suited the project objectives were explored to achieve a more concrete research proposal. This involved roughly two concept directions. The first direction focused on a game with a multi-agent system and two children. The second direction was proposed to investigate the interaction with a multi-agent system without giving the children a task. The scenarios were cooperative or competitive. This resulted in four main scenario options.

Scenario evaluation

The cooperative game scenario appeared to be most suitable to accommodate social interaction between the children. The competitive scenario involved a much more individualistic approach. Implementing a task in a game was supposed to be most engaging to the children. Having a clear goal together should result in increased involvement with the task. The pure interaction scenarios enabled the investigation of particular interaction behaviour, where imitation was the main focus. This would involve a more specialised investigation whereas a broader exploration (levelling the interaction behaviours) is more useful to the field.

So in two directions the cooperative game scenario was most interesting to develop. Firstly it would be most appealing to the children and therefore also come closest to an actual educative implementation. This was also confirmed by the reactions from the autism experts at the Sint Marie educative centre on the research proposal who proposed an implementation in the TOM (theory of mind) groups. The game scenario could suit an imitation training. Secondly the research field is suited best by exploring different interaction behaviours on an even level instead of (on beforehand) assuming a superior behaviour in interaction through movement.

Scenario options

Cooperative game

- Robots perform five different patterns
- Children pick a movement they want to teach
- Robots make random movements
- Children ask a robot for attention by spreading their arms
- Children perform the movement
- Attended robots start imitating the movement
- Surrounding robots take over slowly from other robots
- When children shift the attention to another robot the learning goes much faster
- If the children are satisfied they stop spreading their arms
- Robots keep on performing the learned movement

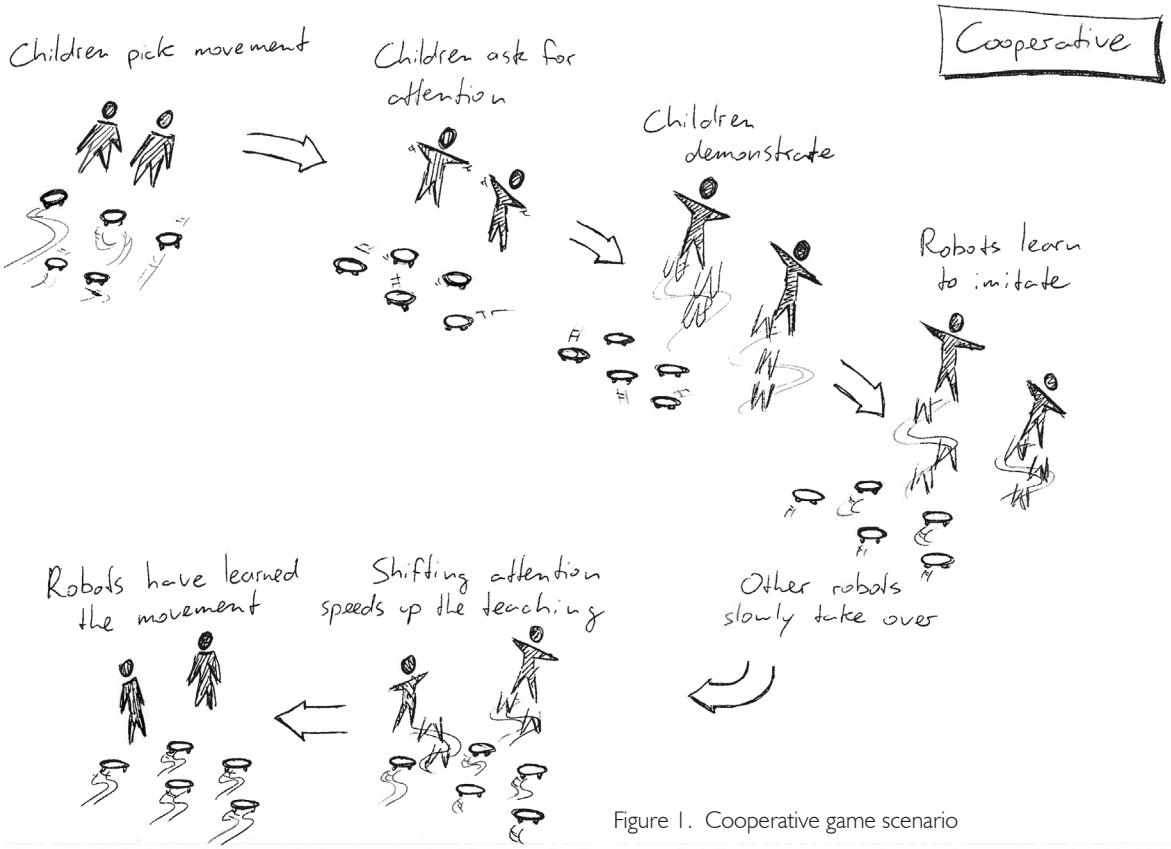


Figure 1. Cooperative game scenario

Competitive game

- Children receive a separate task to learn the robots a movement (performed by the robots)
- Children ask attention to a robot by spreading their arms
- Children perform the movement to teach the robot
- Other robots move randomly
- Other robots take over the performed movement when they are close enough
- One child becomes more successful in teaching the robots the movement
- Child that has taught all robots his/her pattern wins

Figure 2. Competitive game scenario

Interaction through imitation (children lead)

- Robots perform a series of different patterns
- Children pick one movement pattern to imitate
- Children spread their arms and start imitating the chosen movement pattern
- Robots slowly filter all other movement patterns from their behaviour
- Children and robots perform the same movement

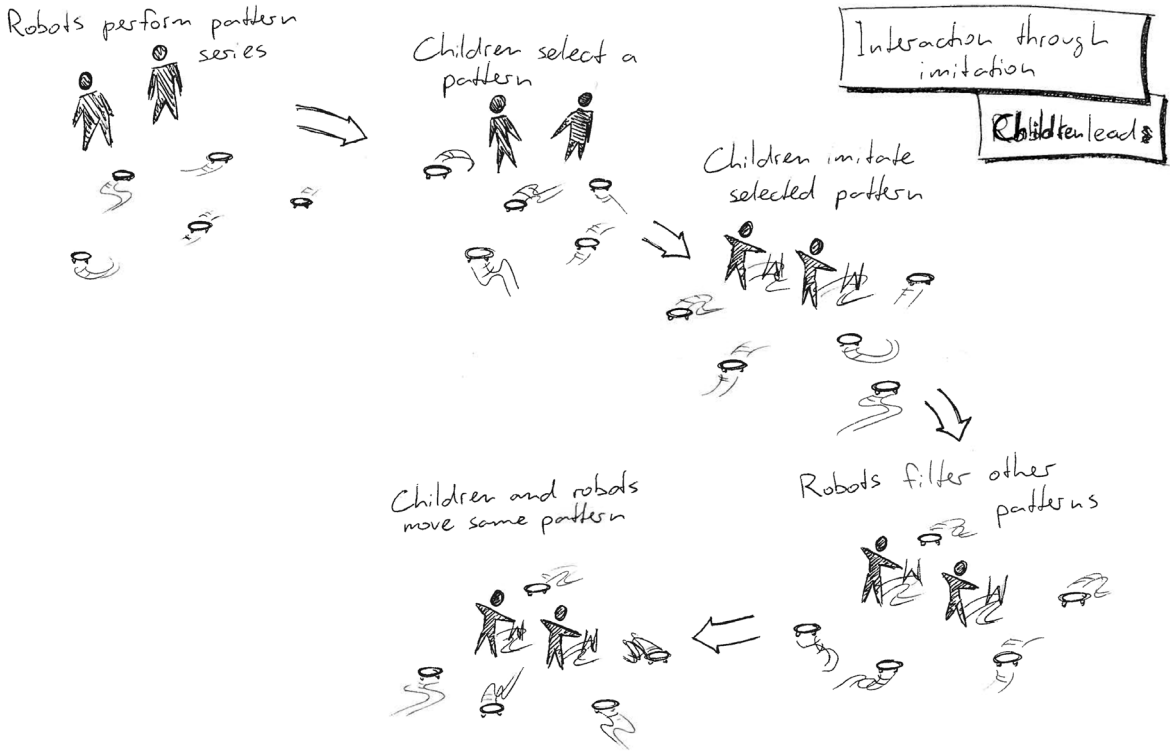


Figure 3. Interaction through imitation scenario (child is leading)

Interaction through imitation (robots lead)

- Children perform series of movement patterns
- Robots select one movement pattern to imitate
- Robots start imitating the selected movement pattern
- Children react on the robots' reaction by imitating, enhancing or counteracting
- Robots react on the children's reaction

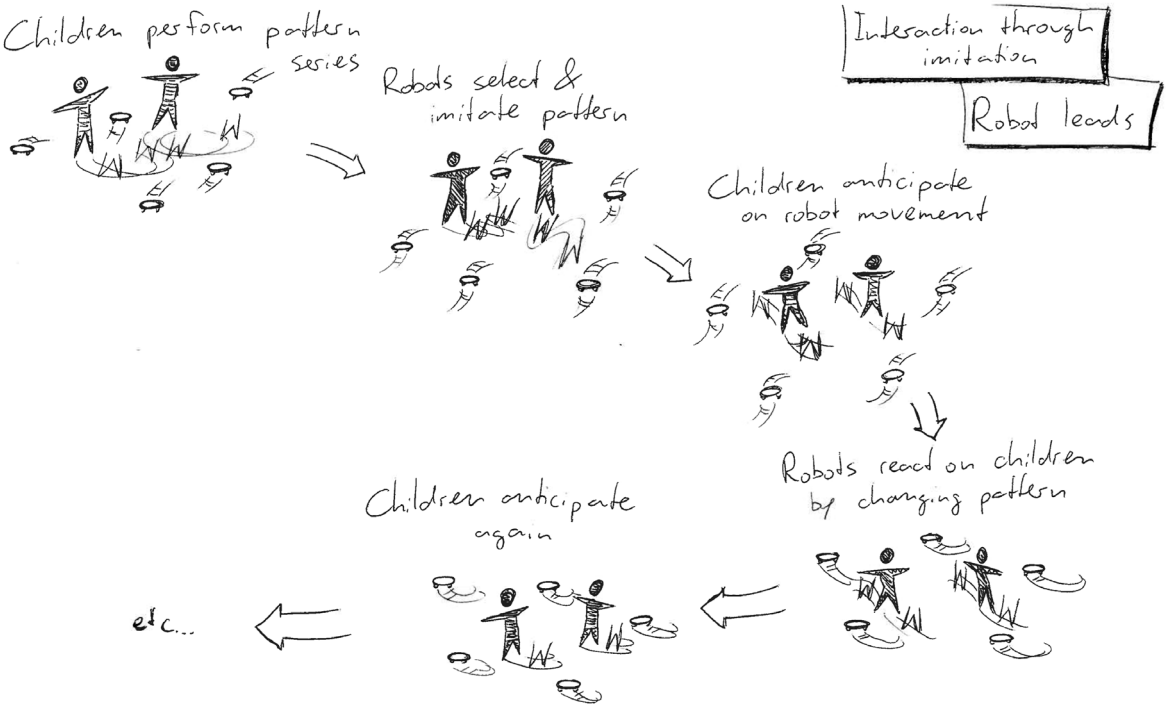


Figure 4. Interaction through imitation scenario (robot is leading)

Scenario description

Three robots are standing on a table, waiting to start a game. The goal of the game is to teach three robots a movement pattern. This pattern is firstly demonstrated by the robots so that the children can rehearse and memorize the pattern. Than the robots stop the demonstration and perform neutral behaviour: driving straight with collision and table edge avoidance. The children need to perform the movement pattern with one hand above the table (for technical reasons). If the children perform this movement together the robots will respond. The responding behaviour will differ per round. The robots can imitate, enhance or counteract. When they perform imitation behaviour the robots will copy the children's hand movement accurately. The enhancing behaviour implies very quick recognition of the children's movements. As soon as the children are somewhat performing the same movement the robots will perform the right pattern. In counteraction mode the robots will not be able to pick up the right movement pattern but start performing different patterns. With this behaviour the children aren't able to finish the game. So a round is finished if the robots perform the right pattern or if three minutes have passed.

Goal

Teach the robots a movement

Means

Rehearse the movement
Perform the movement together

Scenario sequence

- Robots demonstrate a movement pattern
- Children need to describe and rehearse the pattern
- Robots stop performing the demonstration
- Robots perform neutral behaviour
- Children have to teach the robots the pattern through hand movement
- Children have to perform the movement together
- Robots enhance, imitate or counteract (differs per task)
- Robots stop if the right pattern is taught or if 3 minutes have passed
- Game is repeated three times

Technological platform implementation

Introduction

To accommodate the pattern teaching game an autonomous platform was developed. This involved image recognition, serial communication and robot behaviour handling. The movements of the children had to be recognized from a webcam image and processed to understand their movement patterns. The result had to be communicated to the robots and they had to respond correspondingly.

