4th International Symposium on New Frontiers in Human-Robot Interaction
21–22 April 2015, Canterbury, UK
M. Salem, A. Weiss, P. Baxter, K. Dautenhahn (Eds.)

Performing Facial Expression Synthesis on Robot Faces: A Real-time Software System

Maryam Moosaei[1], **Cory J. Hayes, and Laurel D. Riek**

Abstract. The number of social robots used in the research community is increasing considerably. Despite the large body of literature on synthesizing facial expressions for synthetic faces, there is no general solution that is platform-independent. Subsequently, one cannot readily apply custom software created for a specific robot to other platforms. In this paper, we propose a general, automatic, real-time approach for facial expression synthesis, which will work across a wide range of synthetic faces. We implemented our work in ROS, and evaluated it on both a virtual face and 16-DOF physical robot. Our results suggest that our method can accurately map facial expressions from a performer to both simulated and robotic faces, and, once completed, will be readily implementable on the variety of robotic platforms that HRI researchers use.

1 Introduction

Robotics research is expanding into many different areas, particularly in the realm of human-robot collaboration (HRC). Ideally, we would like robots to be capable partners, able to perform tasks independently and effectively communicate their intentions toward us. A number of researchers have successfully designed robots in this space, including museum-based robots that can provide tours [10], nurse robots that can automatically record a patient's bio-signals and report the results [22], wait staff robots which can take orders and serve food [17], and toy robots which entertain and play games with children [30].

To facilitate HRC, it is vital that robots have the ability to convey their intention during interactions with people. In order for robots to appear more approachable and trustworthy, researchers must create robot behaviors that are easily decipherable by humans. These behaviors will help express a robot's intention, which will facilitate understanding of current robot actions or the prediction of actions a robot will perform in the immediate future. Additionally, allowing a person to understand and predict robot behavior will lead to more efficient interactions [18, 20].

Many HRI researchers have explored the domain of expressing robot intention by synthesizing robot behaviors that are human-like and therefore more readily understandable [29, 13, 21, 5]. For example, Takayama et al. [35] created a virtual PR2 robot and applied classic animation techniques that made character behavior more humanlike and readable. The virtual robot exhibited four types of behaviors: forethought and reaction, engagement, confidence, and timing. These behaviors were achieved solely by modifying the robot's body movement. Results from this study suggest that these changes in body movement can lead to more positive perceptions of the robot, such as it possessing greater intelligence, being more approachable, and being more trustworthy.

While robots like the PR2 are highly dexterous and can express intention through a wide range of body movements, one noticeable limitation is that there are some subtle cues they can not easily express without at least some facial features, such as confusion, frustration, boredom, and attention [16]. Indeed, the human face is a rich spontaneous channel for the communication of social and emotional displays, and serves an important role in human communication. Facial expressions can be used to enhance conversation, show empathy, and acknowledge the actions of others [7, 15]. They can be used to convey not only basic emotions such as happiness and fear, but also complex cognitive states, such as confusion, disorientation, and delirium, all of which are important to detect. Thus, robot behavior that includes at least some rudimentary, human-like facial expressions can enrich the interaction between humans and robots, and add to a robot's ability to convey intention.

HRI researchers have used a range of facially expressive robots in their work, such as the ones shown in Figure 1. These robots offer a great range in their expressivity , facial degrees-of-freedom (DOF), and aesthetic appearance. Because different robots have different hardware, it is challenging to develop transferable software for facial expression synthesis. Currently, one cannot reuse the code used to synthesize expressions on one robot's face on another [6]. Instead, researchers are developing their own software systems which are customized to their specific robot platforms, reinventing the wheel.

Another challenge in the community is many researchers need to hire animators to generate precise, naturalistic facial expressions for their robots. This is very expensive in terms of cost and time, and is rather inflexible for future research. A few researchers use commercial off-the-shelf systems for synthesizing expressions on their robots, but these are typically closed source and expensive as well.

