Inverse Kinematics Solver for Android Faces with Elastic Skin

Emarc Magtanong, Akihiko Yamaguchi, Kentaro Takemura, Jun Takamatsu, and Tsukasa Ogasawara

Abstract The ability of androids to display facial expressions is a key factor towards more natural human-robot interaction. However, controlling the facial expressions of such robots with elastic facial skin is difficult due to the complexity of modeling the skin deformation. We propose a method to solve the inverse kinematics of android faces to control the android's facial expression using target feature points. In our method, we use an artificial neural network to model the forward kinematics and minimizing a weighted squared error function for solving the inverse kinematics. We then implement an inverse kinematics solver and evaluate our method using an actual android.

Key words: android, artificial neural networks, facial expressions, inverse kinematics, human-robot interaction.

1 Introduction

One of the main goals in android research is to design robots that are able to interact with humans in a natural manner. To achieve this objective, efforts have been made to incorporate the ability to display facial expressions on android robots [3].

Commonly, the android's face is controlled by directly adjusting the actuator displacements. However, controlling *feature points* on the android face is more suitable for making facial expressions since it directly adjusts the appearance of the face [6]. A feature point is defined as a specific point on the android's face that moves when displacing the facial actuators. On the other hand, there are only a few methods to control the android's face using feature points because solving the inverse kinematics, which is the relationship between the feature point positions and actuators dis-

E. Magtanong · A. Yamaguchi · K. Takemura · J. Takamatsu · T. Ogasawara
Robotics Laboratory, Nara Institute of Science and Technology, Nara, Japan,
e-mail: {emarc-m,akihiko-y}@is.naist.jp

placements, is difficult due to the elastic facial skin of androids. This paper presents a method for solving the inverse kinematics of such an android face using a machine learning technique.

The problem in solving for the inverse kinematics of android faces with elastic skin is that the facial skin surface is *deformable*. This causes the feature points to move with each other when displacing the actuators. Specifically, there is *coupling* between the feature points. Therefore, it is difficult to formulate an analytic solution to the inverse kinematics since the feature points are not fixed on a rigid link. Furthermore, specifying target feature points for the inverse kinematics is complicated because of the coupling problem.

In this paper, we propose three ideas to solve the inverse kinematics of android faces. Initially, the forward kinematics of the face is modeled using an artificial neural network. The forward kinematics model determines the feature point positions given the actuator displacements. Artificial neural networks are employed because of its capability to learn the complex forward kinematics of android faces. Next, using an iterative minimization technique for an error function, we compute the actuator displacements that satisfies the specified target feature point positions. Also, a weighting method is introduced for computing the difference between the target and the computed feature point positions to address the problem resulting from the feature points being coupled. Lastly, we propose a face segmentation technique to group the facial feature points and the actuators. Segmenting the face reduces the complexity of modeling the forward kinematics and solving the inverse kinematics.

The proposed inverse kinematics solver is validated by conducting several experiments using an actual android. The experimental results demonstrate the ability of the proposed inverse kinematic solver to control the android's facial expressions using target feature points.

There have been several research done to control the facial expressions of android robots. A method used in [4] retargets captured human facial expressions from video to an android by converting 2D feature point positions to actuator displacements using partial least squares regression. Another method in [7] retargets human facial motion capture data to actuator displacements of an android by interpolating weights of blendshape models. Unlike these methods, our proposed inverse kinematics solver will provide a proper solution even for infeasible target feature point positions.

The rest of this paper is organized as follows. Section 2 proposes the method to solve the inverse kinematics of the android's face and the face segmentation technique. Section 3 describes the experimental results. Lastly, section 4 concludes the paper.

2 Inverse Kinematics Solver for an Android Face

To model the forward kinematics, we employ an artificial neural network (ANN). Then, the actuator displacements are computed using an iterative minimization of

the difference between the target and the feature point positions computed from the ANN. Here, a weighting method for the feature points is introduced to handle the coupling problem. Also, we describe a technique to segment the face of the android.