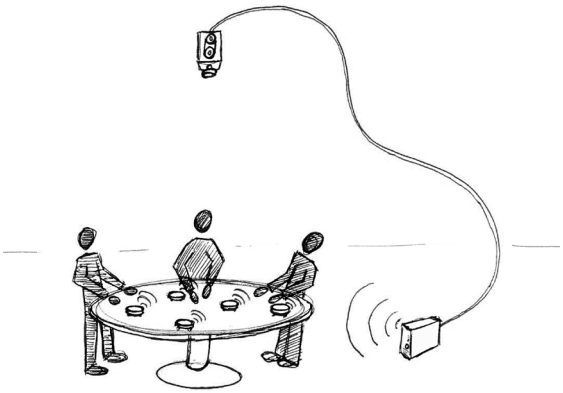


Figure 5. Test setup

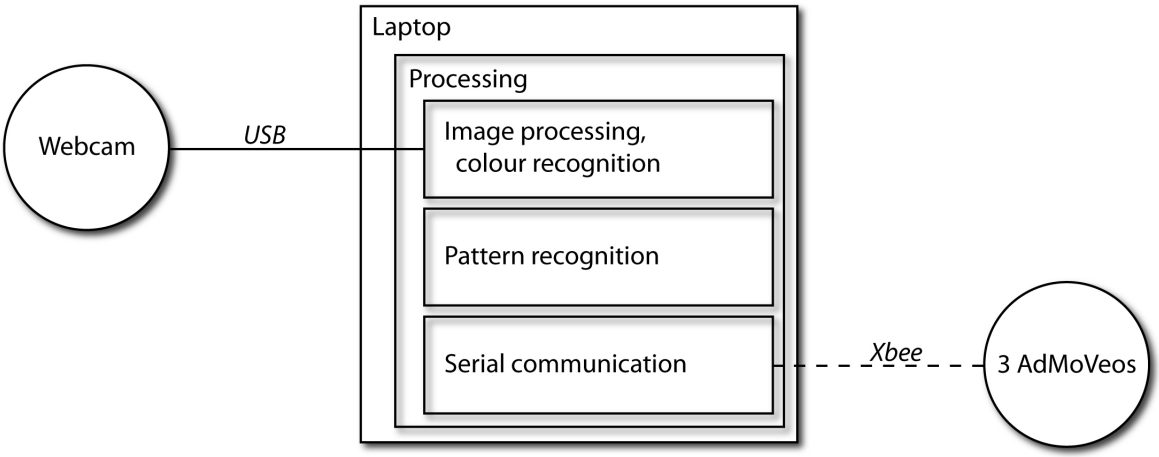


Figure 6. Platform components

AdMoVeo

The robots used for the platform are AdMoVeo robots [7]. They hold many sensors and actuators accommodating a versatility of applications. For this project the driving and lighting functionality was involved as well as sensors for autonomous collision avoidance (Infrared distance sensors). They were programmed in Arduino code (Appendix II) involving autonomous behaviour that could override communicated behaviour. Any robot read serial communication through an Xbee module. The behaviour related to the different interaction behaviours was communicated from laptop to robot. This behaviour could be overruled by avoidance of other robots or avoiding the edge of the table. Neutral behaviour involved driving straight and forward to show that they're active. Coloured lighting was used to enhance the expression of certain behaviour, like the recognition of the right or wrong movement pattern (red or green light) and the overruling collision avoidance behaviour (blue light).

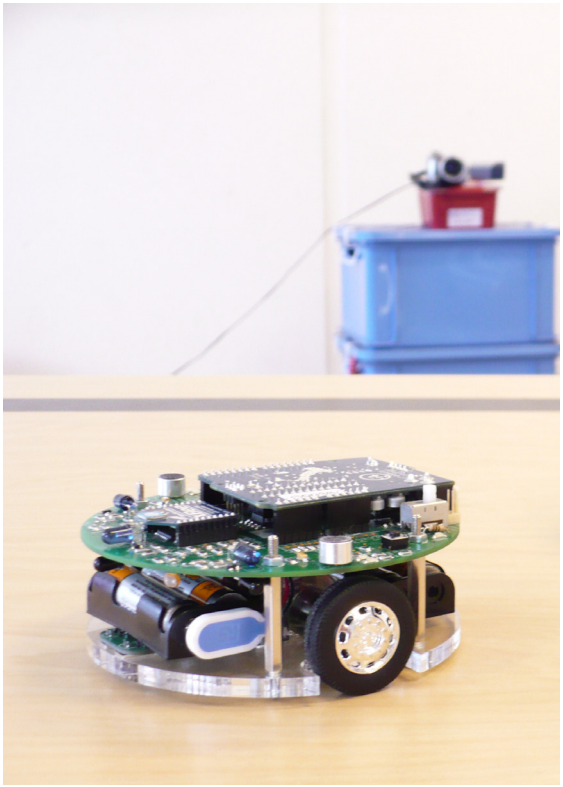


Figure 7. AdMoVeo robots

Image processing

The movements that the children would make had to be recognized from an overview perspective. A webcam was implemented above a table to provide a top view image of the children and game area. This video image had to be processed so several programming platforms were explored to generate reliable data.

To get a better understanding of possible errors that had to be taken care of, different image scenarios were investigated (Appendix V). They gave insight in how the children had to be instructed and provided guidelines for the technological implementation. It became clear that the best way to recognize and compare children's hand movements would involve separation of the video image in two halves and asking the children to stand on opposite sides of the table, corresponding to the image division. This would ensure that the recognized hands wouldn't be mixed up and that the recognized patterns from both image parts could be compared. Restricting the children to place only one hand above the table would ease the recognition of a pattern because it could be assumed that the colour-tracked point resembled the hand that the child was performing the movement with.

An implementation in the C++ language was a strong solution in terms of achieving reliable data from the image. TiViPE software [8] could be used to recognize a hand and get its place, speed and acceleration. This involved serious expertise on programming though and analysis methods had to be integrated yet, as well as the communication to the robots. A serial communication library in C++ [9] was found, modified and implemented (Appendix IV) to steer the AdMoVeo robots. This took more than a week but wasn't the major software element to develop though. Therefore it was decided to shift to a different, less time consuming programming platform.

Programming the image processing in Processing was more accessible but less accurate. Within the timeframe of this project, it was the most feasible solution. A module for communication with the robots, colour recognition and a pattern recognition algorithm [10] was available in JAVA. This

enabled the recognition of children's hands above a table, register and recognize patterns in their movements and communicate this to the robots. So the existing software modules were modified to suit the platform and integrated in one program (Appendix III) to perform the image processing.

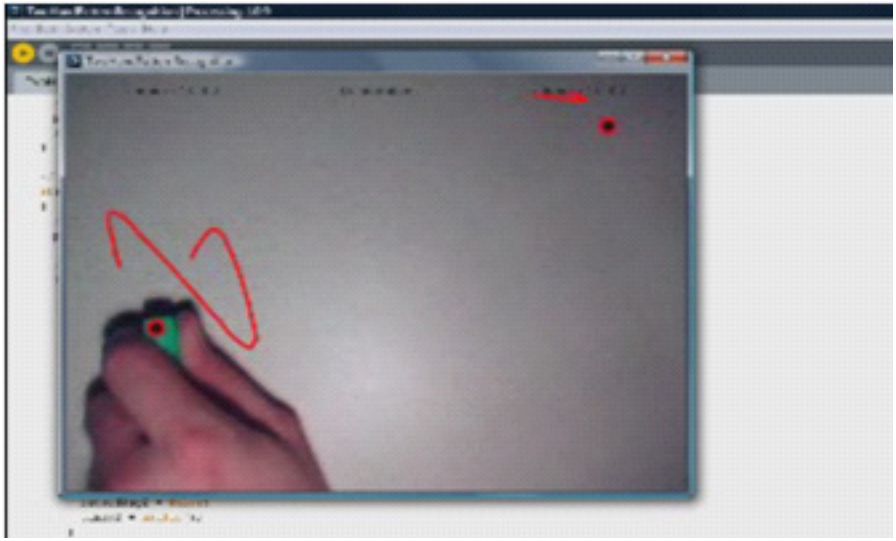


Figure 8. Webcam top view image

User test

Introduction

In this chapter the actual research proposal is described including the actual performance of the tests (mainly in the paragraph about the measurement procedure). They were performed at primary schools with normally developed children because autistic children weren't accessible in December. The time needed to implement the platform reduced the flexibility in planning the tests to this month. The supervisors at the schools with autistic children regarded December as too busy for testing. So one pilot test and one actual test of 6 rounds was performed at different ordinary primary schools.

The research proposal described below (Study design paragraph) is developed for a test with autistic children though. Therefore it was agreed with the autism experts at Sint Marie that a test with autistic children is going to be performed after this writing in January.

Study design

The population targeted for this study is children with an autistic spectrum disorder in the age of 6 to 8. They have to be familiar with participating in "cooperative play" [11]. The research of Parten shows that between the age of 3 and 4 children start playing together in an organized way. So it can be assumed that at the age of 6 most of the children are familiar with cooperative play. The children also need to be familiar with the concept of teaching, which is as well the case at the age of 6 because they generally have 2 years of experience with having a teacher. The complexity of the collaboration task the children get is designed for this age group.

A cross-sectional study will be performed where the social interaction among two children is observed and recorded on video while performing a collaboration task. The coupling of the children has to be performed by the supervisors at school because they know the children and can assess who can work together. The actual test is performed at school because there a large group of the intended study population is available on a

daily basis. The purpose and procedure of the study will be explained to and assessed by the supervisor at school and an appointment will be made to perform the study.

The children will be brought to a separate classroom by their supervisor. The goal of the game is explained through an instruction video and rules will be explicitly mentioned. Using a video instead of personal contact increases the internal validity of the study significantly because autistic children are very sensitive to new people. It is important that the children understand that they have to cooperate to complete the game and the rules are important to guide the children's behaviour during the game; making it appropriate and measurable for the platform.

First it is explained that these robots move in a particular pattern, than the robots execute a predefined pattern. The children need to describe this pattern and imitate it with one hand above the table because the robots will forget the pattern at a certain moment. Research by Jahra on initiating cooperative play [12] showed that describing the task in words has a significant benefit in performing the game, therefore the children are asked to describe the pattern they have to make during this demonstration phase. If the children know the pattern the robots will stop moving.

The children are now asked to teach the robots the pattern they've just practiced. They have to collaborate for this task, because the robots only react if the children make the same movement. In three teaching sessions with a different movement pattern the robots will react differently to the hand movements of the children. They can imitate the movement exactly, perform another random movement (counteract) or interpret the movement of the children (enhance). If the children have taught the robots the correct movement pattern the task is completed. When the robots are counteracting the children won't be able to finish the task. Then the task is finished after approximately 3 minutes, this was the maximum time a test took during the pilot study.

The difficulty level of the patterns is dynamic. To keep the children's attention it is useful to raise the difficulty level at every teaching scenario. Pilot tests show that simple shapes like circle, square and triangle differ enough to keep the children challenged. The tests will start with the circle, which is the easiest pattern. The square is then introduced and after that the triangle, which appeared to be the most difficult pattern. Every couple will perform the teaching game three times, where the robots perform the three different interaction types in controlled random order.

The supervisor can stay in the room while the children play the game. He or she can assist the children in performing the intended hand movement, but not in the cooperative element. The researcher will observe the playing children in a separate room and record moments of social interaction. These records can provide guidance for the video analysis afterwards and answer some of the sub questions regarding the nature of the reaction of the children and how they interpret the different interaction behaviours.

Setting

Sint Marie in Eindhoven, with Juliane Cuperus as contact, is "a remedial education centre for research and treatment of children aged from two years upwards, young people and young adults who have problems with communication". They provide treatment ranging from part-time to 24 hours a day. The study population is present on a daily basis and experts with much experience are available.

The autistic children are best addressed at school, because this is a known environment for them and the supervisors are available there. In a separate room the game platform is set up. The most appropriate room is an empty classroom that is known by the children. The amount of new impressions needs to be limited to the game platform.

Both pilot and final test at the regular primary schools were performed in the children's lunch room, although the pilot test had to shift to the gymnastics room after three tests. All children were familiar with the rooms.

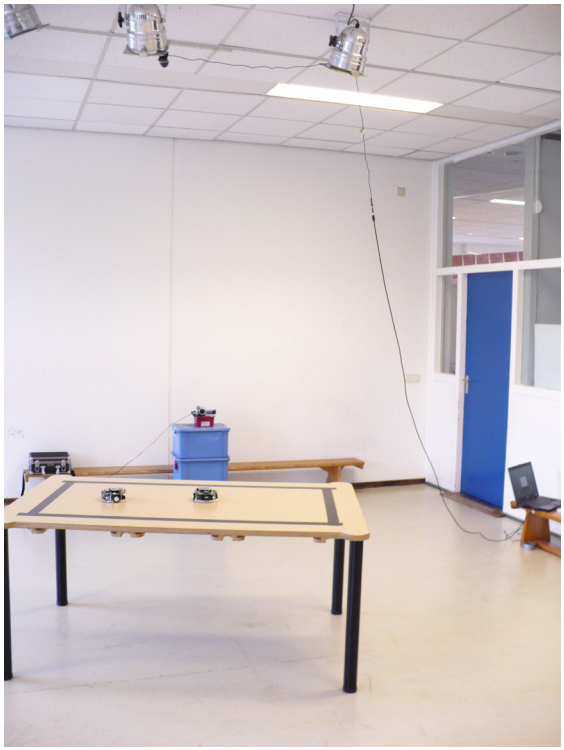


Figure 9. Setting at pilot test (left), setting at final test (right)

Measurement procedure

A cross-sectional study was performed where the social interaction among two children with ASD is observed and recorded on video while performing three collaboration tasks by interacting with a MAS of moving robots on a table. Observation was used to collect data because it is "the most appropriate method to learn about the interaction between people" [13]. The children should not be aware of the tested variables. They had to be engaged in the game. Social interaction is a rather intuitive activity, so the children could not be asked about it after the test. In order to get to know to which extend interaction through movement and teaching robots is an engaging task the children were questioned afterwards through a short open interview. One of the confounds that had to be overcome was the bias of the researcher. This was done by video recording the full test and let it assess on self-initiated social contacts (SISC) by an unbiased expert.

Three types of behaviour of the interactive MAS were compared. They were tested in three tasks of teaching. During one of the sessions the robots directly imitated the movement if both children performed it with one hand above the table. They waited for the children having performed a full movement pattern together and then started imitating; stopping to imitate when the children stopped performing the movement. In another task the robots sabotaged (counteract) the collaborative teaching attempts of the children by performing a random movement pattern different from the pattern the children were performing. During yet another task the robots responded to similar hand movements of the children by extracting one of the possible predefined movement patterns and performing it. In this way the children's movements were enhanced.