Thus, there is a need in the community for an open-source software system that enables low-cost, naturalistic facial expression synthesis. Regardless of the number of a robot's facial DOFs, from EDDIE [34] to Geminoid F [9], the ability to easily and robustly synthesize facial expressions would be a boon to the research community. Researchers would be able to more easily implement facial expressions on a wide range of robot platforms, and focus more on exploring the nuances of expressive robots and their impact on interactions with humans and less on laborious animation practices or the use of expensive closed-source

[1] The authors are with the Computer Science and Engineering department, University of Notre Dame,{mmoosaei,chayes3,lriek}@nd.edu

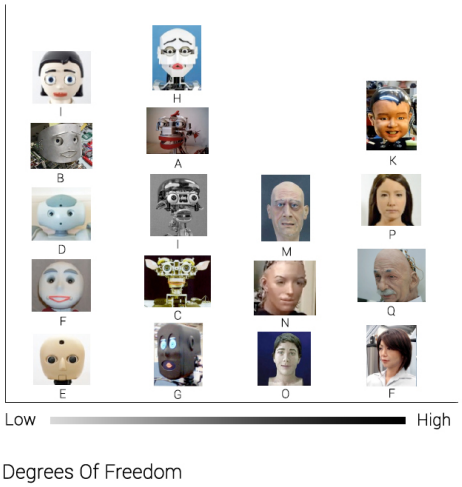

Figure 1. Examples of robots with facial expressivity used in HRI research with varying degrees of freedom. A: EDDIE, B:Sparky, C: Kismet, D: Nao , E: M3-Synchy, F: Bandit, G: BERT2, H: KOBIAN, F: Flobi, K: Diego, M: ROMAN, N: Eva, O: Jules, P: Geminoid F, Q: Albert HUBO , R: Repliee Q2

software.

In this paper, we describe a generalized software framework for facial expression synthesis. To aid the community, we have implemented our framework as a module in the Robot Operating System (ROS), and plan to release it as open source. Our synthesis method is based on performance-driven animation, which directly maps motions from video of a performers face onto a robotic (or virtual) face. However, in addition to enabling live puppeteering or "play-back", our system also provides a basis for more advanced synthesis methods, like shared gaussian process latent variable models [14] or interpolation techniques [23].

We describe our approach and its implementation in Section 2, and its validation in both simulation and on a multi-DOF robot in Section 3. Our results show that our framework is robust to be applied to multiple types of faces, and we discuss these findings for the community in Section 5.

2 Proposed method

Our model is described in detail in the following sections, but briefly our process was as follows: We designed an ROS module with five main nodes to perform performance driven facial expression synthesis for any physical or simulated robotic face.These nodes include:

- *S*, a sensor, capable of sensing the performer's face (e.g., a camera)
- *P*, a point streamer, which extracts some facial points from the sensed face
- *F*, a feature processor, which extracts some features from the facial points coming from the point streamer
- *T*, a translator which translates the extracted features from *F* to either the servo motor commands of physical platforms or the control points of a simulated head
- $C : C_1 \dots C_n$, a control interface which can be either an interface to control the animation of a virtual face or motors on a robot.

These five nodes are the main nodes for our synthesis module. However, if desired, a new node can be added to generate any new functionality.

Figure 2 gives an overview of our proposed method. Assume one has some kind of sensor, (*S*), which senses some information from a person's (*pr*) face. This information might consist of video frames, facial depth, or output of a marker/markerless tracker. *pr* can be either a live or recorded performer. In our general method, we are not concerned about identifying the expressions on the *pr*'s face. We are concerned about how to use the expressions to perform animation/synthesis on the given simulated/physical face. *S* senses *pr* and we aim to map the sensed facial expressions onto the robot's face.

Basically, we use a point streamer *P*, to publish information from a provided face. Any other ROS node can subscribe to the point streamer to synthesize expressions for an simulated/physical face. A feature processor *F*, subscribes to the information published by the point streamer and processes this information. *F* extracts useful features out of all of the facial information published by the point streamer. Then, a translator, *T*, translates extracted features to control points of a physical/simulated face. Finally, a control interface $C : C_1 \dots C_n$ moves the physical/simulated face to a position which matches *pr*'s face.

2.1 ROS implementation

Figure 2 depicts the required parts for our proposed method. The software in our module is responsible for three tasks: (1) Obtaining input, (2) Processing input, (3) Actuating motors/control points accordingly

These responsibilities are distributed over a range of hardware components; in this case, a webcam, an Arduino board, a servo shield, and servo motors.

A local computer performs all processing tasks and collects user input. The data is then passed to the control interface, $C : C_1 \dots C_n$ which can either move actuators on an physical robot or control points on a virtual face. While Figure 2 shows the most basic version of our system architecture, other functionality or services can be added as nodes. Below, we describe each of these nodes in detail as well as the ROS flow of our method.

