Using Visual and Auditory Feedback for Instrument-Playing Humanoids

Daniel Maier Ramin Zohouri Maren Bennewitz

***Abstract*— In this paper, we present techniques that enable a humanoid to autonomously play instruments like the metallophone. The core of our approach is a model-based method to estimate the pose of the instrument and the beaters held by the robot using observations from the onboard camera. For accurate playing, we calibrate the kinematic parameters of the robot and find valid configurations of the arms for beating the individual sound bars of the instrument. To determine these, we rely on the estimated pose of the instrument and the beaters and apply inverse kinematics (IK). Hereby, we use precomputed forward kinematics solutions represented by a reachability tree to accelerate the IK computation and compensate for local minima. The robot automatically validates the computed IK configurations based on visual and auditory feedback using its sensors, and adapts its arm configurations if necessary. Our system parses MIDI-files of whole songs, maps the notes to the corresponding arm configurations for beating, and generates trajectories in joint space to hit the sound bars. As we show in the experiments with a Nao humanoid presented in this paper as well as in the accompanying video, our approach allows for clean and in-time playing of complete songs on a metallophone.**

I. Introduction

In the last decade, there has been a raise of interest in employing robots in the entertainment field, including music. For instance, Chida *et al.* [1] developed the robotic flutist WF-4, Weinberg *et al.* [2] introduced the marimba-playing robot Shimon, and Mizumoto *et al.* [3] demonstrated a theremin-playing HRP-2 robot, while Batula and Kim [4] developed a small-scale humanoid pianist. In addition to entertainment, such robots could also be employed for teaching musical instruments to children or in the treatment of autistic people [5, 6].

In this work, we now present an approach that enables a humanoid robot to autonomously play an instrument like the metallophone. In contrast to the previous robotic musician applications, in our case visual observations are necessary to track the instrument and the beaters held by the robot. To this end, we use a particle filter and developed a novel observation model to enable the robot to accurately track the pose of these objects based on edge observations from its onboard camera. Furthermore, we calibrate the parameters of the robot's kinematic model using a graph-based optimization with respect to marker observations in the camera image. Given the calibration and the pose estimates of the instrument and beaters, we use inverse kinematics (IK) to determine valid configurations of the arms to execute beating motions. To speed up the IK computation and compensate for local minima, we hereby use a so-called reachability tree in which we store precomputed forward kinematics solutions. Since the visual observations and the robot calibration might not be perfect, the robot automatically validates the generated IK configuration by playing the notes and adapting its motion if necessary. Hereby, the robot uses auditory feedback from its onboard microphones and applies pitch detection based on the Fast Fourier Transform (FFT) to identify whether the note was hit correctly. For playing whole songs, our system parses MIDI-files and maps the tones to the corresponding arm configurations that enable the robot to beat the individual sound bars. To reach these arm configurations, the robot generates joint trajectories using Bezier interpolation. Fig. 2 shows an overview of the system.

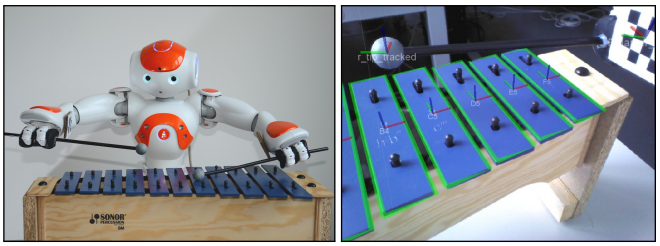

Fig. 1. *Left:* Nao humanoid playing the metallophone. *Right:* View of the robot's onboard camera with estimated pose of the instrument, one beater's head (indicated r_tip_tracked), and calibration marker (coordinate frame inside the checkerboard).