2.1 Forward Kinematics using Neural Network

To model the forward kinematics, we use a multilayer feedforward ANN that is composed of an input, an output, and a hidden layer. Concretely, the ANN is defined as,

$$\mathbf{x} = ANN(\mathbf{u};\Theta), \tag{1}$$

where, Θ denotes the parameters of the neural network and is optimized during training. The vector $\mathbf{u} = (u_1, u_2, ..., u_{N_\mathrm{a}})$ represents the input vector of actuator displacements and $\mathbf{x} = (x_1, y_1, z_1, ..., x_{N_\mathrm{f}}, y_{N_\mathrm{f}}, z_{N_\mathrm{f}})$ defines the output vector of feature point positions, where N_a and N_f denotes the number of actuators and feature points respectively.

For training the ANN, sets of actuator displacements and feature point positions, $\mathscr{D} = \{\mathbf{u}_n, \mathbf{x}_n | n = 1, 2, ...\}$, are used. In training the ANN, a backpropagation algorithm is used to optimize the parameter Θ of the ANN. To avoid overfitting, an early stopping technique is applied during the training [2]. For the experimental section, we implement this ANN using MATLAB's Neural Network Toolbox[1]. The number of neurons in the hidden layer is determined through initial experimentation.

2.2 Solving for the Inverse Kinematics

This section discusses our proposed solution for the inverse kinematics of the android's face based on the forward model learned by the ANN. We aim to address the difficulty of obtaining an analytic solution for the inverse kinematics from the forward kinematics ANN and solve the coupling problem when specifying target feature points. To consider these problems, the inverse kinematics is formulated as the minimization of the weighted squared error of the feature point positions with respect to the actuator displacements. That is,

$$\begin{aligned} \min_{\mathbf{u}} \quad & \sum_{i=1}^{3N_\mathrm{f}} w_i \left[ANN(\mathbf{u})_{[i]} - x^*_{[i]} \right]^2, \\ \text{such that,} \quad & u_{\min j} \leq u_j \leq u_{\max j}. \end{aligned} \tag{2}$$

In Eq. 2, $x^*_{[i]}$ denotes an i-th element of the target feature point position, the subscript $[i]$ denotes the i-th element of the vector, and $(w_1, w_2, ..., w_{3N_\mathrm{f}})$ are the weights for the feature points. The weights handle the coupling problem by emphasizing the error contribution of each feature point.

2.3 Face Segmentation

The inverse kinematics solver described in section 2.1 and 2.2 is applicable to any number of feature points and actuators. We refer to the inverse kinematics solver applied to all feature points and actuators as the *full face* inverse kinematics solver. However, we can improve the precision of the inverse kinematics solver by *segmenting* the feature points and actuators. Segmentation means that the feature points and actuators are grouped to be modeled separately using independent forward kinematics ANNs. Specifically, the set of all feature points $\mathscr{F}_{\mathrm{w}}$ and the set of all actuators $\mathscr{A}_{\mathrm{w}}$ are separated into their subsets: $\{\mathscr{F}_m | m = 1, ..., N_{\mathrm{g}}\}$ and $\{\mathscr{A}_m | m = 1, ..., N_{\mathrm{g}}\}$, where N_{g} denotes the number of segments.

To determine the segmentation, we measure the effect of each actuator to each feature point position. Wherein each facial actuator is independently displaced several times and the feature point positions are recorded. We then compute the *effect index* defined as,

$$\delta_{j,i} = \frac{1}{N_{\mathrm{s}}} \sum_{n=1}^{N_{\mathrm{s}}} \left|\left| \begin{pmatrix} x_{i,n} \\ y_{i,n} \\ z_{i,n} \end{pmatrix} - \begin{pmatrix} x_{i,0} \\ y_{i,0} \\ z_{i,0} \end{pmatrix} \right|\right| \tag{3}$$

where N_{s} denotes the number of samples per each actuator and $(x_{i,n}, y_{i,n}, z_{i,n})$ and $(x_{i,0}, y_{i,0}, z_{i,0})$ denote the current and the neutral (i.e., all facial actuator displacements are at minimum) feature point positions respectively. The effect index indicates the effect of an actuator j to the position of a feature point i.