S, the sensor node, is responsible for collecting and publishing the sensor's information. This node organizes the incoming information from the sensor and publishes its message to the topic `/input` over time. The datatype of the message that this node publishes depends on the sensor. For example, if the sensor is a camera, this node publishes all incoming camera images. Examples of possible sensors include a camera, a Kinect, or a motion capture system. This node can also publish information from pre-recorded data, such as all frames of a pre-recorded video.

P, the point streamer node, subscribes to the topic `/input` and extracts some facial points from the messages it receives. This node extracts some facial points and publishes them to the topic `/points`.

F, the feature processor node, subscribes to the topic `/points`. Node *F* processes all the facial points published by *P*. *F* extracts useful features from these points that can be used to map the facial expressions of a person to the physi-

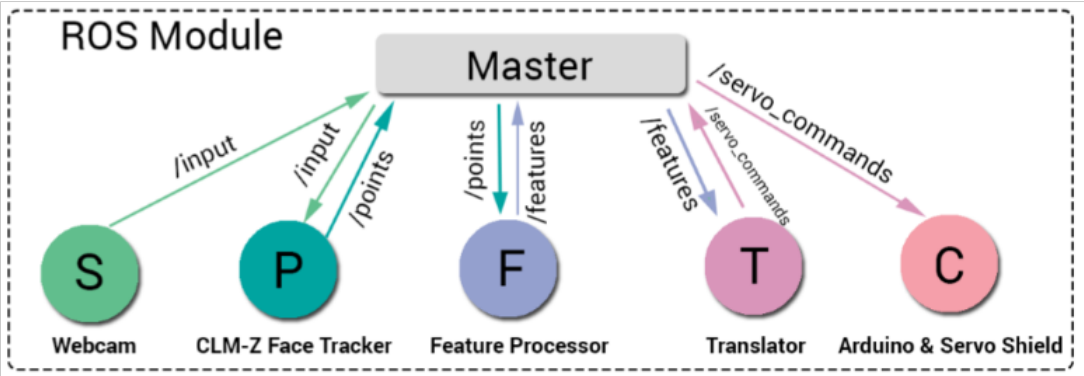

Figure 2. Overview of our proposed method.

cal/simulated face. This node publishes feature vectors to the topic `/features`.

T, the translator node, subscribes to the topic `/features`. and translates the features to DOFs available on the robot's face. Basically, this node processes the message received from the topic `/features` and produces corresponding movements for each of the control points on a robotic or virtual character face. This node publishes its output to the topic `/servo_commands`.

$C : C_1 \dots C_n$: The control interface node subscribes to the topic `/servo_commands` and actuates the motors of a physical robot or control points of a simulated face. We show $C : C_1 \dots C_n$ because a control interface might consist of different parts. For example, in case of a physical robotic head, the control interface might include a microcontroller, a servo shield, etc. We show the combination of all of these pieces as a single node because they cooperate together to actuate the motors. $C : C_1 \dots C_n$ subscribes to the topic `/servo_commands` which contains information about the exact movement for each of the control points of the robotic/simulated face. This node then makes a readable file for the robot containing the movement information and sends it to the robot.

2.2 An example of our method

There are various ways to implement our ROS module. In our implementation in ROS, we used a webcam as *S*. We chose the CLM face tracker as *P*. In *F*, we measured the movement of each of the facial points coming from the point streamer over the time. In *T*, we converted the features to servo commands for the physical robot and slider movements of the simulated head. In *C*, we used an Arduino Uno and a Renbotic Servo Shield Rev2 for sending commands to the physical head. For the simulated faces, *C* generates source files that the Source SDK was capable of processing.

We intended to use this implementation in two different scenarios: a physical robotic face as well as a simulated face. For a physical robot, we used our bespoke robotic head with 16 servo motors. For a simulated face, we used "Alyx", an avatar from video game Half-Life 2 from the Steam Source SDK. We describe each subsystem in detail in the following subsections.

2.2.1 Point streamer P

We employed a Constrained Local Model (CLM)-based face tracker as the point streamer in our example implementation. CLMs are person-independent techniques for facial feature tracking similar to Active Appearance Models (AAMs), with the exception that CLMs do not require manual labeling [12]. In our work, we used an open source implementation of CLM developed by Saragih et al. [1, 33, 11].

We ported the code to run within ROS. In our implementation, `ros_clm` (the point streamer) is an ROS implementation of the CLM algorithm for face detection. The point streamer `ros_clm` publishes one custom message to the topic `/points`. This message to the topic includes 2D coordinates of 68 facial points. This message is used to stream the CLM output data to anyone who subscribes to it.