Ideally the robots had to be able to recognize the children's hand movement through a video camera providing a top view of the table. This camera is connected to a PC. Through image processing the hands of the children had to be detected and their movement recorded in terms of place, speed and acceleration. The children were delegated to opposite sides of the table and summoned to hold one hand above the table to teach the robots the movement. This enabled much easier recognition of the children's hand movements separately. Detected patterns of both children

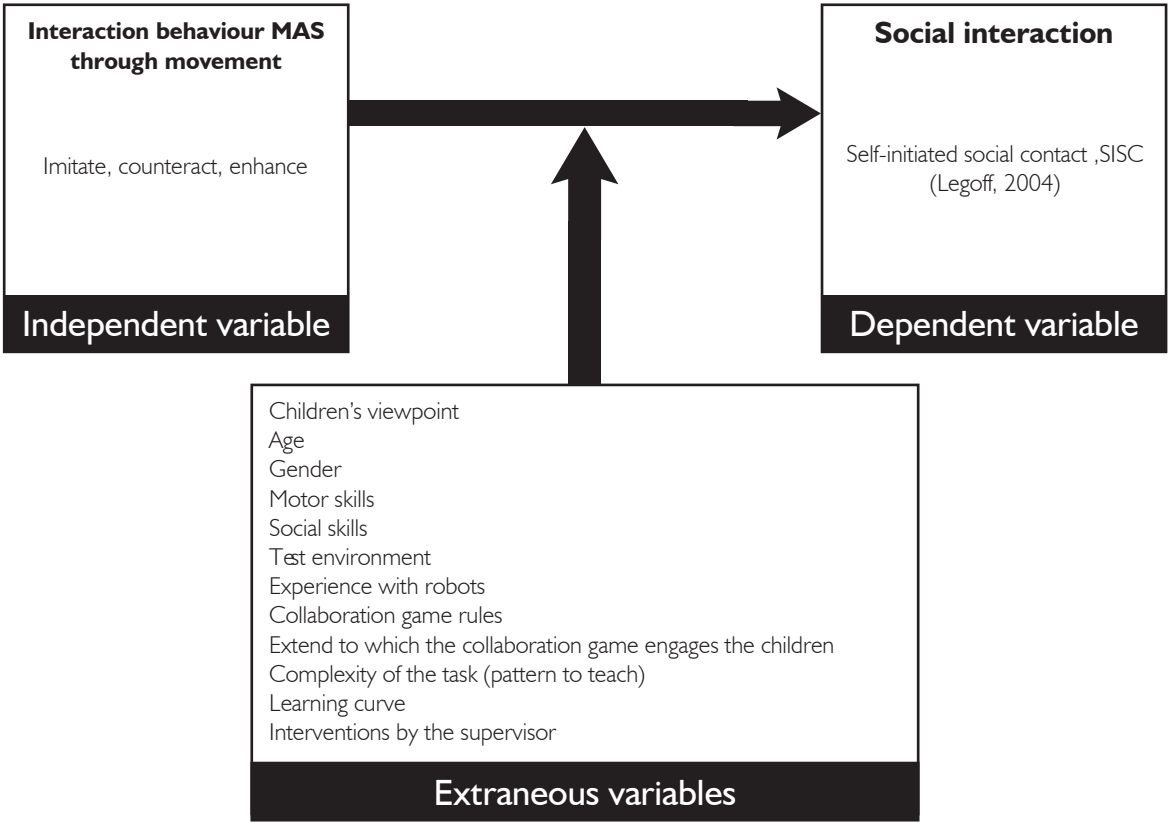


Figure 10. Study variables

could be compared and if they were similar the robots reacted as described before. If the children's movement patterns did not correspond the robots would not react and perform straightforward driving and collision avoidance behaviour.

Eventually the autonomous platform was not accurate enough. Therefore the robots were steered through the laptop by the researcher, mimicking the robots' behaviour. This didn't change anything to the implementation of the interaction scenario, but did decrease the reliability of the implementation due to the interpretation of the person steering the robots. For the test with the autistic children it is important that the steering of the robots happens in a different room, avoiding distraction from the children. This can be done through the webcam that was originally installed for the autonomous system pattern recognition system.

Because the three different interaction behaviours were tested directly after each other it was most likely that the following sessions were influenced by the previous one. Therefore it was important that at least all different orders were tested once. There were six possible orders so at least six tests were needed for a confident result. This corresponds with 12 children that had to perform the test. Due to time limitations it was not possible to test with more children, because it would at least double the amount of video analyzing time.

Imitate	counteract	enhance
Imitate	enhance	counteract
Enhance	imitate	counteract
Enhance	counteract	imitate
Counteract	imitate	enhance
Counteract	enhance	imitate

During each teaching task the amount of SISC was measured. This measurement tool is developed by Legoff to measure social interaction during therapy. In his research on using Lego© as a therapeutic medium social interaction is distinguished in three elements: "(1) initiation of social contact with peers, reflective of social interest and motivation for social contact; (2) duration of social interaction, which reflects the development of communication and play skills; and (3) decreases in autistic aloofness and rigidity, with development of age-appropriate social and play behaviours." For this study mainly the first

element (SISC) is useful because element 2 and 3 are interesting for longitudinal studies with more than one contact moment. The duration of social interaction was nonetheless considered to be interesting because it does reflect a difference in quality of the interaction between the children.

A SISC is counted if it meets the following criteria:
"(1) it was unprompted and spontaneous;
(2) it was not part of a daily routine or required activity;
(3) it involved either verbal or nonverbal communication or a clear attempt to communicate with a peer;
(4) the peer had to be of approximately the same age or developmental level as the subject (i.e., not a much older or younger child); and
(5) it was not a reciprocal response to another child's approach." [1]

SISC is an easy way to measure social interaction; suiting planning for data analysis. The downside is that it heavily reduces the quality of data. Therefore each recorded task was cut into chunks of 15 seconds, which is the longest social interaction measured between the children from the pilot study. Each chunk was rated on the length of SISC. This rating is divided in four categories of social contact with the SISC criteria:

1. No social contact
2. Short social contact (1 word or sentence)
3. Medium social contact (more than 1 sentence, less than 10 sec.)
4. Long social contact (10 sec. or more)

It would also be useful to code the content of the social interaction (in short terms) besides coding the intensity of SISC. This can provide some qualitative insight in the effect of the robots' behaviour on the children's communication.

In previous studies a duration of 10 minutes per child (or couple of children) appeared to be successful in terms of getting familiar with the game and having some time to play with it. The pilot test showed that each session takes approximately 15 minutes. Each task takes about 2 to 3 minutes. In combination with: the short explanation at the start, the demonstration of the movement during and

short questioning at the end of the test; this adds up to 15 minutes. The children start losing concentration after 15 minutes, so it was not possible to increase the amount of tasks. With 6 tests to perform this could be performed in a morning (from 9h to 12h).

Sampling

The study population is rather specific and homogeneous. The main differences within the population occur on the level of development of the children. For the pilot test at a primary school 2 couples were drawn from three different groups (group 3, 4 and 5). The couples were made by the corresponding teacher. The pilot study showed that there's a clear difference between children of group 3, 4 or 5. Children of group 3 (age 6 - 7) have difficulties with focussing on the task whereas children of group 5 (age 8 - 9) are more aware of the system. Typically developed children from group 4 (age 7 – 8) appear to be most suitable for the game. They can concentrate on the collaborative task without trying to see through the system.

Many regular primary schools in Eindhoven were contacted to perform the actual test with children in the right age group (age 6 – 8, preferably group 4). Only primary school 'De Hasselbraam' responded positively, offering to test the robots with children from group 3. This was not the ideal target group but at least within the given boundaries.

The main criterion for the sample size relies on the time available for testing and analyzing as explained before. This means that 12 children with ASD and development level of group 4 are required for this study. They are drawn from the Sint Marie educational centre. During a meeting with Juliane Cuperus (head of autism department) and Marleen Vissers (expert on young autistic children) it was decided that the TOM-group (theory of mind) would be most suitable for this test. The children are selected and coupled for participation by the particular supervisors. This is necessary to avoid a low probability of social interaction between the two children on beforehand. Social relations are delicate for children with ASD which makes any random sampling method inappropriate, also due to the low amount of tests that can be done. Besides, it is "common to use purposive sampling to test something about which little is known" [13].

Data analysis

The result of the test was a recorded video accompanied with notes regarding the nature of the children's reactions. First the video was analysed on the amount of SISC per task (Appendix VII). This resulted in 18 values, 6 for every interaction behaviour type (Table 1). To equalize the values they were calculated to an amount of SISC per minute. This kind of data gave a simplified insight in how much social interaction had taken place per interaction scenario (see Figure 11). The figure suggests that counteracting behaviour encourages social interaction most. Visually it seems that the hypothesis can be confirmed. A "Kruskal-Wallis one-way analysis of variance by ranks" was used "for deciding whether the independent samples are from a different population" [14]. As shown in Figure 12 the three scenarios aren't significantly different. So statistically the interaction behaviours aren't different in terms of encouraging social contact, as shown by the overlapping confidence intervals in Figure 11.

Imitate	Counteract	Enhance
0,00	0,47	2,55
2,36	4,55	3,83
2,81	5,81	3,61
6,43	7,35	3,33
2,83	3,82	4,00
2,91	3,26	2,12

Table 1: SISC per minute of 18 tasks

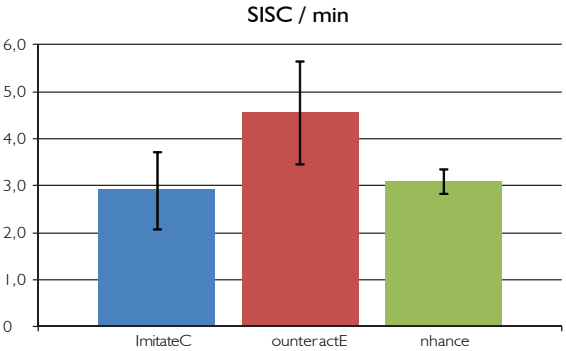


Figure 11. Average SISC per minute for different interaction behaviours with confidence levels

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of SISC is the same across categories of Interaction.	Independent-Samples Kruskal-Wallis Test	.291	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 12. Kruskal-Wallis analysis result

Assessing the social interaction through counting the amount of SISC per minute seemed not to reflect the richness of the data that was recorded. So the length of the interaction was implemented by differently coding the video, as described in the previous chapter in the measurement procedure. The social interaction was categorized; measuring frequencies over time slots.

Figure 13 shows the percentage from the total amount of time slots that a certain kind of social contact was measured. Figure 14 shows the percentage from the total amount of time slots of any kind of measured social contact. It can be seen that the distribution over categories differs per interaction behaviour in Figure 13. Counteracting behaviour seems to provoke shorter social contact whereas imitation provokes longer social contact. In general it can be seen that enhancing behaviour arouses the least social contact and imitation and counteracting don't differ a lot. Interesting is the difference between Figure 11 and Figure 14. It shows that when adding the element of duration the mutual relations change clearly and relate more to the observations.

This procedure provided 118 measurements (N); enough to perform Chi-square tests. The interaction scenarios were compared in couples on the four SISC categories. None of them differed significantly. Categorizing the data in 'no social contact' and 'any social contact' provided no significant result either (Appendix IX) as can be seen in the confidence intervals of Figure 14. It can be said though that enhancing behaviour is most likely to have a different effect on social interaction; imitating and counteracting seem to have a similar effect. In order to measure a significant difference with enhancing behaviour 30 more tests are needed, assuming that the current results are valid.

A validating video analysis by an unbiased student that was familiar with SISC showed that the coding procedure leaves a little room for interpretation. When comparing my observations with the unbiased observations they differ in 5% of the observations. This would have been acceptable if the differences were larger, but in this case this uncertainty makes a big difference. It is interesting to see that these differences are all in scenarios with enhancing behaviour. The other scenarios are as good as similarly rated.

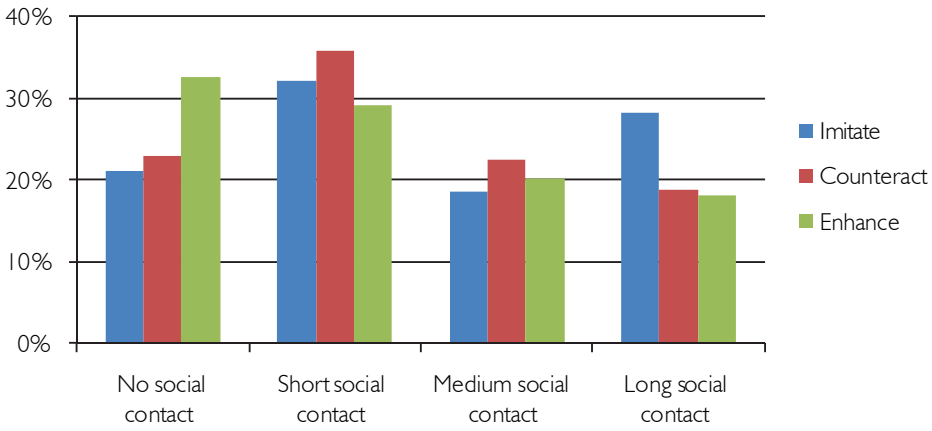


Figure 13. Frequency of categorized social contact per couple

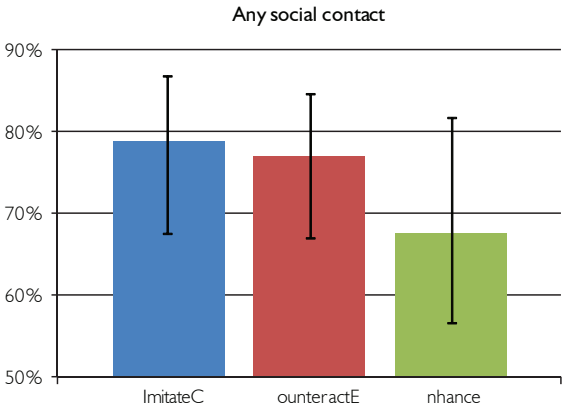


Figure 14. Frequency of any social contact per couple

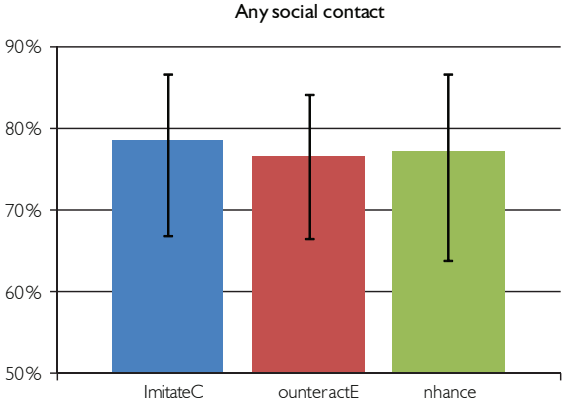


Figure 15. Frequency of any social contact per couple from unbiased observations

The above described ways of gathering the data involved the social interaction per couple because than reciprocal communication was included. The amount of social contact per child was also measured though, to see what the effect of the robots behaviour was on the individuals. This data set (Figure 16 and Figure 17) clearly differed from the one described above. Figure 16 shows a fairly even distribution of categories of social contact over the interaction behaviours. The only conspicuous bar is the amount of short social contact with enhancing behaviour.