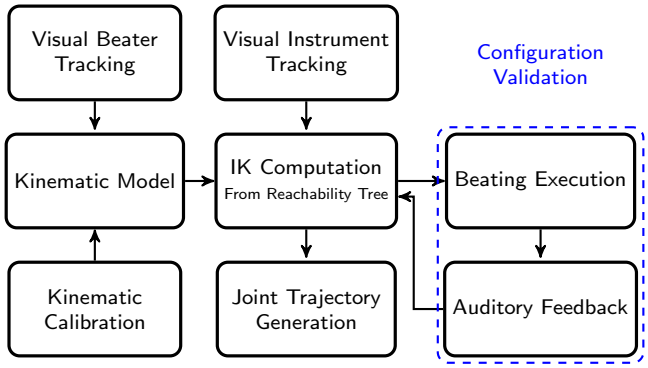

Fig. 2. Overview of the proposed system.

We implemented our approach for a Nao humanoid. Fig. 1 shows the robot playing the instrument and illustrates the tracking result for the metallophone and one beater's head in its camera image. As the experiments presented in this paper as well as the accompanying video demonstrate, the robot is able to play complete songs cleanly and in time using the methods presented in this paper.

All authors are with the Department of Computer Science, University of Freiburg, Germany. M. Bennewitz is also with the Humanoid Robots Lab, Univ. of Bonn. This work has been supported by the German Research Foundation (DFG) within the SFB/TR-8 *Spatial Cognition*, the Research Training Group 1103 *Embedded Microsystems*, and the *BrainLinks-BrainTools* Cluster of Excellence (grant number EXC 1086).

II. Humanoid Robot and Instrument

We implemented and evaluated our system on a Nao V4 humanoid. The robot is 57 cm in height while standing and its arms have a length of approximately 31 cm. Each arm has five degrees of freedom and is equipped with sensors to measure the position of each joint. To determine the pose of the instrument and the beaters' heads the robot analyzes images from the lower monocular camera located in its head, which has a diagonal field of view of 73°.

We use a Sonor SM soprano-metallophone with 11 sound bars of 3 cm in width. The instrument has a size of 49 cm × 20 cm × 22 cm, including the resonating body. The smallest sound bar is playable in an area of 5.5 cm × 3 cm, the largest in an area of 9.5 cm × 3 cm. The instrument is diatonically tuned in C-Major. As beaters, we use Sonor SCH2 with modified grips (see Fig. 1) to allow the Naos's simple grippers to hold them. The beaters are approximately 26 cm in length with a head of 1 cm radius.

III. Calibration of the Kinematic Parameters

Knowledge about the parameters of the robot's kinematic model is essential for tasks requiring high precision such as playing the metallophone. While the kinematic structure is known from the construction plan, errors can occur, e.g., due to imperfect manufacturing. We therefore use a calibration method to accurately estimate these errors.

We use a self-calibration procedure based on checkerboard-markers (see Fig. 1) to calibrate the kinematic model, similar to [7, 8]. More concrete, we minimize the error F between the marker locations observed in the camera image and the projection of the estimated marker locations according to the kinematic model. The projection of a 3D-point $\tilde{\mathbf{P}}$ in homogeneous coordinates is according to the pinhole model

$$\mathbf{p} = \begin{bmatrix} u & v & 1 \end{bmatrix}^\intercal = \mathcal{P}\,\tilde{\mathbf{P}}, \tag{1}$$

$$\text{with} \quad \mathcal{P} = K\,R\,[I_{3\times 3} \; -\mathbf{c}]\,, \tag{2}$$

$$\text{and} \quad K = \begin{bmatrix} f_x & 0 & k_x \\ 0 & f_y & k_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

Here, K are the camera intrinsics (focal length (f_x, f_y)) and principle point (k_x, k_y), and R and $\mathbf{c}$ are the camera's rotation and center, respectively.