As shown in the Figure 2, when the *S* node (webcam) receives a new image, it publishes a message containing the image data to the `/input` topic. The master node then takes the message and distributes it to the P node (ros_clm) because it is the only node that subscribes to the `/input` topic.

This initiates a callback in the *P* `ros_clm` node, causing it to begin processing the data which is basically tracking a mesh with 68 facial points over time. The `ros_clm` node sends its own message on the `/points` topic with the 2D coordinates of the 68 facial feature points.

2.2.2 Feature processor F

The *F* node subscribes to the topic `/points`. The *F* node receives these facial points. Using the position of two eye corners, *F* removes the effects of rotation, translation, and scaling. Next, in each frame, *F* measures the distance of each facial point to the tip of the nose as a reference point and saves 68 distances in a vector. The tip of the nose stays static in transition from one facial expression to the other. If the face has any in-plane translation or rotation, the distances of facial points from the tip of the nose will not be affected.

Therefore, any change in the distance of a facial point relative to the tip of the nose point over time would mean a facial expression is occurring. *F* publishes its calculated features to the topic `/features`.

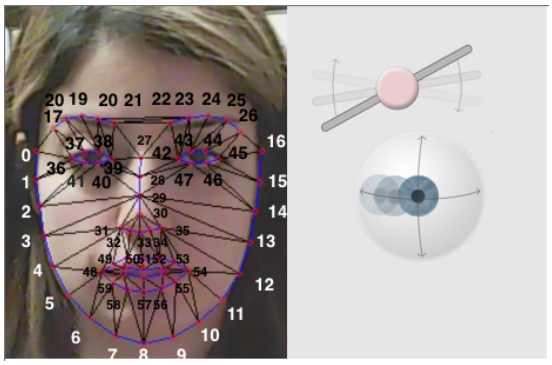

Figure 3. Left: the 68 facial feature points of CLM face tracker, Right: an example robotic eyebrow with one degree of freedom and an example robotic eyeball with two degrees of freedom

2.2.3 *Translator T*

The *T* node subscribes to the topic `/features` and produces appropriate commands for servos of a physical robot or control points of a simulated face. *F* keeps track of any changes in the distances of each facial point to the tip of the nose and publishes them to the `/features` topic. The *T* node has the responsibility of mapping these features to corresponding servo motors and servo ranges of a physical face, or to the control points of a simulated head. *T* performs this task in three steps. The general idea of these steps is similar to the steps Moussa et al. [28] used to map the MPEG-4 Facial Action Parameters of a virtual avatar to a physical robot.

In the first step, for each of the servo motors, we found a group of one or multiple CLM facial points whose movement significantly affected the motor in question. For example, as Figure 3 shows, the CLM tracker tracks five feature points on left eyebrow (22,23,24,25,26). However, the robot face shown in Figure 3 has only one motor for its left eyebrow. Therefore, the corresponding feature group for the robots left eyebrow, would be 22,23,24,25,26.

T converts the movement of each group of the CLM feature points to a command for the corresponding servo motor of a physical robot or control point of a simulated face. We used two examples in this paper, one with a simulated face and one with our bespoke robot. As an example, Table 1 shows the corresponding group of CLM points for each of the 16 servo motors of our bespoke robot

We averaged the movements of all of the points within a given group to compute only one number as the command for each motor/control point. To demonstrate this principle, our bespoke robot has a single motor for the right eyebrow. However, as Figure 3 shows, the CLM face tracker tracks five feature points on right eyebrow. If a performer raises their right eyebrow, the distance of these five points to the tip of the nose increases. We average the movements of these five points and use that value to determine the servo command for the the robot's right eyebrow.

Servo motors have a different range of values than that of feature points. Therefore, in the second step, we created a conversion between these values. The servos in our robot accept values between 1000 and 2000.

To find the minimum and maximum movement of each group of points associated with each servo, we asked a performer to make a wide range of extreme facial movements while seated in front of a webcam connected to a computer running CLM. For example, we asked the performer to raise their eyebrows to their extremities, or open their mouth to its maximum. Then, we manually modified the robot's face to match the extreme expressions on the subject's face and recorded the value of each motor. This way, we found the minimum and maximum movement for each group of facial feature points as well as for each servo motor.

In the last step, we mapped the minimum, maximum, and default values of the CLM facial points and the servo motors. Some servo motors had a reversed orientation with the facial points. For those servos, we flipped the minimum and maximum. In order to find values for a neutral face, we measured the distance of feature points to the tip of the nose while the subject had a neutral face. We also manually adjusted the robot's face to look neutral and recorded servo values.