Also when looking at the social contact per individual child the interaction behaviours don't differ significantly (Appendix X). The significances do show that again enhancing behaviour is most likely to have a different effect on social interaction.

When comparing Figure 14 and Figure 17, it appears that they to show opposite results. In both graphs it can be seen that enhancing behaviour differs most. This is confirmed by the calculated significances. The ones related to comparing imitation and counteracting, have more or less the same effect on the social interaction between the children because $\alpha = 1,0$ or close to it. Enhancing behaviour can have a different effect on the social interaction although the direction is unclear. In this case it appears that enhancing behaviour slightly encourages social interaction for the individual child whereas the amount of social interaction between the couples in general is reduced.

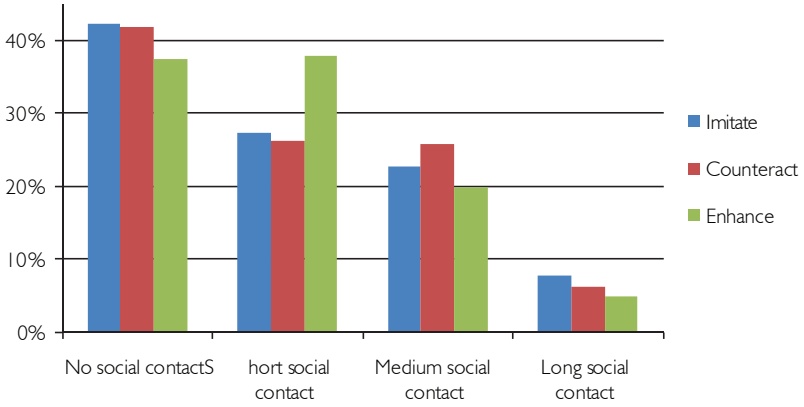


Figure 16. Frequency of categories of social contact per child

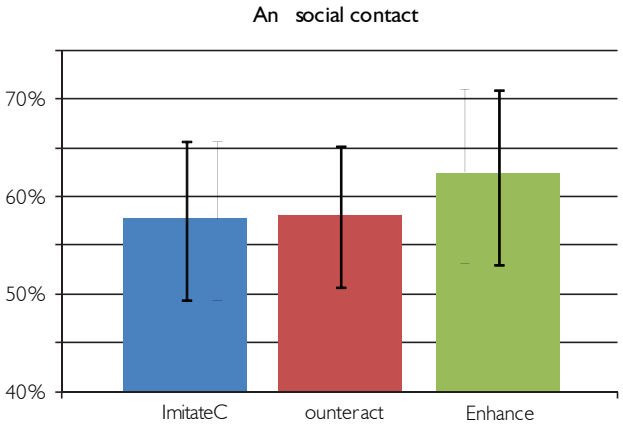


Figure 17. Frequency of any social contact per child

Conclusion

The quantitative data shows that there's no significant difference between the interaction behaviours. So the relation between the MAS interaction behaviour and the amount of social interaction occurring during a collaborative task can't be quantified further.

The fact that on the qualitative level the graphs suggest a different effect on social interaction with enhancing behaviour is interesting, as well as the suggested similarity between imitation and counteracting. The negative effect of enhancing behaviour on the social interaction by a couple of children (+/- 10%) clearly outweighs the positive effect on the individual child (+/- 5%). But nothing can be said with much certainty on this matter because the unbiased observations changed the outcome enough to shift the shape of the histogram.

The most valid explanation for the differences and similarities between the interaction behaviours should be sought in the implementation of these behaviours. When looking at the reactions of the children while playing they found the robots 'bad listeners' mainly with the counteracting behaviour, but also during imitating behaviour. Sometimes the robots didn't listen because the children weren't cooperating but unclear performance of the patterns was also a big factor. The overriding collision avoidance blurred the pattern performance. So the robots were actually performing the movement but this wasn't recognized by the children. With enhancing behaviour the right pattern was performed much quicker, increasing the chance that one of the robots would perform a recognizable pattern.

So it appears that imitation behaviour was experienced as counteracting behaviour. The fact that enhancing behaviour seems to discourage social interaction therefore strengthens the assumption that counteracting behaviour encourages social interaction.

The fact that most of the children were talking about the robots 'listening' or not and reacted surprised if one robot did perform a

movement correctly shows that the task was engaging. Some children left their place at the table and started running around it trying to get the attention of particular robots. The amount of social interaction seemed to increase with more movement. The children that sat calmly at opposite sides of the table clearly interacted less with each other and the robots. So it is possible to assume that interaction through movement enhances social interaction. It is most likely that this can be verified through existing literature.

The hypothesis that counteracting behaviour would enhance social interaction most was derived from the results of the test with the lighting blocks [6]. There it was assumed that imitating interaction between the blocks would encourage social interaction, but eventually counteracting behaviour provoked discussion between the children resulting in more social contact. It appears that this holds also for interaction through movement and might be the case for interaction during collaborative tasks in general.

The results of this study do not lead to a relation between the explorative character of multi-agent games and the enhancement of social interaction by the MAS. Exploration actually seems to make them less communicative because than the children get occupied with exploring the system individually.

Discussion

The test provides interesting starting-points for further investigation on interaction through movement with multi-agent systems. It seems that in this case counteracting and enhancing interaction behaviour were rightfully implemented. For correct implementation of imitation behaviour the platform needs further technological development.

Using the TiViPE software [8] for this appears to be a promising solution in terms of image processing. It was unfortunate that the implementation for this project didn't suit the time schedule. But not only is the image processing a factor in creating confident imitation behaviour. The operation resolution of the AdMoVeo is not accurate enough for exact imitation. So solutions for direct translation of human movements to robot movements have to be investigated as well. It is likely that there are ways to simulate imitation. It would be interesting to investigate this imitation simulation with AdMoVeo robots. How can human movements be directly imitated in an abstracted way?

In this study the factor of imitation was mainly in the collaborative element in the game. This is what the experts at Sint Marie found most interesting, and saw as a possible supplement to an existing education method regarding imitation for TOM-groups (theory of mind). Therefore it is still interesting to perform the test with autistic children and see how the different interaction behaviours affect the social interaction.

For this test with autistic children it is important that the method with a video introduction is implemented to avoid the researcher's presence in the room. It was clear that the presence of me (as the researcher) during the tests at the primary schools had an impact on the data. Some children were more occupied with talking to me then performing the task. This had a clear negative effect on the validity of the test.

Another distracting factor seemed to be the emergent behaviour of the moving robots in the social interaction between normally developed children of 6 years old. The continuous motion of the robots requires full

attention, resulting in difficulties with focussing on their collaborative task. A difference can be seen with older children (7 or 8 year old): they have less difficulty with focussing on the task. So for children younger than the age of 7 it seems not advisable to implement a moving multi-agent system for educative tasks.

An interesting experiment following this study would be to change the scale of interaction. In this study the space was restricted to a table top. This was mainly done for practical reasons, making technical implementation feasible. But it would be interesting to investigate the same game in an empty room with more robots driving on the floor and the children performing the movements with their total body. This was the initial idea for the described interaction scenario options. It would improve the performance of patterns by the robots because there would be less overriding collision behaviour. The setup that was tested now was too small for clear pattern performance, although the recognition of patterns is generally easier for autistic children and overriding collision behaviour might less of a problem.

Acknowledgements

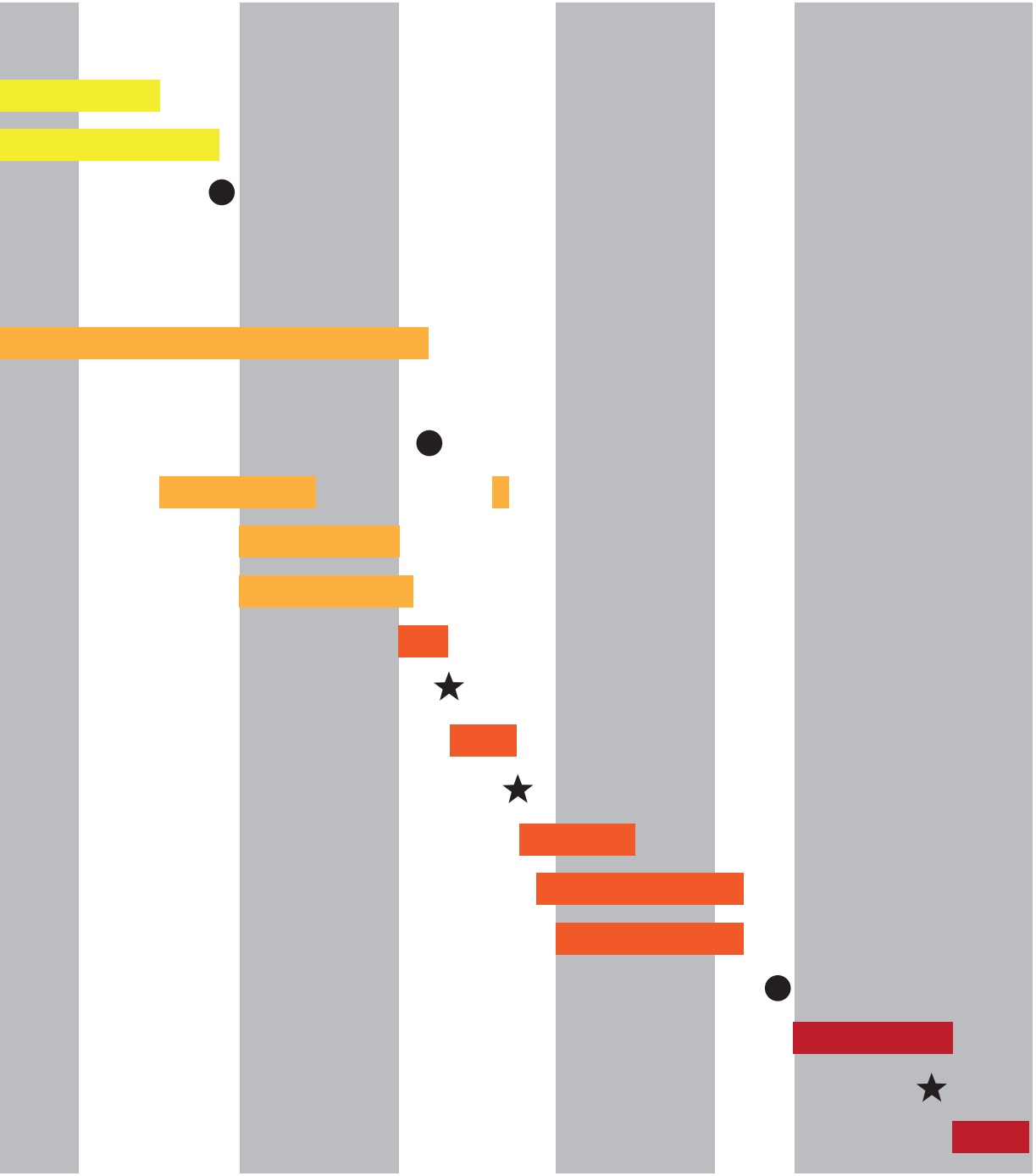
To acquire the knowledge to perform this research project many experts from within the faculty but also from outside helped me, thanks for that. I also want to thank the teachers and children that were involved in the tests from the two primary schools: ‘De Tweesprong’ in Breda and ‘De Hasselbraam’ in Eindhoven.

References

1. Use of LEGO(c) as a Therapeutic Medium for Improving. **LeGoff, Daniel B.** No. 5, s.l. : Springer Science+Business Media, Inc., October 2004, Journal of Autism and Developmental Disorders, Vol. Vol. 34.
2. Modeling emotional movements for design of social games with robots. **Barakova, E.** 2009, Eindhoven University of Technology.
3. Expressing and interpreting emotional movements in social games with robots. Barakova, E and Lourens, T. Eindhoven University of Technology : s.n., 2009.
4. Using an emergent system concept in designing interactive games for autistic children. **Barakova, E, et al.** Eindhoven : s.n., 2007. Bekker, T et al. Proc. Of IDC 07.
5. From spreading of behaviour to dyadic interaction - a robot learns what to imitate. **Barakova, E and Vanderelst, D.** s.l. : International Journal of Intelligent Systems, 2009, Vol. In press.
6. Social training of autistic children with interactive intelligent agents. **Barakova, E, Gilessen, J and Feijs, L. I**, Eindhoven : Imperial College Press, 2009, Journal of Integrative Neuroscience, Vol. 8.
7. **Alers, S.** AdMoVeo: an Educational Arduino Robot. [Online] Department of Industrial Design, Eindhoven University, 2009. http://www.admoveo.nl/.
8. **Lourens, T.** TiViPE. [Online] TiViPE, 2009. [Cited: 31 December 2009.] http://www.tivipe.com/.
9. **de Klein, R.** The code project: your development resource. Serial library for C++. [Online] 13 November 2003. [Cited: 10 November 2009.] http://www.codeproject.com/KB/system/serial.aspx.
10. **Wobbrock, J, Wilson, A and Yang, L.** $1 Unistrok Recognizer in JavaScript. [Online] University of Washington and Microsoft Research, 2007. [Cited: 5 December 2009.] http://depts.washington.edu/aimgroup/proj/dollar/.
11. Social Participation Among Pre-School Children. **Parten, Mildred B.** 3, Minnesota : University of Minnesota, October 1932, The Journal of Abnormal and Social Psychology, Vol. 27.
12. Teaching children with autism to initiate and sustain cooperative play. **Jahra, E, Eldevika, S and Eikesethb, S. Akershus** : Elsevier Science Ltd, 2000, Research in Developmental Disabilities, Vol. 21, pp. 151–169. S0891-4222(00)00031-7.
13. **Kumar, Ranjit**. Research Methodology: A Step-by-Step Guide for Beginners. London : SAGE publications, 1999. 0-7619-6213-1.
14. **Siegel, S and Castellan Jr., N J.** Nonparametric statistics for the behavioural sciences. Singapore : McGraw-Hill Book Co., 1988. 0-07-100326-6.