Using Eq. 3 and a threshold, we can define a subset of feature points $\mathscr{F}_m$ which is affected by an actuator subset $\mathscr{A}_m \in \mathscr{A}_{\mathrm{w}}$. Concretely,

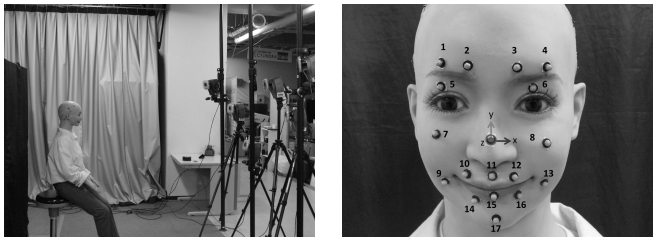
$$\mathscr{F}_m = \{i |^{\forall} i \in \mathscr{F}_{\mathrm{w}}, \delta_{j,i} > threshold, ^{\forall} j \in \mathscr{A}_m\}. \tag{4}$$

To segment properly, the feature point subsets $\mathscr{F}_1, ..., \mathscr{F}_{N_{\mathrm{g}}}$ should not overlap with each other; the same requirement goes for the actuator subsets $\mathscr{A}_1, ..., \mathscr{A}_{N_{\mathrm{g}}}$. Choosing actuator subsets that satisfy these requirements, independent inverse kinematics solvers can be created for each group of feature points and actuators. The resulting inverse kinematics solver is referred to as the *segmented face* inverse kinematics solver.

3 Experiments of Controlling an Android Face

3.1 Capturing Feature Point Positions

The android used for the evaluation of the proposed method is an Actroid-SIT android from Kokoro Co. The android's facial expression is controlled using 11 actuators. For the detailed explanation about the android and the experimental setup please refer to [5].



(a) Motion capture camera setup. (b) Feature point layout.

Fig. 1: Motion capture setup used for capturing feature point positions.

The android's feature point positions are captured using a motion capture system for gathering training data for training the forward kinematics ANN in section 2.1 (see Fig. 1). On the face, 17 feature point markers are placed where significant feature point movement occurs [7]. For the training data of the forward kinematics ANN, several actuator configurations such as independent, combination, and random actuator displacements are recorded. Feature point positions are recorded while keeping the actuator displacements stationary. Additional random actuator configurations are captured for testing the generalization of the forward kinematics ANN.

Using the defined feature points shown in Fig. 1(b), the forward kinematics ANN for the full face inverse kinematics solver has 11 actuators as input and 51 (17 feature points x 3 dimensions) feature point dimensions as output.

3.2 Grouping Feature Points and Actuators

To segment the feature points and actuators of the android, the method mentioned in section 2.3 is applied. The threshold value for $\delta_{j,i}$ is manually selected through experimentation. Fig. 2 shows the feature points that are affected by each actuator. This figure indicates that the feature points and actuators can be segmented into two parts: the upper face and the lips. The upper face segment has 4 actuator displacements as input and 24 dimensional feature point positions as output, while the lips segment has 7 actuator displacements as input and 27 dimensional feature point as output.

3.3 Evaluation of the Forward Kinematics Model

Since the forward kinematics of the android's face is modeled using an ANN, the generalization of the model should be evaluated. The generalization measures the accuracy of the trained ANN when the input data are data not used during training. This tests if the training data is overfitted by the ANN.