Using the recorded maximum and minimum values, we applied linear mapping and interpolation (c.f., Moussa et al.) to find the criteria of mapping facial distances to servo values [28]. These criteria are used to translate facial points in each unseen incoming frame to the robot's servo values. The *T* node publishes a set of servo values to the topic `/servo_commands`.

2.2.4 *Control interface* $C : C_1 \ldots C_n$

The *C* node subscribes to the topic `/servo_commands` and sends the commands to the robot. The servo motors of our robot are controlled by an interface consisting of an Arduino UNO connected to a Renbotic Servo Shield Rev2. ROS has an interface that communicates with Arduino through the rosserial stack [2]. By using rosserial_arduino, a subpackage of rosserial, one can add libraries to the Arduino source code to integrate Arduino-based hardware in ROS. This allows communication and data exchange between the Arduino and ROS.

Our system architecture uses rosserial to publish messages containing servo motor commands to the Arduino in order to move the robot's motors. The control interface receives the desired positions for the servo motors at 24 frames-per-second (fps). For sending commands to the simulated face, C generates source files that the simulated face is capable of processing.

Table 1. The facial parts on the robot, and corresponding servo motors and CLM tracker points.

Facial Part	Servo Motor #	CLM Points
Right eyebrow	1	17,18,19,20
Left eyebrow	2	23,24,25,26
Middle eyebrow	3	21,22
Right eye	4 (x direction), 5 (y direction)	37,38,40,41
Left eye (x and y direction)	6 (x direction), 7 (y direction)	43,44,46,47
Right inner cheek	8	49,50
Left inner cheek	9	51,52
Right outer cheek	10	49,50,51
Left outer cheek	11	51,52,53
Jaw	12	56,57,58
Right lip corner	13	48
Left lip corner	14	54
Right lower lip	15	57,58
Left lower lip	16	55,56

3 Validation

To ensure our system is robust, we performed two evaluations. First, we validated our method using a simulated face (we used "Alyx", an avatar in the Steam Source SDK [3]). Then, we tested our system on a bespoke robot with 16 DOFs in its face.

3.1 Simulation-based evaluation

We conducted a perceptual experiment in simulation to validate our synthesis module This is a common method for evaluating synthesized facial expressions [9, 25]. Typically, participants observe synthesized expressions and then either answer questions about their quality or generate labels for them. By analyzing collected answers, researchers evaluate different aspects of the expressions of their robot or virtual avatar.

3.1.1 Method

In our perceptual study, we extracted three source videos of pain, anger, and disgust (total of nine videos) from the UNBC-McMaster Pain Archive [24] and MMI database [31], and mapped them to a virtual face. The UNBC-McMaster Pain Archive is a naturalistic database of 200 videos from 25 participants suffering from shoulder pain. The MMI database [31] is a database of images/videos of posed expressions from 19 participants who were instructed by a facial animation expert to express six basic emotions (surprise, fear, happiness, sadness, anger, and disgust). We selected pain, anger, and disgust as these three expressions are commonly conflated, and were replicating the approach taken by Riva et al. [32].

Using our synthesis module, we mapped these nine facial expressions to three virtual characters from the video game Half-Life 2 from Steam Source SDK. We used three different virtual avatars, and overall we created 27 stimuli videos 3 *(Expression: pain, anger, or disgust)* $\times$ 3 *(Gender: androgynous, male, and female)*. Figure 4 shows example frames of the created stimuli videos.

In order to validate people's ability to identify expressions synthesized using our performance-driven synthesis module, we conducted an online study with 50 participants on Amazon MTurk. Participant's ages ranged from 20-57 (mean age = 38.6 years). They were of mixed heritage, and had all lived in the United States for at least 17 years. Participants watched the stimuli videos in randomized order and were asked to label the avatar's expression in each of the 27 videos.

3.1.2 Results and discussion

We found that people were able to identify expressions when expressed by a simulated face using our performance-driven synthesis module (overall accuracy: 67.33%, 64.89%, and 29.56%[2] for pain, anger and disgust respectively) [19, 26]. Riva et al. [32] manually synthesized painful facial expressions on a virtual avatar with the help of facial animation experts, and found 60.4% as the overall pain labeling accuracy rate [32]. Although we did not set out to conduct a specific test to compare our findings to those of manual animation of the same expressions (c.f. Riva et al. [32]), we found our synthesis method achieved arithmetically higher labeling accuracies for pain. These results are encouraging, and suggest that our synthesis module is effective in conveying naturalistic expressions. The next evaluation is to see how well it does on a robot.