Appendix

I. Work schedule



II. AdMoVeo code

```
// AdMoVeo digital pins
#define encoderRight_Pin  2
#define buzzer_Pin  3
#define encoderLeft_Pin  4
#define speedRight_Pin  5
#define speedLeft_Pin  6
#define dirRight_Pin  7
#define dirLeft_Pin  8
#define ledBlue_Pin  9
#define ledRed_Pin  10
#define ledGreen_Pin  11
#define leftRight_Pin  12
#define frontRear_Pin  13
// AdMoVeo analog pins
#define line_Pin  2
#define distance_Pin  3
// AdMoVeo variables
#define LEFT   0
#define RIGHT  1
#define BOTH   2
#define FRONT  3
#define NONE   4
#define LINE   5
#define DISTANCE 8
// Behaviour variables
int leftinput = 0;
int rightinput = 0;
int lineLeft = 0;
int lineRight = 0;
int distanceFront = 0;
int distanceLeft = 0;
int distanceRight = 0;
int startLineLeft = 0;
int startLineRight = 0;
int lineThreshold = 175;
int collisionThreshold = 900;
int lineTime = 200;
int collisionTime = 300;
unsigned long moveTimer = millis();
char rightPattern = 'q';
char play = '0';
boolean demo = true;
boolean patternstart = true;
//XBee variables
char values[5] = {
 '0','0','0','0'};
char message;

void setup(){
  // Serial communication configuration
  Serial.begin(57600);
  // AdMoVeo minimal pin configuration
pinMode(leftRight_Pin, OUTPUT);
digitalWrite(leftRight_Pin, HIGH);
pinMode(frontRear_Pin, OUTPUT);
digitalWrite(frontRear_Pin, HIGH);

pinMode(dirRight_Pin, OUTPUT);
digitalWrite(dirRight_Pin, HIGH);
pinMode(dirLeft_Pin, OUTPUT);
digitalWrite(dirLeft_Pin, HIGH);
analogWrite(speedRight_Pin, 0);       // motor STOP
analogWrite(speedLeft_Pin, 0);        // motor STOP

  startLineLeft = readSensor(LINE,LEFT);
  startLineRight = readSensor(LINE,RIGHT);
  analogWrite(buzzer_Pin,100);
  delay(50);
  analogWrite(buzzer_Pin,0);
  Serial.println("setup");
}

void loop(){
  // Waiting for the goal pattern
  while(rightPattern == 'q'){
    play = '0';
    if (Serial.available() > 0) {
      message = Serial.read();
     if (message == 'c' || message == 't' || message
== 'x'){
       rightPattern = message;
      }
     }
    }
    if (play != 'p'){
     if (Serial.available() > 0) {
       play = Serial.read();
     }
     demo = true;
    }
    else {
     demo = false;
    }
    decideMove(checkSensors());
}

byte checkSensors(){
  lineLeft = readSensor(LINE,LEFT);
  lineRight = readSensor(LINE,RIGHT);
  distanceFront = readSensor(DISTANCE,FRONT);
  distanceLeft = readSensor(DISTANCE,LEFT);
  distanceRight = readSensor(DISTANCE,RIGHT);

  if(lineLeft > startLineLeft + lineThreshold ||
    lineLeft < startLineLeft - lineThreshold ||
    lineRight > startLineRight + lineThreshold ||
    lineRight < startLineRight - lineThreshold){
    // robot drives almost from the table
    return 0;
  }
  if(distanceFront < collisionThreshold ||
    distanceLeft < collisionThreshold ||
    distanceRight < collisionThreshold){
    // collision is close
    return 1;
  }
}
void decideMove(byte moves){
  switch(moves){
  case 0:
    //Serial.println("avoidLine");
    analogWrite(ledGreen_Pin,0);
    analogWrite(ledBlue_Pin,255);
```

```
    analogWrite(ledRed_Pin,0);
    avoidLine(lineLeft > startLineLeft + lineThreshold
     || lineLeft < startLineLeft - lineThreshold);
    break;
   case 1:
    //Serial.println("avoidCollision");
    analogWrite(ledGreen_Pin,0);
    analogWrite(ledBlue_Pin,255);
    analogWrite(ledRed_Pin,0);
    avoidCollision(distanceFront < collisionThreshold,
    distanceLeft < collisionThreshold,
    distanceRight < collisionThreshold);
    break;
   default:
    if (demo){
     if (rightPattern == 'c'){
      setColor('g');
      Drive(120,230);
     }
     if (rightPattern == 't'){
      setColor('g');
      moveTriangle();
     }
     if (rightPattern == 'x'){
      setColor('g');
      moveSquare();
     }
    }
    else {
     serialDrive();
    }
    break;
  }
}

void serialDrive()
{
  // Receive pattern character and move
  if (Serial.available() > 0) {
    char prevMessage = message;

    message = Serial.read();
    if(prevMessage != message){
     patternstart = true;
    }
   }
   if (message == 'c'){
    //Serial.println("Circling");
    if(message == rightPattern){
     setColor('g');
    }
    else {
     setColor('r');
    }
    Drive(120,230);
   }
   if (message == 't'){
    if(message == rightPattern){
     setColor('g');
    }
    else {
     setColor('r');
    }
    moveTriangle();
   }
   if (message == 'x'){
    if(message == rightPattern){
     setColor('g');
    }
    else {
     setColor('r');
    }
    moveSquare();
   }
   if (message == 'n'){
    //Serial.println("n");
    setColor('o');
    Drive(170,170);
   }
   if (message == 'q'){
    rightPattern = message;
    setColor('o');
    Drive(0,0);
   }
   if (message == 'p'){
    setColor('b');
    Drive(0,0);
    Serial.print(".");
   }
  }

void Drive(int LeftSpeed, int RightSpeed)
{
  //set direction
  if (LeftSpeed >= 0) digitalWrite(dirLeft_Pin, HIGH);
//forward
   else digitalWrite(dirLeft_Pin, LOW);                    //
backward
      if (RightSpeed >=0) digitalWrite(dirRight_Pin,
HIGH); //forward
   else digitalWrite(dirRight_Pin, LOW);                   //
backward
  // set speed
  analogWrite(speedLeft_Pin, abs(LeftSpeed));
  analogWrite(speedRight_Pin, abs(RightSpeed));
}

void setColor(char color){
  // Receive pattern character and move
  if (color == 'r'){
    analogWrite(ledGreen_Pin,0);
    analogWrite(ledBlue_Pin,0);
    analogWrite(ledRed_Pin,255);
  }
  if (color == 'g'){
    analogWrite(ledGreen_Pin,255);
    analogWrite(ledBlue_Pin,0);
    analogWrite(ledRed_Pin,0);
  }
  if (color == 'b'){
    analogWrite(ledGreen_Pin,0);
    analogWrite(ledBlue_Pin,255);
    analogWrite(ledRed_Pin,0);
  }
  if (color == 'o'){
```

```
    analogWrite(ledGreen_Pin,0);
    analogWrite(ledBlue_Pin,0);
    analogWrite(ledRed_Pin,0);
  }
}

void avoidLine(boolean left){
  // make a move to avoid the table border
  if (left){
    Drive(255,-255);
    delay(lineTime);
  }
  else{
    Drive(-255,255);
    delay(lineTime);
  }
}

void avoidCollision(boolean front, boolean left,
boolean right){
  // make a move to avoid collision
  if(front){
    Drive(-255,-255);
    delay(collisionTime/2);
    Drive(-255,255);
    delay(collisionTime);
  }
  if(right){
    Drive(-255,255);
    delay(collisionTime/2);
  }
  if(left){
    Drive(255,-255);
    delay(collisionTime/2);
  }
}

void setChannel(int selection)
{
  switch (selection){
  case LEFT:
    digitalWrite(frontRear_Pin,LOW);
    digitalWrite(leftRight_Pin,LOW);
    break;
  case RIGHT:
    digitalWrite(frontRear_Pin,LOW);
    digitalWrite(leftRight_Pin,HIGH);
    break;
  case FRONT:
    digitalWrite(frontRear_Pin,HIGH);
    digitalWrite(leftRight_Pin,LOW);
    break;
  case NONE:
    digitalWrite(frontRear_Pin,HIGH);
    digitalWrite(leftRight_Pin,HIGH);
    break;
  default:
    digitalWrite(frontRear_Pin,HIGH);
    digitalWrite(leftRight_Pin,HIGH);
    break;
   }
}

int readSensor(int Sensor, int Side)
{
  int returnData;
  setChannel(Side);     // activate measurement channel
  switch (Sensor){
  case LINE:
    returnData = analogRead(line_Pin);
    break;
  case LIGHT:
    returnData = analogRead(light_Pin);
    break;
  case SOUND:
    returnData = analogRead(sound_Pin);
    break;
  case DISTANCE:
    delay[1];                    // delay needed for reliable
measurement infrared
    returnData = analogRead(distance_Pin);
    break;
  }
   setChannel(NONE);        // set channel to unused
channel, for power reasons
  return returnData;
}

void moveTriangle(){
  int forward = 800;
  int turn = 380;

  if (patternstart){
    moveTimer = millis();
    patternstart = false;
  }
  if(millis() - moveTimer < forward){
    Drive(170,170);
  }
  else if(millis() - moveTimer < forward+turn){
    Drive(-170,170);
  }
  else {
    moveTimer = millis();
  }
}

void moveSquare(){
  int forward = 800;
  int turn = 280;

  if (patternstart){
    moveTimer = millis();
    patternstart = false;
  }
  if(millis() - moveTimer < forward){
    Drive(170,170);
    //Serial.println("forward");
  }
  else if(millis() - moveTimer < forward+turn){
    Drive(-170,170);
  }
  else {
    moveTimer = millis();
  }
}
```