Let $\boldsymbol{\theta}$ be the set of calibration parameters. For a set of joint readings $\hat{\mathbf{q}}_i$ and corresponding marker locations $\mathbf{z}_i$ observed in image i, we seek the values $\boldsymbol{\theta}^*$ minimizing the error function F, i.e.,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}), \tag{4}$$

where F is given by

$$F(\boldsymbol{\theta}) = \sum_i e_i(\boldsymbol{\theta}, \mathbf{z}, \hat{\mathbf{q}})^\intercal e_i(\boldsymbol{\theta}, \mathbf{z}, \hat{\mathbf{q}}), \tag{5}$$

$$\text{with} \quad e_i(\boldsymbol{\theta}, \mathbf{z}, \hat{\mathbf{q}}) := \left\| \mathbf{z}_i - P\,\mathsf{forward}^{\mathsf{C}}_{\mathsf{M}}(\boldsymbol{\theta}, \hat{\mathbf{q}}_i) \right\|. \tag{6}$$

Here, $\mathsf{forward}^{\mathsf{C}}_{\mathsf{M}}$ computes the marker location in the camera frame according to the kinematic structure, the calibration parameters $\boldsymbol{\theta}$, and the joint readings $\hat{\mathbf{q}}_i$.

For our Nao robot, the calibration parameters $\boldsymbol{\theta}$ to be estimated include the marker location relative to its gripper, the camera relative to the torso and the joint offsets $\mathbf{q}^{\text{off}}$. Here, we assume a constant error $\mathbf{q}^{\text{off}}$ affecting the joint encoder readings $\hat{\mathbf{q}}_i$ due to imprecise assembly, repair or wear. That means, the true joint position is modeled as

$$\mathbf{q}_i = \hat{\mathbf{q}}_i + \mathbf{q}^{\text{off}}. \tag{7}$$

To solve Eq. 4, we use the $\mathbf{g}^2\mathbf{o}$ framework [9]. More details on our approach to self-calibration can be found in [10].

IV. Model-Based Object Pose Tracking

To play the metallophone, the robot needs to be able to adjust its motions according to the estimated relative poses of the instrument and the heads of the beaters it is holding. Our approach to estimating these poses uses a model-based technique within a particle filter framework[1]. The main idea is, given a hypothesis about an object's pose, one can project the contour of the object's model into the camera image and compare them to actually observed contour. In this way, it is possible to estimate the likelihood of the pose hypothesis. Using independent particle filters, our system maintains multiple hypotheses and propagates them over time to track the instrument as well as the beaters' heads. Our approach hence shares the same idea as e.g., the works [11, 12, 13].

In more detail, the particle filter represents the posterior density function over an object's pose at time t by a set of weighted samples $S_t = \{(X_t^1, w_t^i), \cdots, (X_t^n, w_t^n)\}$. Each sample represents a hypothesis about the pose of the object $X_t^i \in \mathbf{SE}(3)$. The weight w_t^i is proportional to the likelihood of the hypothesis given the history of observations. At each update step, the particles are first propagated according to a random walk as

$$X_t^i = M_t^i\, X_{t-1}^i \tag{8}$$

$$\text{where } M_t^i = \exp \mu_t^i, \quad \text{with } \mu_t^i \sim \mathcal{N}(\mathbf{0}, \Sigma_t). \tag{9}$$

Here, Σ_t is the motion covariance at time t and exp is the matrix exponential mapping $\mathfrak{se}(3) \mapsto \mathbf{SE}(3)$. From the current camera image img_t, the importance weights of the particles are updated according to the observation model as

$$w_t^i = p(\mathsf{img}_t \mid X_t^i), \tag{10}$$

which is described in the next two sections for the metallophone and the beaters' heads. Finally, the filter applies resampling, i.e., a new set of particles is drawn according to the distribution of the importance weights $\{w_t^i\}$. Our algorithm uses the mean of the particles after the resampling step as pose estimate for the given time step.

[1]It is possible to integrate the pose estimation in the kinematic calibration (Sec. III). However, we chose a particle filter framework, as it provides continuous updates of the pose. In this way, it is possible to adjust the motions while playing in case the relative poses change, as we plan to address in future work.

Fig. 3. Example for likelihood computation in instrument tracking. Green lines are the model edges according to a pose hypothesis, with sampled points (green dots). Gray indicates canny edges, blue dots are detected matches to the green dots. The magenta lines indicate the residuals, used in the likelihood computation. Only successful matches smaller than a maximum distance are shown.