[2] Low disgust accuracies are not surprising; it is known to be a poorly distinguishable in the literature [8].

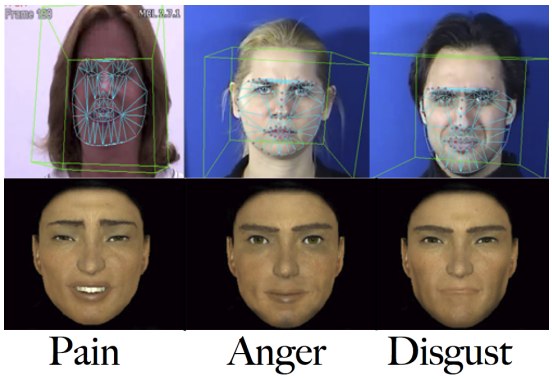

Figure 4. Sample frames from the stimuli videos and their corresponding source videos, with CLM meshes.

3.2 Physical robot evaluation

To test our synthesis method with a physical robot, we used a 16-facial-DOF bespoke robot. Evaluating facial expressions on a physical robot is more challenging than on a simulated face because their physicality changes the physical generation of synthesis. Moving motors in real-time on a robot is far more complex a task due to the number of a robot's motors, their speed, and their range of motion.

We needed to understand if our robot's motors were moving in real time to their intended positions. Since the skin of our robot is still under development, we did not run a complete perceptual study similar to the one we ran in simulation. However, as we were testing how the control points on the robot's head moved in a side-by-side comparison to a person's face, we do not believe this was especially problematic for this evaluation.

3.2.1 Method

We ran a basic perceptual study with 12 participants to test both the real-time nature of the system, and the similarity between expressions of a performer and the robot. We recorded videos of a human performer and a robot mimicking the performer's face. The human performer could not see the robot. However, facial expressions made by the performer were transferred to the robot in real time.

The performer sat in front of a webcam connected to a computer. During the study, the performer was instructed to perform 10 face-section expressions, two times each (yielding a total of 20 videos). The computer instructed the performer to express each of the face-section expressions step by step. Face-section expressions were: *neutral*, *raise eyebrows*, *frown*, *look right*, *look left*, *look up*, *look down*, *raise cheeks*, *open mouth*, *smile*.

We recorded videos of both the performer and the robot mimicking the performer's face. Each video was between 3-5 seconds in length. We ran a basic perceptual study by using side-by-side

Table 2. Full results for each of the 10 face-section expressions.

Face-section expression	Average similarity score	s.d	Average synchrony score	s.d
Neutral	4.12	1.07	4.16	1
Raise eyebrows	4.33	0.86	4.25	1.13
Frown	4	1.02	4.08	1.24
Look right	4.54	0.5	4.66	0.74
Look left	4.5	0.77	4.37	1.08
Look up	2.83	1.29	3.7	1.31
Look down	3.54	1.41	4.45	0.77
Raise cheeks	3.79	1.4	4.25	0.85
Open mouth	4.12	1.16	4.62	0.56
Smile	2.79	1.14	4.41	0.91
Overall	3.85	1.28	4.3	1.01

comparison or "copy synthesis", which we have described in our previous work [27]. In a side-by-side comparison, one shows synthesized expressions on a simulated/physical face side-by-side with the performer's face to participants, and asks them to answer some questions [4, 36].

We showed side-by-side face-section videos of the performer and the robot to participants. Participants viewed the videos in a randomized order. We asked participants to rate the similarity to and synchrony with the performer's expressions and the robot expressions through use of a 5-point Discrete Visual Analogue Scale (DVAS). A five on the scale corresponded to " similar/synchronous" and a one to "not similar/synchronous" .

3.2.2 Results and discussion

Participants were all American and students at our university. Their ages ranged from 20-28 years old (mean age = 22 years). Eight female and four male students participated.

The overall average score for similarity between the robot and the performer expressions was 3.85 (s.d. = 1.28). The overall average score for synchrony between the robot and performer expressions was 4.30 (s.d. = 1.01).

Table 2 reports the full results for each of the 10 face-section expressions. The relatively high overall scores of similarity and synchrony between the performer and the robot expressions suggest that our method can accurately map facial expressions of a performer onto a robot in real-time. However, as this figure shows, we had a low average similarity score for *lookup* and *smile*.