III. Image processing code

```
import processing.video.*;
import processing.serial.*;

// Variable for capture device
Capture video;
color trackColor;
color trackColor2;
int[] arrayX1 = {
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int[] arrayY1 = {
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int trackX1 = 0;
int trackY1 = 0;
int[] arrayX2 = {
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int[] arrayY2 = {
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int trackX2 = 0;
int trackY2 = 0;

Serial myPort;

// Dollar, from http://depts.washington.edu/aimgroup/
proj/dollar/dollar.js
// http://depts.washington.edu/aimgroup/proj/dollar/
//
// Recognizer class constants
//
int NumTemplates = 16;
int NumPoints = 64;
float SquareSize = 250.0;
float HalfDiagonal = 0.5 * sqrt(250.0 * 250.0 + 250.0
* 250.0);
float AngleRange = 90.0;
float AnglePrecision = 2.0;
float Phi = 0.5 * (-1.0 + sqrt(5.0)); // Golden Ratio

String name1 = "unknown";
String ratio1 = "1.0";
String score1 = "0.0";
String name2 = "unknown";
String ratio2 = "1.0";
String score2 = "0.0";
int timer1;
int timelength1 = 4000;
int timer2;
int timelength2 = 3000;
int moveTimer;

float threshold = 0.8; // or 0.9 for imitation
//float threshold = 0.6; // for enhancing

Recognizer recognizer;
Recorder recorder;
Result result = null;
PFont font;

void setup()
{
 size(640, 480);

 myPort = new Serial(this, "COM7", 57600);

 frameRate(30);
 colorMode(RGB,255,255,255,100);
 // Using the default capture device
 video = new Capture(this, width, height, 30);
 trackColor = color(170,95,103); // Start off tracking
for skin colour
 trackColor2 = color(170,95,103); // Start off tracking
for skin colour

 recognizer = new Recognizer();
 recorder = new Recorder();

 noFill();
 smooth();
 strokeWeight(4.0);
 stroke(0);
 font = loadFont("ArialMT-12.vlw");
 textFont(font);

 timer1 = millis();
 timer2 = millis();
}

void draw()
{
 colorTracking();
 recorder.update();
 recorder.draw();
 //drawPattern(0,255,255,0);
 //drawPattern(1,0,255,255);
 //drawPattern(2,255,0,255);

 if( result != null)
 {
  textAlign(CENTER, CENTER);
  fill( color( 255));

  name1 = recorder.name1;
  ratio1 = str(recorder.ratio1);
  score1 = str(recorder.score1);
  name2 = recorder.name2;
  ratio2 = str(recorder.ratio2);
  score2 = str(recorder.score2);

   text( name1 + " - " + ratio1 + ", " + score1, 10,
10, 200, 20);
  text( name2 + " - " + ratio2 + ", " + score2, 430,
10, 200, 20);


  if(name1 == name2){
   text("Collaboration", 220, 10, 200, 20);
   if(name1 == "circle"){
    myPort.write('c');
    /* remove for counteracting
     float number = random(-1,1);
     if(number >=0){
     myPort.write('t');
```

```
      }
      else {
      myPort.write('x');
      }
      */       //remove for counteracting
    }
    if(name1 == "triangle"){
     myPort.write('t');
     /* remove for counteracting
      float number = random(-1,1);
      if(number >=0){
      myPort.write('c');
      }
      else {
      myPort.write('x');
      }
      */       //remove for counteracting
    }
    if(name1 == "x"){
     myPort.write('x');
     /* remove for counteracting
      float number = random(-1,1);
      if(number >=0){
      myPort.write('c');
      }
      else {
      myPort.write('t');
      }
      */       //remove for counteracting
    }
    if(name1 == "- none - "){
     myPort.write('n');
    }
   }
  }
}

void drawPattern(int n, int r, int g, int b){
  color c = color(r, g, b);
   for( int i = 1; i < recognizer.Templates[n].Points.
length; i++)
  {
    int shiftx = 640/2;
    int shifty = 480/2;
    stroke( c );
      line( recognizer.Templates[n].Points[i-1].X+shiftx,
recognizer.Templates[n].Points[i-1].Y+shifty,
                recognizer.Templates[n].Points[i].X+shiftx,
recognizer.Templates[n].Points[i].Y+shifty);
  }
}

void mousePressed() {
 // Save color where the mouse is clicked in trackColor
variable
  int loc = mouseX + mouseY*video.width;
  if (mouseX < video.width/2){
    trackColor = video.pixels[loc];
  }
  else {
    trackColor2 = video.pixels[loc];
  }
  recorder.points1 = new Point[0];
  recorder.points2 = new Point[0];
  recorder.recording1 = true;
  recorder.recording2 = true;
}

void keyPressed() {
  myPort.write(key);
}

void captureEvent(Capture camera)
{
  camera.read();
}

void colorTracking(){

  loadPixels();

  // Reset tracked coordinates
  trackX1 = 0;
  trackY1 = 0;
  trackX2 = 0;
  trackY2 = 0;

  // Draw the video image on the background
  image(video,0,0);

  // Local variables to track the color
  float closestDiff1 = 500.0f;
  int closestX1 = 0;
  int closestY1 = 0;
  float closestDiff2 = 500.0f;
  int closestX2 = 0;
  int closestY2 = 0;

  // Begin loop to walk through every pixel
  // Left half of the image
  for ( int x = 0; x < video.width/2; x++) {
    for ( int y = 0; y < video.height; y++) {
      int loc = x + y*video.width;
      // What is current color
      color currentColor = video.pixels[loc];
      float r1 = red(currentColor);
      float g1 = green(currentColor);
      float b1 = blue(currentColor);
      float r2 = red(trackColor);
      float g2 = green(trackColor);
      float b2 = blue(trackColor);
      // Using euclidean distance to compare colors
      float d = dist(r1,g1,b1,r2,g2,b2);
       // If current color is more similar to tracked color
than
       // closest color, save current location and current
difference
      if (d < closestDiff1) {
        closestDiff1 = d;
        closestX1 = x;
        closestY1 = y;
      }
    }
  }
  // Right half of the image
  for ( int x = video.width/2; x < video.width; x++) {
```

```
    for ( int y = 0; y < video.height; y++) {
      int loc = x + y*video.width;
      // What is current color
      color currentColor = video.pixels[loc];
      float r1 = red(currentColor);
      float g1 = green(currentColor);
      float b1 = blue(currentColor);
      float r2 = red(trackColor2);
      float g2 = green(trackColor2);
      float b2 = blue(trackColor2);
      // Using euclidean distance to compare colors
      float d = dist(r1,g1,b1,r2,g2,b2);
       // If current color is more similar to tracked color
than
       // closest color, save current location and current
difference
      if (d < closestDiff2) {
        closestDiff2 = d;
        closestX2 = x;
        closestY2 = y;
      }
    }
  }
  // Draw a circle at the tracked pixel
  ellipse(closestX1,closestY1,16,16);
  ellipse(closestX2,closestY2,16,16);

   // Store average tracked points over the latest 10
measurements from left and right hand
       bufMeasure(closestX1, closestY1, closestX2,
closestY2);
}

void bufMeasure(int x1, int y1, int x2, int y2){
  arrayX1 = append(arrayX1, x1);
  arrayY1 = append(arrayY1, y1);
  arrayX2 = append(arrayX2, x2);
  arrayY2 = append(arrayY2, y2);

  // Sum the [measureBuf] latest measured points
  int measureBuf = 15;
  if (arrayX1.length == arrayY1.length){
    if (arrayX1.length > measureBuf){
      arrayX1 = subset(arrayX1,1,measureBuf);
      arrayY1 = subset(arrayY1,1,measureBuf);
    }
  }
  else {
    println("arrayX1 != arrayY1, " + arrayX1.length +
" != " +arrayY1.length);
  }
  if (arrayX2.length == arrayY2.length){
    if (arrayX2.length > measureBuf){
      arrayX2 = subset(arrayX2,1,measureBuf);
      arrayY2 = subset(arrayY2,1,measureBuf);
    }
  }
  else {
    println("arrayX2 != arrayY2, " + arrayX2.length +
" != " +arrayY2.length);
  }

  // Remove extreme measurements
  trackX1 = removeExtreme(arrayX1);
  trackY1 = removeExtreme(arrayY1);
  trackX2 = removeExtreme(arrayX2);
  trackY2 = removeExtreme(arrayY2);

}

int removeExtreme(int[] copiedArray){
  for(int j = 0; j < 5; j++){
    int trackDiff = 0;
    int posDiff = 0;
    int copiedAvg = 0;

    // Get the copied array mean
    for (int i = 0; i < copiedArray.length; i++){
      copiedAvg += copiedArray[i];
    }

    copiedAvg /= copiedArray.length;

     // Calculate the value with maximum difference to
the mean
    for (int i = 0; i < copiedArray.length; i++){
      trackDiff = abs(copiedAvg - copiedArray[i]);
      if(abs(copiedAvg - copiedArray[i]) > trackDiff){
        trackDiff = abs(copiedAvg - copiedArray[i]);
        posDiff = i;
      }
    }
    // Remove the value with maximum difference
      copiedArray[posDiff] = copiedArray[copiedArray.
length-1];
    copiedArray = shorten(copiedArray);
  }

  // Get the final mean
  int copiedAvg = 0;
  for (int i = 0; i < copiedArray.length; i++){
    copiedAvg += copiedArray[i];
  }

  copiedAvg /= copiedArray.length;

  return copiedAvg;
}

// simple class for recording points
class Recorder
{
  Point [] points1;
  Point [] points2;
  boolean recording1;
  boolean recording2;
  String name1;
  float score1;
  float ratio1;
  String name2;
  float score2;
  float ratio2;
```

```
 Recorder()
 {
  points1 = new Point[0];
  points2 = new Point[0];
  recording1 = false;
  recording2 = false;
 }

 void update()
 {
  // Record and recognize left hand
  if( recording1)
  {
   // Record array of points to compare
      points1 = (Point[])append(points1, new Point(
trackX1, trackY1));

   // Pattern recognition of left hand
   if (points1.length > 10){
    // Start pattern recognition if more than 10 points
are recorded
     result = recognizer.Recognize( points1 );
     if (result.Ratio != 1.0) //result.Ratio > threshold
&&
     {
      name1 = result.Name;
      score1 = result.Score;
      ratio1 = result.Ratio;
      recording1 = false;
      timer1 = millis();
     }
     if (millis() - timer1 > timelength1){
      name1 = result.Name;
      score1 = result.Score;
      ratio1 = result.Ratio;
      recording1 = false;
      timer1 = millis();
     }
    }
   }
   else
   {
    // Reset recorded pattern
    points1 = new Point[0];
    recording1 = true;
   }

   // Record and recognize right hand
   if( recording2)
   {
    // Record array of points to compare
      points2 = (Point[])append(points2, new Point(
trackX2, trackY2));

    // Pattern recognition of left hand
    if (points2.length > 10){
     // Start pattern recognition if more than 10 points
are recorded
     result = recognizer.Recognize( points2 );
     if (result.Ratio != 1.0) //result.Ratio > threshold
&&
     {
      name2 = result.Name;
      score2 = result.Score;
      ratio2 = result.Ratio;
      recording2 = false;
      timer2 = millis();
     }
     if (millis() - timer2 > timelength2){
      name2 = result.Name;
      score2 = result.Score;
      ratio2 = result.Ratio;
      recording2 = false;
      timer2 = millis();
     }
    }
   }
   else
   {
    // Reset recorded pattern
    points2 = new Point[0];
    recording2 = true;
   }
  }

 void draw( )
 {
  // Draw the recorded path of the left hand
  color c1 = color(255);
  if( recording1 )
  {
   c1 = color(255, 0, 0);
  }
  if( points1.length > 1)
  {
   for( int i = 1; i < points1.length; i++)
   {
    stroke( c1 );
    line( points1[i-1].X, points1[i-1].Y,
    points1[i ].X, points1[i ].Y);
   }
  }

  // Draw the recorded path of the right hand
  color c2 = color(255);
  if( recording2 )
  {
   c2 = color(255, 0, 0);
  }
  if( points2.length > 1)
  {
   for( int i = 1; i < points2.length; i++)
   {
    stroke( c2 );
    line( points2[i-1].X, points2[i-1].Y,
    points2[i ].X, points2[i ].Y);
   }
  }
 }
}


float Infinity = 1e9;


// What follows here is a translation of the javascript
```

```
to java.
// There is probably a better way to do it, but this
works.

// Base point class.
class Point
{
 float X;
 float Y;
 Point( float x, float y)
 {
  X = x;
  Y = y;
 }

 float distance( Point other)
 {
  return dist( X, Y, other.X, other.Y);
 }
}

class Rectangle
{
 float X;
 float Y;
 float Width;
 float Height;
 Rectangle( float x, float y, float width, float height)
 {
  X = x;
  Y = y;
  Width = width;
  Height = height;
 }
}

// A template holds a name and a set of reduced points
that represent
// a single gesture.
class Template
{
 String Name;
 Point [] Points;
 Template( String name, Point [] points)
 {
  Name = name;
  Points = Resample( points, NumPoints);
  Points = RotateToZero( Points );
  Points = ScaleToSquare( Points, SquareSize);
  Points = TranslateToOrigin( Points );
 }
}

class Result
{
 String Name;
 float Score;
 float Ratio;
 Result( String name, float score, float ratio)
 {
  Name = name;
  Score = score;
  Ratio = ratio;
 }
}


class Recognizer
{
 Template [] Templates = {
 };
 Recognizer()
 {

  // The triangle, circle or rectangle can be recognized

  // triangle
  Point [] point0 = {
       new Point(137,139),new Point(135,141),new
Point(133,144),new Point(132,146),
       new Point(130,149),new Point(128,151),new
Point(126,155),new Point(123,160),
       new Point(120,166),new Point(116,171),new
Point(112,177),new Point(107,183),
       new Point(102,188),new Point(100,191),new
Point(95,195),new Point(90,199),
       new Point(86,203),new Point(82,206),new
Point(80,209),new Point(75,213),
       new Point(73,213),new Point(70,216),new
Point(67,219),new Point(64,221),
       new Point(61,223),new Point(60,225),new
Point(62,226),new Point(65,225),
       new Point(67,226),new Point(74,226),new
Point(77,227),new Point(85,229),
       new Point(91,230),new Point(99,231),new
Point(108,232),new Point(116,233),
       new Point(125,233),new Point(134,234),new
Point(145,233),new Point(153,232),
       new Point(160,233),new Point(170,234),new
Point(177,235),new Point(179,236),
       new Point(186,237),new Point(193,238),new
Point(198,239),new Point(200,237),
       new Point(202,239),new Point(204,238),new
Point(206,234),new Point(205,230),
       new Point(202,222),new Point(197,216),new
Point(192,207),new Point(186,198),
       new Point(179,189),new Point(174,183),new
Point(170,178),new Point(164,171),
       new Point(161,168),new Point(154,160),new
Point(148,155),new Point(143,150),
    new Point(138,148),new Point(136,148)
    };
    AddTemplate("triangle", point0);

  // circle
  Point [] point1 = {
       new Point(127,141),new Point(124,140),new
Point(120,139),new Point(118,139),
       new Point(116,139),new Point(111,140),new
Point(109,141),new Point(104,144),
        new Point(100,147),new Point(96,152),new
Point(93,157),new Point(90,163),
        new Point(87,169),new Point(85,175),new
```

```
Point(83,181),new Point(82,190),
        new Point(82,195),new Point(83,200),new
Point(84,205),new Point(88,213),
        new Point(91,216),new Point(96,219),new
Point(103,222),new Point(108,224),
        new Point(111,224),new Point(120,224),new
Point(133,223),new Point(142,222),
        new Point(152,218),new Point(160,214),new
Point(167,210),new Point(173,204),
        new Point(178,198),new Point(179,196),new
Point(182,188),new Point(182,177),
        new Point(178,167),new Point(170,150),new
Point(163,138),new Point(152,130),
        new Point(143,129),new Point(140,131),new
Point(129,136),new Point(126,139)
    };
    AddTemplate("circle", point1);

  // x
  Point [] point2 = {
        new Point(87,142),new Point(89,145),new
Point(91,148),new Point(93,151),
        new Point(96,155),new Point(98,157),new
Point(100,160),new Point(102,162),
        new Point(106,167),new Point(108,169),new
Point(110,171),new Point(115,177),
        new Point(119,183),new Point(123,189),new
Point(127,193),new Point(129,196),
        new Point(133,200),new Point(137,206),new
Point(140,209),new Point(143,212),
        new Point(146,215),new Point(151,220),new
Point(153,222),new Point(155,223),
        new Point(157,225),new Point(158,223),new
Point(157,218),new Point(155,211),
        new Point(154,208),new Point(152,200),new
Point(150,189),new Point(148,179),
        new Point(147,170),new Point(147,158),new
Point(147,148),new Point(147,141),
        new Point(147,136),new Point(144,135),new
Point(142,137),new Point(140,139),
        new Point(135,145),new Point(131,152),new
Point(124,163),new Point(116,177),
        new Point(108,191),new Point(100,206),new
Point(94,217),new Point(91,222),
        new Point(89,225),new Point(87,226),new
Point(87,224), new Point(87,220),
        new Point(87,216),new Point(87,210),new
Point(87,206),new Point(87,200),
        new Point(87,194),new Point(87,190),new
Point(87,185),new Point(87,182),
        new Point(87,178),new Point(87,172),new
Point(87,169),new Point(87,165),
        new Point(87,161),new Point(87,158),new
Point(87,155),new Point(87,150),
    new Point(87,146),new Point(87,142)
    };
    AddTemplate("x", point2);
  }

  Result Recognize( Point [] points)
  {
   points = Resample( points, NumPoints);
   points = RotateToZero( points );
   points = ScaleToSquare(points, SquareSize);
   points = TranslateToOrigin(points);
   float best = Infinity;
   float sndBest = Infinity;
   int t = -1;
   for( int i = 0; i < Templates.length; i++)
   {
    float d = DistanceAtBestAngle( points, Templates[i],
-AngleRange, AngleRange, AnglePrecision);
    if( d < best )
    {
     sndBest = best;
     best = d;
     t = i;
    }
    else if( d < sndBest)
    {
     sndBest = d;
    }
   }
   float score = 1.0 - (best / HalfDiagonal);
   float otherScore = 1.0 - (sndBest / HalfDiagonal);
   float ratio = otherScore / score;
    // The threshold is arbitrary, and not part of the
original code.
   if( t > -1 && score > threshold)
   {
    return new Result( Templates[t].Name, score, ratio
);
   }
   else
   {
    return new Result( "- none - ", 0.0, 1.0);
   }
  }

  int AddTemplate( String name, Point [] points)
  {
   Templates = (Template []) append( Templates, new
Template(name, points));
   int num = 0;
   for( int i = 0; i < Templates.length; i++)
   {
    if( Templates[ i ].Name == name)
    {
     num++;
    }
   }
   return num;
  }

  void DeleteUserTemplates( )
  {
      Templates = (Template [])subset(Templates, 0,
NumTemplates);
  }

}

float PathLength( Point [] points)
{
  float d = 0.0;
  for( int i = 1; i < points.length; i++)
```

```
 {
  d += points[i-1].distance( points[i]);
 }
 return d;
}

float PathDistance( Point [] pts1, Point [] pts2)
{
 if( pts1.length != pts2.length)
 {
  recorder.recording1 = false;
  recorder.recording2 = false;
   println( "Lengths differ. " + pts1.length + " != " +
pts2.length);
  return Infinity;
 }
 float d = 0.0;
 for( int i = 0; i < pts1.length; i++)
 {
  d += pts1[i].distance( pts2[i]);
 }
 return d / (float)pts1.length;
}

Rectangle BoundingBox( Point [] points)
{
 float minX = Infinity;
 float maxX = -Infinity;
 float minY = Infinity;
 float maxY = -Infinity;

 for( int i = 1; i < points.length; i++)
 {
  minX = min( points[i].X, minX);
  maxX = max( points[i].X, maxX);
  minY = min( points[i].Y, minY);
  maxY = max( points[i].Y, maxY);
 }
  return new Rectangle( minX, minY, maxX - minX,
maxY - minY);
}

Point Centroid( Point [] points)
{
 Point centriod = new Point(0.0, 0.0);
 for( int i = 1; i < points.length; i++)
 {
  centriod.X += points[i].X;
  centriod.Y += points[i].Y;
 }
 centriod.X /= points.length;
 centriod.Y /= points.length;
 return centriod;
}

Point [] RotateBy( Point [] points, float theta)
{
 Point c = Centroid( points );
 float Cos = cos( theta );
 float Sin = sin( theta );

 Point [] newpoints = {
 };
 for( int i = 0; i < points.length; i++)
 {
  float qx = (points[i].X - c.X) * Cos - (points[i].Y - c.Y)
* Sin + c.X;
    float qy = (points[i].X - c.X) * Sin + (points[i].Y -
c.Y) * Cos + c.Y;
      newpoints = (Point[]) append(newpoints, new
Point( qx, qy ));
 }
 return newpoints;
}

Point [] RotateToZero( Point [] points)
{
 Point c = Centroid( points );
 float theta = atan2( c.Y - points[0].Y, c.X - points[0].X);
 return RotateBy( points, -theta);
}

Point [] Resample( Point [] points, int n)
{
 float I = PathLength( points ) / ( (float)n - 1.0 );
 float D = 0.0;
 Point [] newpoints = {
 };
 Stack stack = new Stack();
 for( int i = 0; i < points.length; i++)
 {
  stack.push( points[ points.length - 1 - i]);
 }

 while( !stack.empty())
 {
  Point pt1 = (Point) stack.pop();

  if( stack.empty())
  {
   newpoints = (Point [])append( newpoints, pt1);
   continue;
  }
  Point pt2 = (Point) stack.peek();
  float d = pt1.distance( pt2);
  if( (D + d) >= I)
  {
   float qx = pt1.X + (( I - D ) / d ) * (pt2.X - pt1.X);
   float qy = pt1.Y + (( I - D ) / d ) * (pt2.Y - pt1.Y);
   Point q = new Point( qx, qy);
   newpoints = (Point [])append( newpoints, q);
   stack.push( q );
   D = 0.0;
  }
  else {
   D += d;
  }
 }

 if( newpoints.length == (n - 1) )
 {
   newpoints = (Point [])append( newpoints, points[
points.length - 1 ]);
```

```
  }
  return newpoints;

}

Point [] ScaleToSquare( Point [] points, float sz)
{
  Rectangle B = BoundingBox( points );
  Point [] newpoints = {
  };
  for( int i = 0; i < points.length; i++)
  {
    float qx = points[i].X * (sz / B.Width);
    float qy = points[i].Y * (sz / B.Height);
      newpoints = (Point [])append( newpoints,   new
Point(qx, qy));
  }
  return newpoints;
}

float DistanceAtBestAngle( Point [] points, Template T,
float a, float b, float threshold)
{
  float x1 = Phi * a + (1.0 - Phi) * b;
  float f1 = DistanceAtAngle(points, T, x1);
  float x2 = (1.0 - Phi) * a + Phi * b;
  float f2 = DistanceAtAngle(points, T, x2);
  while( abs( b - a ) > threshold)
  {
    if( f1 < f2 )
    {
      b = x2;
      x2 = x1;
      f2 = f1;
      x1 = Phi * a + (1.0 - Phi) * b;
      f1 = DistanceAtAngle(points, T, x1);
    }
    else
    {
      a = x1;
      x1 = x2;
      f1 = f2;
      x2 = (1.0 - Phi) * a + Phi * b;
      f2 = DistanceAtAngle(points, T, x2);
    }
  }
  return min(f1, f2);
}

float DistanceAtAngle( Point [] points, Template T, float
theta)
{
  Point [] newpoints = RotateBy( points, theta);
  return PathDistance( newpoints, T.Points);
}

Point [] TranslateToOrigin( Point [] points)
{
  Point c = Centroid( points);
  Point [] newpoints = {
  };
  for( int i = -0; i < points.length; i++)
  {
    float qx = points[i].X - c.X;
    float qy = points[i].Y - c.Y;
      newpoints = (Point [])append( newpoints,   new
Point(qx, qy));
  }
  return newpoints;
}
```