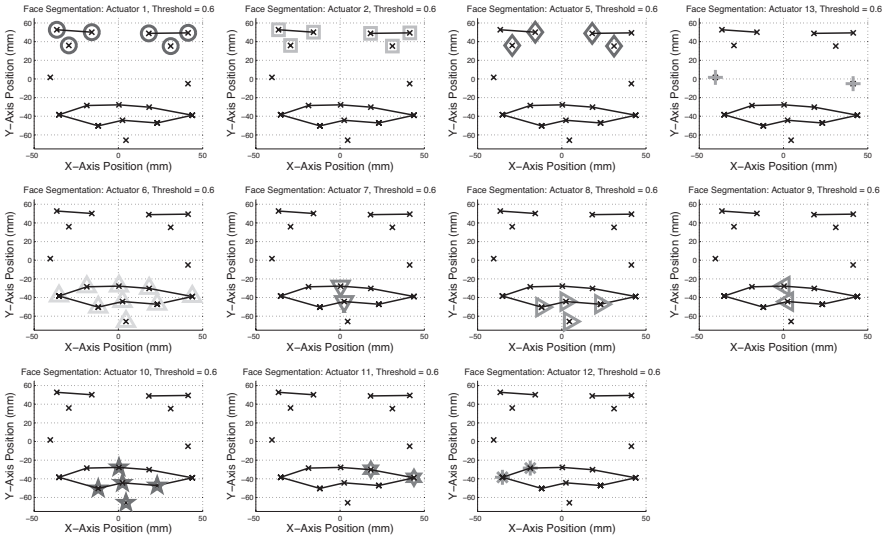


Fig. 2: Face segmentation of the android robot at *threshold* = 0.6.

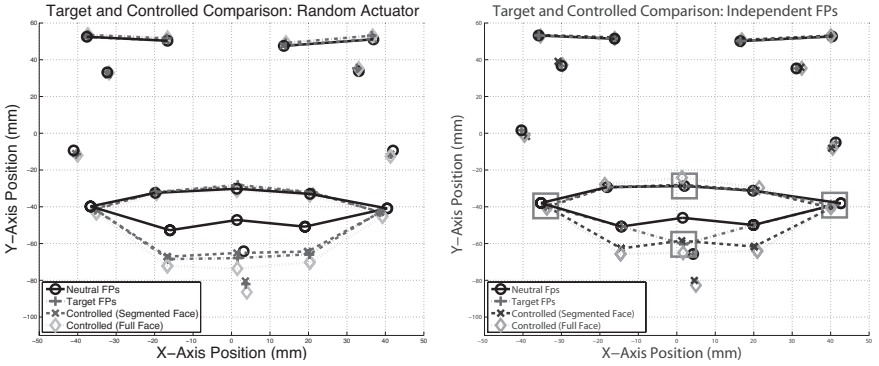
Table 1: Average Norm Error of the Forward Kinematics Artificial Neural Network.

Model Type	Mean (mm)	STD (mm)
Segmented Face	0.54	0.30
Full Face	1.15	0.62

The norm error of the feature points are computed between the output feature point positions of the forward kinematics ANN and actual feature points positions captured by the motion capture system. This is done over 500 samples of random actuator displacements and the results are averaged. In this experiment, the full face forward kinematics ANN and the segmented face forward kinematics ANN are compared. As shown in Table 1 the error is small compared to the average displacement range of the feature points which is 13.32 mm. This suggests that the forward kinematics ANN has good generalization. However, it should be noted that the segmented face has better generalization compared to the full face forward kinematics ANN. The reason for this is that the complexity of modeling the forward kinematics is decreased by reducing the dimensions for each segmented ANN.

3.4 Evaluation of the Inverse Kinematics

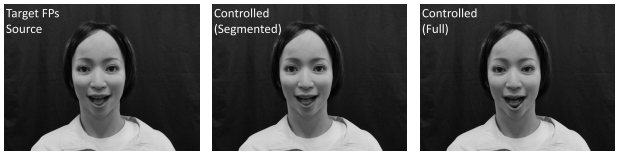
Next, we evaluate the inverse kinematics solvers using two cases; also, the full face inverse kinematics solver and the segmented face inverse kinematics solver are compared. First, the target feature point positions captured from random actuator dis-



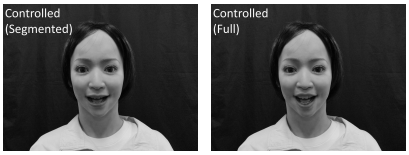
(a) Target feature points captured from random actuator displacements. The Z-axis is omitted.