One reason might be that the CLM tracker that we used in our experiment does not accurately track vertical movements of the eyes. Therefore, we could not accurately map the performer's vertical eye movements to the robot. Also, since our robot still does not have skin, its lips do not look very realistic. This might be a reason why participants did not find the robot's lip movements to be similar to the performer's lips movements.

4 General discussion

In this paper, we described a generalized solution for facial expression synthesis on robots, its implementation in ROS using performance-driven synthesis, and its successful evaluation with a perceptual study. Our method can be used both to map facial expressions from live performers to robots and virtual characters, as well as serve as a basis for more advanced animation techniques.

Our work is robust, not limited by or requiring a specific number of degrees of freedom. Using ROS as an abstraction of the code, other researchers may later upgrade the software and increase functionality by adding new nodes to our ROS module.

Our work is also a benefit to the robotics, HRI, and affective agents communities, as it does not require a FACS-trained expert or animator to synthesize facial expressions. This will reduce researchers' costs and save them significant amounts of time. We plan to release our ROS module to these communities within the next few months.

One limitation of our work was that we could not conduct a complete evaluation of our work on a physical robot, since its skin is still under development. Once the robot's skin is completed, we will run a full perceptual test. A second limitation was that the eye-tracking capabilities in CLM are poor, which may have caused the low similarity scores between the robot and performer. In the future as eye tracking technology advances (such as with novel, wearable cameras), we look forward to conducting our evaluation again.

Robots that can convey intentionality through facial expressions are desirable in HRI since these displays can lead to increased trust and more efficient interactions with users. Researchers have explored this domain of research, though in a somewhat fragmented way due to variations in robot platforms that require custom synthesis software. In this paper, we introduced a real-time platform-independent framework for synthesizing facial expressions on both virtual and physical faces. The best of our knowledge, this is the first attempt to develop an open-source generalized performance-driven facial expression synthesis system. We look forward to continuing work in this area.

5 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1253935.

REFERENCES

[1] `github.com/kylemcdonald/FaceTracker`. Accessed: 2014-09-26.

[2] `http://wiki.ros.org/rosserial`. Accessed: 2014-09-26.

[3] Valve Software: Source SDK. `source.valvesoftware.com/sourcesdk.php`.

[4] B. Abboud and F. Davoine. Bilinear factorisation for facial expression analysis and synthesis. In *Proceedings of VISP '05*, pages 327–333, 2005.

[5] H. Admoni, A. Dragan, S. S. Srinivasa, and B. Scassellati. Deliberate delays during robot-to-human handovers improve compliance with gaze communication. In *Proceedings of HRI '14*, pages 49–56. ACM, 2014.

[6] A. Araújo, D. Portugal, M. S. Couceiro, and R. P. Rocha. Integrating arduino-based educational mobile robots in ros. In *Autonomous Robot Systems (Robotica) '13*, pages 1–6. IEEE, 2013.

[7] S. Baron-Cohen. Reading the mind in the face: A cross-cultural and developmental study. *Visual Cognition*, 3(1):39–60, 1996.

[8] D. Bazo, R. Vaidyanathan, A. Lentz, and C. Melhuish. Design and testing of a hybrid expressive face for a humanoid robot. In *IROS '10*, 2010.

[9] C. Becker-Asano and H. Ishiguro. Evaluating facial displays of emotion for the android robot geminoid f. In *WACI '11*, pages 1–8. IEEE, 2011.

[10] M. Bennewitz, F. Faber, D. Joho, M. Schreiber, and S. Behnke. Towards a humanoid museum guide robot that interacts with multiple persons. In *Humanoids '05*, pages 418–423. IEEE, 2005.

[11] S. W. Chew, P. Lucey, S. Lucey, J. Saragih, J. F. Cohn, and S. Sridharan. Person-independent facial expression detection using constrained local models. In *FG '11*, pages 915–920. IEEE, 2011.

[12] D. Cristinacce and T. F. Cootes. Feature detection and tracking with constrained local models. In *BMVC '06*, volume 2, page 6, 2006.

[13] P. F. Dominey and F. Warneken. The basis of shared intentions in human and robot cognition. *New Ideas in Psychology*, 29(3):260–274, 2011.

[14] C. H. Ek, P. Jaeckel, N. Campbell, N. D. Lawrence, and C. Melhuish. Shared gaussian process latent variable models for handling ambiguous facial expressions. In *American Institute of Physics Conference Series*, volume 1107, pages 147–153, 2009.