IV. Serial communication in C++

```
// ControlAdMoveo.cpp written by Niko Vegt
#define STRICT
#include <tchar.h>
#include <time.h>
#include <stdio.h>
#include <string.h>
#include <windows.h>

#include "Serial.h"

int ShowError (LONG lError, LPCTSTR lptszMessage)
{
	// Generate a message text
	TCHAR tszMessage[256];
	wsprintf(tszMessage,_T("%s\n(error code %d)"), lptszMessage, lError);

	// Display message-box and return with an error-code
	::MessageBox(0,tszMessage,_T("Listener"),	MB_ICONSTOP|MB_
OK);
	return 1;
}

void wait ( int seconds )
{
 clock_t endwait;
 endwait = clock() + seconds * CLOCKS_PER_SEC ;
 while (clock() < endwait) {}
}

int __cdecl _tmain (int /*argc*/, char** /*argv*/)
{
	printf("Setup serial port\n");
   CSerial serial;
	LONG	lLastError = ERROR_SUCCESS;

	bool waiting = true;
   // Attempt to open the serial port (COM8)
	do
	{
		if (waiting)
		{
			printf("Waiting for serial port\n");
			int status = serial.CheckPort(_T("COM8"));
			printf("Serial status: %d\n", status);
			waiting = false;
		}
	}
	while(serial.CheckPort(_T("COM8")) != 0);

   lLastError = serial.Open(_T("COM8"),0,0,false);
	if (lLastError != ERROR_SUCCESS)
		return ::ShowError(serial.GetLastError(), _T("Unable to
open COM-port"));

	printf("Serial port opened\n");

   // Setup the serial port (57600,N81) using hardware handshaking

    serial.Setup(CSerial::EBaud57600,CSerial::EData8,CSerial::EParNone,CSerial::E
Stop1);
   serial.SetupHandshaking(CSerial::EHandshakeXbee);
	serial.SetEventChar(27);

   // The serial port is now ready and we can send/receive data. If
   // the following call blocks, then the other side doesn't support
   // hardware handshaking.

	// Wait for setup from AdMoveo
	printf("Waiting for AdMoveo setup (reset)\n");
	lLastError = serial.WaitEvent();
	if (lLastError != ERROR_SUCCESS)
	   return ::ShowError(serial.GetLastError(), _T("Unable to wait for a
COM-port event."));

	bool fLoop = true;
	while(true)
	{
		do
		{
			// Send motor side
```

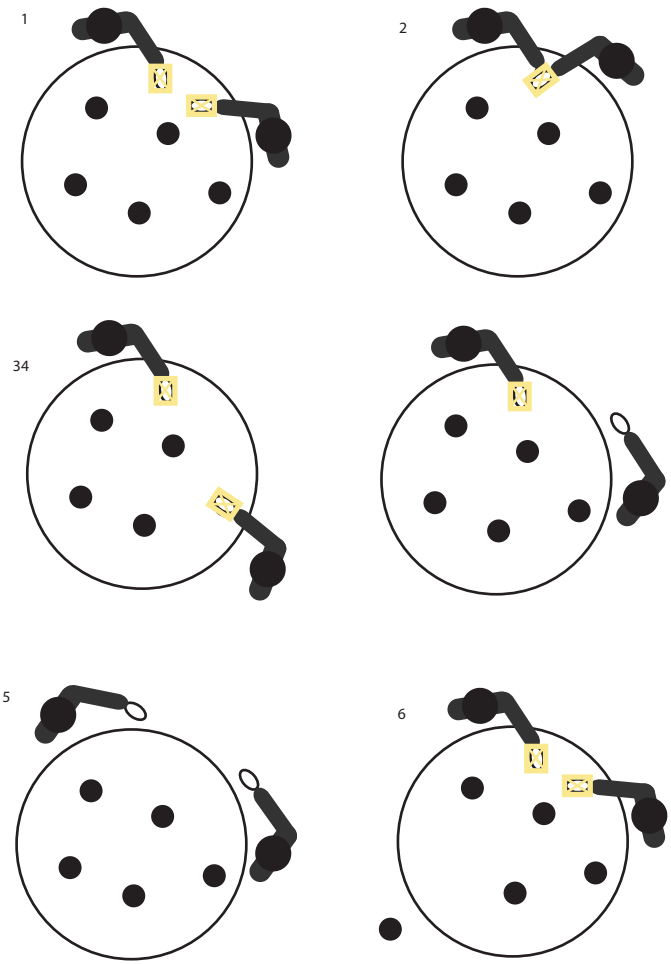
```
                printf("Sending L\n");
                lLastError = serial.Write("l");
                if (lLastError != ERROR_SUCCESS)
                        return  ::ShowError(serial.GetLastError(),
_T("Unable to send data"));
                // Handle data receive event
                if (CSerial::EEventRecv)
                        fLoop = false;
        }
        while(fLoop);

        wait[1];

        do
        {
                // Send motor speed
                printf("Sending 100\n");
                lLastError = serial.Write("150");
                if (lLastError != ERROR_SUCCESS)
                        return  ::ShowError(serial.GetLastError(),
_T("Unable to send data"));
                // Handle data receive event
                if (CSerial::EEventRecv)
                        fLoop = false;
        }
        while(fLoop);

        wait[1];

        do
        {
                // Send motor side
                printf("Sending R\n");
                lLastError = serial.Write("r");
                if (lLastError != ERROR_SUCCESS)
                        return  ::ShowError(serial.GetLastError(),
_T("Unable to send data"));
                // Handle data receive event
                if (CSerial::EEventRecv)
                        fLoop = false;
        }
        while(fLoop);

        wait[1];

        do
        {
                // Send motor speed
                printf("Sending -100\n");
                lLastError = serial.Write("020");
                if (lLastError != ERROR_SUCCESS)
                        return  ::ShowError(serial.GetLastError(),
_T("Unable to send data"));
                // Handle data receive event
                if (CSerial::EEventRecv)
                        fLoop = false;
        }
        while(fLoop);

        wait[1];
    }

    // Close the port again
    serial.Close();
    return 0;
}
```

```
////////////////////////////////////////////////////////////////////////////
//      Serial.h - Definition of the CSerial class
//
//      Copyright (C) 1999-2003 Ramon de Klein (Ramon.de.Klein@ict.nl)
//
// This library is free software; you can redistribute it and/or
// modify it under the terms of the GNU Lesser General Public
// License as published by the Free Software Foundation; either
// version 2.1 of the License, or (at your option) any later version.
//
// This library is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU
```

```
// Lesser General Public License for more details.
//
// You should have received a copy of the GNU Lesser General Public
// License along with this library; if not, write to the Free Software
// Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

        // Handshaking (the changed part)
        typedef enum
        {
                EHandshakeUnknown       = -1,   // Unknown
                EHandshakeOff                   =  0,
// No handshaking
                EHandshakeHardware      =  1,   // Hardware
handshaking (RTS/CTS)
                EHandshakeSoftware      =  2,   Software
handshaking (XON/XOFF)
                EHandshakeXbee                  =  3
// [TU/e] Added for Xbee communication
        }
        EHandshake;


--------------------------------------------------------------------------------
//      Serial.cpp - Implementation of the CSerial class
//
//      Copyright (C) 1999-2003 Ramon de Klein (Ramon.de.Klein@ict.nl)
//
// This library is free software; you can redistribute it and/or
// modify it under the terms of the GNU Lesser General Public
// License as published by the Free Software Foundation; either
// version 2.1 of the License, or (at your option) any later version.
//
// This library is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU
// Lesser General Public License for more details.
//
// You should have received a copy of the GNU Lesser General Public
// License along with this library; if not, write to the Free Software
// Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

////////////////////////////////////////////////////////////////
LONG CSerial::SetupHandshaking (EHandshake eHandshake) // (the changed part
of the library)
{
        // Reset error state
        m_lLastError = ERROR_SUCCESS;

        // Check if the device is open
        if (m_hFile == 0)
        {
                // Set the internal error code
                m_lLastError = ERROR_INVALID_HANDLE;

                // Issue an error and quit
                _RPTF0(_CRT_WARN,"CSerial::SetupHandshaking      -
Device is not opened\n");
                return m_lLastError;
        }

        // Obtain the DCB structure for the device
        CDCB dcb;
        if (!::GetCommState(m_hFile,&dcb))
        {
                // Obtain the error code
                m_lLastError = ::GetLastError();

                // Display a warning
                _RPTF0(_CRT_WARN,"CSerial::SetupHandshaking      -
Unable to obtain DCB information\n");
                return m_lLastError;
        }

        // Set the handshaking flags
        switch (eHandshake)
        {
        case EHandshakeOff:
                dcb.fOutxCtsFlow = false;
// Disable CTS monitoring
                dcb.fOutxDsrFlow = false;
```

```
// Disable DSR monitoring
                        dcb.fDtrControl = DTR_CONTROL_DISABLE;
// Disable DTR monitoring
                        dcb.fOutX = false;
// Disable XON/XOFF for transmission
                        dcb.fInX = false;
// Disable XON/XOFF for receiving
                        dcb.fRtsControl = RTS_CONTROL_DISABLE;
// Disable RTS (Ready To Send)
                        break;

                case EHandshakeHardware:
                        dcb.fOutxCtsFlow = true;
// Enable CTS monitoring
                        dcb.fOutxDsrFlow = true;
// Enable DSR monitoring
                        dcb.fDtrControl = DTR_CONTROL_HANDSHAKE;
// Enable DTR handshaking
                        dcb.fOutX = false;
// Disable XON/XOFF for transmission
                        dcb.fInX = false;
// Disable XON/XOFF for receiving
                        dcb.fRtsControl = RTS_CONTROL_HANDSHAKE;
// Enable RTS handshaking
                        break;

                case EHandshakeSoftware:
                        dcb.fOutxCtsFlow = false;
// Disable CTS (Clear To Send)
                        dcb.fOutxDsrFlow = false;
// Disable DSR (Data Set Ready)
                        dcb.fDtrControl = DTR_CONTROL_DISABLE;
// Disable DTR (Data Terminal Ready)
                        dcb.fOutX = true;
// Enable XON/XOFF for transmission
                        dcb.fInX = true;
// Enable XON/XOFF for receiving
                        dcb.fRtsControl = RTS_CONTROL_DISABLE;
// Disable RTS (Ready To Send)
                        break;

                case EHandshakeXbee:
                        dcb.fOutxCtsFlow = false;
// Disable CTS (Clear To Send)
                        dcb.fOutxDsrFlow = false;
// Disable DSR (Data Set Ready)
                        dcb.fDtrControl = DTR_CONTROL_ENABLE;
// Disable DTR (Data Terminal Ready)
                        dcb.fOutX = true;
// Enable XON/XOFF for transmission
                        dcb.fInX = true;
// Enable XON/XOFF for receiving
                        dcb.fRtsControl = RTS_CONTROL_ENABLE;
// Disable RTS (Ready To Send)
                        break;

                default:
                        // This shouldn't be possible
                        _ASSERTE(false);
                        m_lLastError = E_INVALIDARG;
                        return m_lLastError;
                }

                // Set the new DCB structure
                if (!::SetCommState(m_hFile,&dcb))
                {
                        // Obtain the error code
                        m_lLastError = ::GetLastError();

                        // Display a warning
                        _RPTF0(_CRT_WARN,"CSerial::SetupHandshaking -
Unable to set DCB information\n");
                        return m_lLastError;
                }

                // Return successful
                return m_lLastError;
}
```