(b) Independently moved feature points as target feature points. The Z-axis is omitted.

Fig. 3: Plot comparison of target (target FPs) and controlled feature points.



(a) (b) (c)



(d) (e)

Fig. 4: Fig. 4(b)–(c) are controlled using target feature points from random actuator displacements (Fig. 4(a)) and correspond to plots in Fig. 3(a). Fig. 4(d)–(e) are controlled using independently moved target feature points and correspond to plots in Fig. 3(b).

placements is used as input for the inverse kinematics solver. The weights of each feature point are set to 1 since such target feature points are assured to be feasible. The resulting feature points are shown in Fig. 3(a). Inspecting the plot, observe that the target and the controlled feature point positions are close to each other indicating that the inverse kinematics solver can estimate the actuator displacements.

For the second evaluation, independently displaced feature points are set as target feature points for the inverse kinematics solver. This verifies if the weighting method proposed in Eq. 2 solves the coupling problem. The weight of the moved feature points are assigned as 1 and others as 0.01 to emphasize the moved feature points during minimization of the error function in Eq. 2. The results in Fig. 3(b) shows

that the controlled feature points are close to the target feature points. This signifies that the inverse kinematics solver is able to handle the coupling of the feature points.

Furthermore, we can observe from the plots in Fig. 3 that the segmented face inverse kinematics solver achieves better results than the full face inverse kinematics solver. The difference can also be seen by visually inspecting the controlled android face shown in Fig. 4(b)–(e). This is because the full face inverse kinematics solver considers all the feature points and actuator displacements using a single ANN which makes modeling the forward kinematics and the minimization process more complex. This proves that the proposed face segmentation in section 2.3 increased the precision of the inverse kinematics solver.

4 Conclusion

This paper presented a method to solve the inverse kinematics of androids with elastic faces. We addressed the problem of solving the inverse kinematics of such android faces, that is, the complexity of modeling the deformable face and the coupling of the feature points. Our proposed method employed an artificial neural network to model the forward kinematics. Then, the inverse kinematics was solved by using an iterative minimization technique, where a weighted squared error is introduced to handle the coupling of the feature points. This solution to the coupling enables the input of infeasible target feature point positions. Lastly, a face segmentation technique for grouping the feature points and the actuators was proposed to improve the accuracy of the inverse kinematics solver. Experimental results showed that the proposed inverse kinematics solver can control the android's facial expression using target feature points.

References

[1] Beale, M., Demuth, H.: Neural network toolbox. For Use with MATLAB, Users Guide, The MathWorks, Natick (1998)
[2] Bishop, C.: Pattern recognition and machine learning, vol. 4. Springer New York (2006)
[3] Hashimoto, T., Hitramatsu, S., Tsuji, T., Kobayashi, H.: Development of the face robot SAYA for rich facial expressions. In: Int. Joint Conf. on SICE-ICASE, pp. 5423–5428. IEEE (2006)
[4] Jaeckel, P., Campbell, N., Melhuish, C.: Facial behaviour mapping from video footage to a robot head. Robotics and Autonomous Systems **56**(12, 31), 1042–1049 (2009)
[5] Magtanong, E., Yamaguchi, A., Takemura, K., Takamatsu, J., Ogasawara, T.: Inverse kinematics solver for an android face using neural network. In: 29th Annual Conf. of the Robotics Society of Japan, pp. 1Q3–1 (2011)
[6] Tolani, D., Goswami, A., Badler, N.: Real-time inverse kinematics techniques for anthropomorphic limbs. Graphical models **62**(5), 353–388 (2000)
[7] Wilbers, F., Ishi, C., Ishiguro, H.: A blendshape model for mapping facial motions to an android. In: Int. Conf. on Intelligent Robots and Systems, pp. 542–547. IEEE (2007)