[15] P. Ekman. About brows: Emotional and conversational signals. In *Human ethology: Claims and limits of a new discipline: contributions to the Colloquium*, pages 169–248, 1979.

[16] R. El Kaliouby and P. Robinson. Generalization of a vision-based computational model of mind-reading. In *Affective computing and intelligent interaction*, pages 582–589. Springer, 2005.

[17] V. Emeli and H. Christensen. Enhancing the robot service experience through social media. In *RO-MAN '11*, pages 288–295. IEEE, 2011.

[18] F. Eyssel, D. Kuchenbrandt, and S. Bobinger. Effects of anticipated human-robot interaction and predictability of robot behavior on perceptions of anthropomorphism. In *Proceedings of HRI '11*, pages 61–68. ACM, 2011.

[19] M. Gonzales, M. Moosaei, and L. Riek. A novel method for synthesizing naturalistic pain on virtual patients. *Simulation in Healthcare*, 2013.

[20] P. A. Hancock, D. R. Billings, K. E. Schaefer, J. Y. Chen, E. J. De Visser, and R. Parasuraman. A meta-analysis of factors affecting trust in human-robot interaction. *Human Factors*, 53(5):517–527, 2011.

[21] H. Knight. Eight lessons learned about non-verbal interactions through robot theater. In *Social Robotics*, pages 42–51. Springer, 2011.

[22] X. Li, B. MacDonald, and C. I. Watson. Expressive facial speech synthesis on a robotic platform. In *IROS '09*, pages 5009–5014. IEEE, 2009.

[23] J. Lieberman and C. Breazeal. Improvements on action parsing and action interpolation for learning through demonstration. In *Humanoids '04*, volume 1, pages 342–365. IEEE, 2004.

[24] P. Lucey, J. F. Cohn, K. M. Prkachin, P. E. Solomon, and I. Matthews. Painful data: The unbc-mcmaster shoulder pain expression archive database. In *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pages 57–64. IEEE, 2011.

[25] D. Mazzei, N. Lazzeri, D. Hanson, and D. De Rossi. Hefes: An hybrid engine for facial expressions synthesis to control human-like androids and avatars. In *BioRob '12*. IEEE, 2012.

[26] M. Moosaei, M. J. Gonzales, and L. D. Riek. Naturalistic pain synthesis for virtual patients. In *Fourteenth International Conference on Intelligent Virtual Agents (IVA 2014)*.

[27] M. Moosaei and L. D. Riek. Evaluating facial expression synthesis on robots. In *HRI Workshop on Applications for Emotional Robots at the 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2013.

[28] M. B. Moussa, Z. Kasap, N. Magnenat-Thalmann, and D. Hanson. Mpeg-4 fap animation applied to humanoid robot head. *Proceeding of Summer School Engage*, 2010.

[29] B. Mutlu, F. Yamaoka, T. Kanda, H. Ishiguro, and N. Hagita. Nonverbal leakage in robots: communication of intentions through seemingly unintentional behavior. In *Proceedings of HRI '09*, pages 69–76. ACM, 2009.

[30] K. Nagasaka, Y. Kuroki, S. Suzuki, Y. Itoh, and J. Yamaguchi. Integrated motion control for walking, jumping and running on a small bipedal entertainment robot. In *ICRA'04*, volume 4, pages 3189–3194. IEEE, 2004.

[31] M. Pantic, M. Valstar, R. Rademaker, and L. Maat. Web-based database for facial expression analysis. In *ICME '05*, 2005.

[32] P. Riva, S. Sacchi, L. Montali, and A. Frigerio. Gender effects in pain detection: Speed and accuracy in decoding female and male pain expressions. *EUR J PAIN*, 2011.

[33] J. M. Saragih, S. Lucey, and J. F. Cohn. Face alignment through subspace constrained mean-shifts. In *ICCV '09*. IEEE, 2009.

[34] S. Sosnowski, A. Bittermann, K. Kuhnlenz, and M. Buss. Design and evaluation of emotion-display eddie. In *IROS '06*, pages 3113–3118. IEEE, 2006.

[35] L. Takayama, D. Dooley, and W. Ju. Expressing thought: improving robot readability with animation principles. In *Proceedings of HRI '11*, pages 69–76. ACM, 2011.

[36] Q. Zhang, Z. Liu, B. Guo, and H. Shum. Geometry-driven photorealistic facial expression synthesis. In *Proceedings of SCA '03*, pages 177–186, 2003.