V. Image scenarios



VI. Test protocol

Install platform
Webcam
Table with tape (robot vs. hand area)
Lighting
Place video camera
Startup software (for webcam and robot communication)

Prepare for children
    Put robots in waiting mode

Pick up children from classroom

Start recording

Introduce myself and the robots
    They can move
    They can listen to your hands

Explain the game
    Purpose
        Teach the robots a movement
    Means
        Rehearse the movement
        Perform the movement together
    Rules
        Don't touch the robots
        One hand above the table
        Stand at opposite sides of the table

Play the game (with right interaction behaviour)
Demonstrate the movement
    Reset the robots
    Stimulate the children to perform the movement together
    Steer the robots
    Stop the robots if movement is taught or if 3 minutes have passed
    Repeat the task

Play the game again (with right interaction behaviour)
    Take pictures during the game

And again (with right interaction behaviour)

Ask how they liked the game

Stop recording (check video camera status)

Thank the children for cooperation and bring back to classroom

---

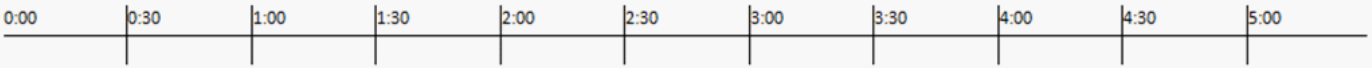
VII. Video analysis sheet 1

Counted by:

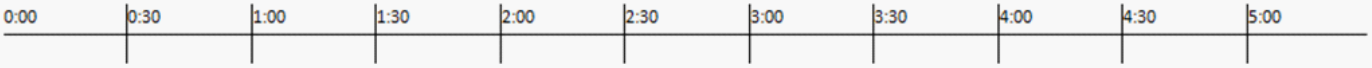
_____

1. Unprompted, spontaneous
2. Not daily routine, required activity
3. Verbal or nonverbal communication or attempt with peer
4. Not reciprocal response

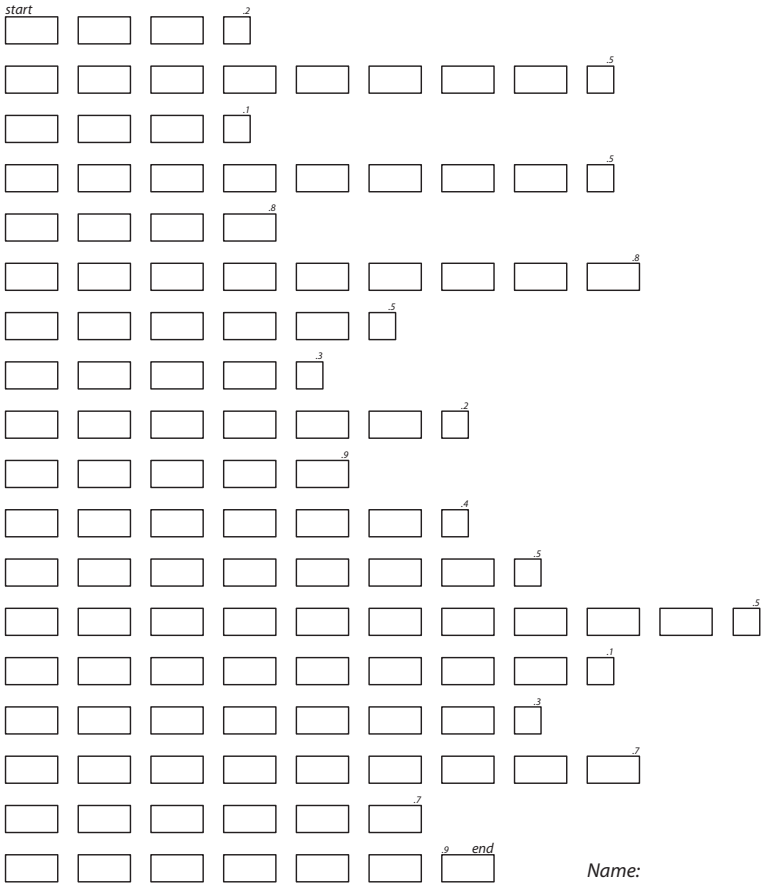
Couple #1



Couple #2



VIII. Video analysis sheet 2



IX. Video analysis results 2 (children combined)

Chi square test 2x2 (Imitate vs. Enhance)

Self initiated social contact per 15 sec. * Robot interaction behaviour Crosstabulation

			Robot interaction behaviour		
			Imitate	Enhance	Total
Self initiated social contact per 15 sec.	No social contact	Count	8	9	17
		Expected Count	9,8	7,3	17,0
	Any social contact	Count	31	20	51
		Expected Count	29,3	21,8	51,0
Total		Count	39	29	68
		Expected Count	39,0	29,0	68,0

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)	Point Probability
Pearson Chi-Square	,982[a]	1	,322	,400	,239	
Continuity Correction[b]	,501	1	,479			
Likelihood Ratio	,974	1	,324	,400	,239	
Fisher's Exact Test				,400	,239	
Linear-by-Linear Association	,968[c]	1	,325	,400	,239	,137
N of Valid Cases	68					

a. 0 cells (,0%) have expected count less than 5. The minimum expected count is 7,25.

b. Computed only for a 2x2 table

c. The standardized statistic is -,984.

Chi square test 2x2 (Imitate vs. Counteract)

Self initiated social contact per 15 sec. * Robot interaction behaviour Crosstabulation

			Robot interaction behaviour		
			Imitate	Counteract	Total
Self initiated social contact per 15 sec.	No social contact	Count	8	11	19
		Expected Count	8,4	10,6	19,0
	Any social contact	Count	31	38	69
		Expected Count	30,6	38,4	69,0
Total		Count	39	49	88
		Expected Count	39,0	49,0	88,0

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)	Point Probability
Pearson Chi-Square	,048[a]	1	,826	1,000	,519	
Continuity Correction[b]	,000	1	1,000			
Likelihood Ratio	,048	1	,826	1,000	,519	
Fisher's Exact Test				1,000	,519	
Linear-by-Linear Association	,048[c]	1	,827	1,000	,519	,201
N of Valid Cases	88					

a. 0 cells (,0%) have expected count less than 5. The minimum expected count is 8,42.

b. Computed only for a 2x2 table

c. The standardized statistic is -,218.

Chi square test 2x2 (Counteract vs. Enhance)

Self initiated social contact per 15 sec. * Robot interaction behaviour Crosstabulation

			Robot interaction behaviour		
			Counteract	Enhance	Total
Self initiated social contact per 15 sec.	No social contact	Count	11	9	20
		Expected Count	12,6	7,4	20,0
	Any social contact	Count	38	20	58
		Expected Count	36,4	21,6	58,0
Total		Count	49	29	78
		Expected Count	49,0	29,0	78,0

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)	Point Probability
Pearson Chi-Square	,704[a]	1	,401	,431	,282	
Continuity Correction[b]	,326	1	,568			
Likelihood Ratio	,694	1	,405	,431	,282	
Fisher's Exact Test				,431	,282	
Linear-by-Linear Association	,695[c]	1	,404	,431	,282	,147
N of Valid Cases	78					

a. 0 cells (,0%) have expected count less than 5. The minimum expected count is 7,44.

b. Computed only for a 2x2 table

c. The standardized statistic is -,834.

Chi square test 4x3

Self initiated social contact per 15 sec. * Robot interaction behaviour Crosstabulation

			Robot interaction behaviour			Total
			Imitate	Counteract	Enhance	
Self initiated social contact per 15 sec.	No social contact	Count	8	11	10	29
		Expected Count	9,5	11,9	7,6	29,0
	Short social contact	Count	13	18	9	40
		Expected Count	13,1	16,5	10,4	40,0
	Medium social contact	Count	7	11	6	24
		Expected Count	7,9	9,9	6,3	24,0
	Long social contact	Count	11	9	6	26
		Expected Count	8,5	10,7	6,8	26,0
Total		Count	39	49	31	119
		Expected Count	39,0	49,0	31,0	119,0

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)	Point Probability
Pearson Chi-Square	2,753[a]	6	,839	,846		
Likelihood Ratio	2,658	6	,850	,859		
Fisher's Exact Test	2,714			,857		
Linear-by-Linear Association	1,188[b]	1	,276	,294	,151	,025
N of Valid Cases	119					

a. 0 cells (,0%) have expected count less than 5. The minimum expected count is 6,25.

b. The standardized statistic is -1,090.

X. Video analysis results 2 (children separate)

Chi square test 2x2 (Imitate vs. Enhance)

Self initiated social contact per 15 sec. * Robot interaction behaviour Crosstabulation

			Robot interaction behaviour		
			Imitate	Enhance	Total
Self initiated social contact per 15 sec.	No social contact	Count	33	23	56
		Expected Count	31,4	24,6	56,0
	Any social contact	Count	45	38	83
		Expected Count	46,6	36,4	83,0
Total		Count	78	61	139
		Expected Count	78,0	61,0	139,0

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)	Point Probability
Pearson Chi-Square	,301[a]	1	,583	,606	,354	
Continuity Correction[b]	,140	1	,708			
Likelihood Ratio	,302	1	,583	,606	,354	
Fisher's Exact Test				,606	,354	
Linear-by-Linear Association	,299[c]	1	,584	,606	,354	,119
N of Valid Cases	139					

a. 0 cells (,0%) have expected count less than 5. The minimum expected count is 24,58.

b. Computed only for a 2x2 table

c. The standardized statistic is ,547.

Chi square test 2x2 (Imitate vs. Counteract)

Self initiated social contact per 15 sec. * Robot interaction behaviour Crosstabulation

			Robot interaction behaviour		
			Imitate	Counteract	Total
Self initiated social contact per 15 sec.	No social contact	Count	33	41	74
		Expected Count	32,8	41,2	74,0
	Any social contact	Count	45	57	102
		Expected Count	45,2	56,8	102,0
Total		Count	78	98	176
		Expected Count	78,0	98,0	176,0

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)	Point Probability
Pearson Chi-Square	,004[a]	1	,950	1,000	,536	
Continuity Correction[b]	,000	1	1,000			
Likelihood Ratio	,004	1	,950	1,000	,536	
Fisher's Exact Test				1,000	,536	
Linear-by-Linear Association	,004[c]	1	,950	1,000	,536	,122
N of Valid Cases	176					

a. 0 cells (,0%) have expected count less than 5. The minimum expected count is 32,80.

b. Computed only for a 2x2 table

c. The standardized statistic is ,063.

Self initiated social contact per 15 sec. * Robot interaction behaviour Crosstabulation

			Robot interaction behaviour		
			Counteract	Enhance	Total
Self initiated social contact per 15 sec.	No social contact	Count	41	23	64
		Expected Count	39,4	24,6	64,0
	Any social contact	Count	57	38	95
		Expected Count	58,6	36,4	95,0
Total		Count	98	61	159
		Expected Count	98,0	61,0	159,0

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)	Point Probability
Pearson Chi-Square	,267[a]	1	,605	,622	,364	
Continuity Correction[b]	,123	1	,726			
Likelihood Ratio	,268	1	,605	,622	,364	
Fisher's Exact Test				,622	,364	
Linear-by-Linear Association	,265[c]	1	,607	,622	,364	,116
N of Valid Cases	159					

a. 0 cells (,0%) have expected count less than 5. The minimum expected count is 24,55.

b. Computed only for a 2x2 table

c. The standardized statistic is ,515.

Self initiated social contact per 15 sec. * Robot interaction behaviour Crosstabulation

			Robot interaction behaviour			Total
			Imitate	Counteract	Enhance	
Self initiated social contact per 15 sec.	No social contact	Count	33	41	23	97
		Expected Count	31,9	40,1	25,0	97,0
	Short social contact	Count	21	26	23	70
		Expected Count	23,0	28,9	18,0	70,0
	Medium social contact	Count	18	25	12	55
		Expected Count	18,1	22,7	14,2	55,0
	Long social contact	Count	6	6	3	15
		Expected Count	4,9	6,2	3,9	15,0
Total		Count	78	98	61	237
		Expected Count	78,0	98,0	61,0	237,0

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)	Point Probability
Pearson Chi-Square	3,050[a]	6	,803	,808		
Likelihood Ratio	2,970	6	,813	,821		
Fisher's Exact Test	2,975			,820		
Linear-by-Linear Association	,066[b]	1	,797	,822	,416	,035
N of Valid Cases	237					

a. 2 cells (16,7%) have expected count less than 5. The minimum expected count is 3,86.

b. The standardized statistic is -,258.