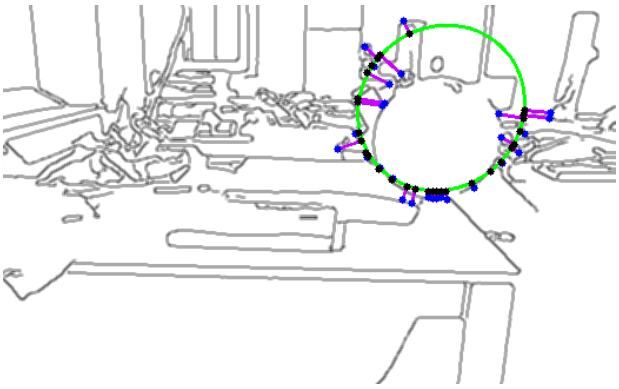

Fig. 4. Example for the likelihood computation for tracking a beater's head. The green ellipse corresponds to the contour of a pose hypothesis, with sampled points (black dots). Gray indicates canny edges, blue dots are detected matches to the green dots. The magenta lines indicate the residuals, used in the likelihood computation. Only successful matches smaller than a maximum distance are shown.

A. Tracking of the Instrument

Fundamental to the localization capabilities of the particle filter is the choice of the observation model $p(\mathsf{img} \mid X)$ in Eq. (10). To estimate the pose of the metallophone, our approach matches the camera image img to the projection of the instrument's edge model according to the pose estimate X. To this end, our system uniformly samples points $\mathbf{P}_m^k$ from the model's edges and projects them into the image according to the standard pinhole model (see Eq. (1)). Here, the camera's rotation and translation $(R, \mathbf{c})$ are with respect to the robot's torso frame. The projected model edge points $\mathbf{p}_m^k$ are then matched to the closest edge points $\mathbf{p}_e^k$, obtained from the Canny edge detector [14]. To find the correspondences, our method follows the model edge normal at $\mathbf{p}_m^k$ until the line intersects with a Canny edge point. If the difference in orientation between the model edge and the corresponding image edge at $\mathbf{p}_e^k$ falls below a threshold, we consider the points matching and store the residual $r^k = \left\|\mathbf{p}_m^k - \mathbf{p}_e^k\right\|$. Otherwise, we follow the line until the next Canny edge with matching orientation. If no match can be found within a maximum distance, this maximum distance is assigned to r^k. Our algorithm approximates the image edge orientation by convolving the image with a horizontal and a vertical Sobel operator and computing the $\mathrm{atan2}$ of the two resulting gradients[2].

With the residual vector $\mathbf{r} = \left(r^1 \cdots, r^{N_p}\right)^\intercal$, where N_p is the total number of sampled points, our approach then computes the observation likelihood from the the arithmetic mean $\bar{\mathbf{r}}$ of $\mathbf{r}$. We assume the observation likelihood to be distributed according to an exponential distribution over $\bar{\mathbf{r}}$ where the distribution parameter λ was determined experimentally:

$$p(\mathsf{img} \mid X) = \lambda \; e^{-\lambda \, \bar{\mathbf{r}}} \tag{11}$$

Fig. 3 shows an example for the likelihood computation. The green lines are the projected model edges, gray are the canny edges, green dots are the $\mathbf{p}_m^k$, blue dots the $\mathbf{p}_e^k$, and magenta lines the residuals r^k. Only edge points that could be matched are shown in the image.

B. Tracking of the Beaters' Heads

To estimate the pose of the beaters attached to the robot's grippers, we use the same particle filter framework[3]. The observation model we employ is based on a mathematical model of the spherical head of the beaters, which we project into the image and sample points from to match to the Canny edge features. A similar mathematical model was previously used to calibrate the camera intrinsics [15].

Specifically, we describe a beater's head as quadric $S \in \mathbb{R}^{4\times4}$ in homogeneous coordinates, with

$$S = \begin{bmatrix} I_{3\times3} & -\mathbf{x} \\ -\mathbf{x}^\intercal & \gamma \end{bmatrix} \tag{12}$$

$$\text{where } \; \gamma = \mathbf{x}^\intercal \, \mathbf{x} - r^2. \tag{13}$$

Here, $\mathbf{x}$ is the estimated translation from the gripper to the center of the head, and r its radius. To evaluate the likelihood of $\mathbf{x}$, we consider the matching of the camera image img, with the projection of S into img. The projection of S is a conic $C \in \mathbb{R}^{3\times3}$ [16], with

$$C = \left(\mathcal{P} \, S^{-1} \, \mathcal{P}^\intercal\right)^{-1}, \tag{14}$$

where $\mathcal{P}$ is the projection matrix from Eq. (1). Here, $(R, \mathbf{c})$ are the rotation and translation of the camera relative to the gripper. The conic C describes an ellipse that is the outline of the projected sphere, i.e., $\tilde{\mathbf{p}} \, C \, \tilde{\mathbf{p}}^\intercal = 0$ for all $\tilde{\mathbf{p}}$ in homogeneous coordinates on the ellipse. In inhomogeneous coordinates, the ellipse can be described by

$$0 = \mathbf{p}^\intercal \, A \, \mathbf{p} + \mathbf{b}^\intercal \mathbf{p} + d \tag{15}$$

$$\text{with } C = \begin{bmatrix} A & \mathbf{b} \\ \mathbf{b}^\intercal & d \end{bmatrix}. \tag{16}$$

[2]One underlying assumption for finding the correspondences is that the poses represented by the particles are not too far away from the true pose of the instrument. We currently achieve that by a rough manual initialization.

[3]Again, it would also be possible to integrate the pose estimation into the kinematic calibration described in Sec. III, however, we observed suboptimal estimation results from such an approach. This is likely due to non-linearities in the encoder readings near the joint limits, unmodeled effects such as joint elasticity, and local minima in the optimization. The particle filter approach automatically handles such problems by updating the pose estimate between gripper and beater.

From Eq. (15), we compute the ellipse's center $\left[h_x, h_y\right]^\intercal$, major and minor axes (l_1, l_2) and rotation ψ, which are needed to sample points from the ellipse. To do so, we bring the ellipse in center-oriented from and perform an eigendecomposition to obtain the parameters. Thus, with definition

$$\mathbf{o} := -A^{-1}\frac{1}{2}\,\mathbf{b} \tag{17}$$

$$\text{and } B := A\left(\mathbf{b}^\intercal A^{-1}\mathbf{b} - d\right)^{-1} \tag{18}$$

and can rewrite Eq. (15) as

$$(\mathbf{p}-\mathbf{o})^\intercal\, B\,(\mathbf{p}-\mathbf{o}) = 1\,. \tag{19}$$

By the eigendecomposition

$$B = R\,D\,R^\intercal, \quad \text{with } D = \operatorname{diag}(\nu_1, \nu_2) \tag{20}$$

we finally obtain the ellipse parameters as: $\left[h_x, h_y\right]^\intercal = \mathbf{o}$, $l_1 = \sqrt{\frac{1}{\nu_1}}$, $l_2 = \sqrt{\frac{1}{\nu_2}}$, and $\psi = \frac{1}{2}\operatorname{atan2}\left(-2A_{12}, A_{22} - A_{11}\right)$. This allows us to write the ellipse equation as

$$x(u) = l_1\,\cos\psi\cos u - l_2\sin\psi\sin u + h_x \tag{21}$$

$$y(u) = l_1\,\sin\psi\cos u + l_2\cos\psi\sin u + h_y, \tag{22}$$

where $(x(u), y(u))^\intercal$ are the contour points of the ellipse, with $u \in [0, 2\pi]$. Our algorithm uniformly samples a fixed number of values for u and follows the normals at $(x(u), y(u))^\intercal$ until they intersect with the Canny edges. From there, we proceed as with the instrument model, using the same exponential distribution as in Eq. (10) to evaluate the likelihood. The normals of the ellipse are computed from its gradients as

$$\mathbf{n}(u) = \begin{bmatrix} y'(u) & -x'(u)\end{bmatrix}^\intercal, \text{with} \tag{23}$$

$$x'(u) = -l_1\,\cos\psi\sin u - l_2\,\sin\psi\,\cos u \tag{24}$$

$$y'(u) = -l_1\,\sin\psi\sin u + l_2\,\cos\psi\,\cos u \tag{25}$$

Fig. 4 shows an example for the likelihood computation of a single particle, while tracking a beater's head. Even though the background contains clutter, the sampled model points are matched to the contour of the beater's head, as a result of considering the edge orientations.

V. Inverse Kinematics and Beating

Based on the estimated pose of the instrument, the beaters' heads, and the calibrated kinematic model, our system computes for each sound bar a suitable beating configuration for the arm kinematic chain. Suitable means that the beater's head can be placed on the surface of the sound bar at a desired angle. From this configuration, the control points of a predefined beating motion are updated. To compute the corresponding joint configuration, the system applies IK based on a resolved-rate approach [17]. To specify the target configurations for the individual sound bars, we provide 4D target poses (position $\mathbf{x_S}$ and roll angle ϕ_S) for the beaters, as well as two general reference joint configuration $\mathbf{q}_r$ (for the left and the right arm), from which the computed arm configurations should deviate as little as possible. The latter is to ensure that the robot does not perform too large motions when switching between different notes and to exploit the redundancy. $\mathbf{q}_r$ is automatically generated once per arm as the beating configuration for the respective central sound bar, which is calibrated first.

In more detail, let $X_B(\mathbf{q}) := \mathsf{forward}^{\mathsf{T}}_{\mathsf{B}}(\boldsymbol{\theta}, \mathbf{q})$ the pose of the beater's head (relative to the torso) according to the forward kinematic model, the estimated transform from the gripper to the beater's head (Sec. IV-B), as well as the calibration parameters $\boldsymbol{\theta}$ (Sec. III), and a joint configuration $\mathbf{q}$. Let $\mathbf{x_B}(\mathbf{q})$ and $\phi_B(\mathbf{q})$ be the translational component and roll angle of $X_B(\mathbf{q})$, respectively. Further, let X_S be the pose of the target sound bar (i.e., its center relative to the robot's torso) according to the tracking result (Sec. IV-A) and the instrument model and let $\mathbf{x_S}$ be the translational component of X_S. With the desired roll angle from the target pose of the beater ϕ_S, we define the error for a configuration $\mathbf{q}$

$$\begin{aligned}\mathbf{e}(\mathbf{q}) &= \mathbf{0}_6\\ \mathbf{e}_{[1:3]}(\mathbf{q}) &= X_S^{-1}\,\mathbf{w}_{[1:3]} \circ X_S(\mathbf{x_S} - \mathbf{x_B}(\mathbf{q}))\\ \mathbf{e}_4(\mathbf{q}) &= \mathbf{w}_4(\phi_S - \phi_B(\mathbf{q})),\end{aligned} \tag{26}$$

where $\circ$ is the Hadamard product, i.e., for $\mathbf{v} = \mathbf{w} \circ \mathbf{x}$, it is $\mathbf{v}_i := \mathbf{w}_i\,\mathbf{x}_i$. Here, $\mathbf{w}$ is a vector of error weights. The idea is to give more weight to deviations in direction of the shorter side of the sound bar, as a misplacement in the other direction is less crucial for hitting the bar. Therefore, the displacement vector $(\mathbf{x_S} - \mathbf{x_B})$ is transformed into the frame of the sound bar, weighted, and transformed back. Further, we consider the deviations of the roll angles.

To reach the target pose, we iteratively compute joint velocities $\dot{\mathbf{q}}_t$ for a discrete time interval Δ_t, and update the joint positions $\mathbf{q}_t$ until $\|\mathbf{e}(\mathbf{q})\|$ falls below a threshold. Let

$$\dot{\mathbf{q}}_t = J_t^{\#}\,\mathbf{e}(\mathbf{q}_t) + (I - J_t^{\#}\,J_t)(\beta_l \nabla f_t + \beta_d \nabla g_t) \tag{27}$$

$$\mathbf{q}_t = \mathbf{q}_{t-1} + \Delta_t\,\dot{\mathbf{q}}_t. \tag{28}$$

Here, J_t is the manipulator Jacobian of the beater for configuration $\mathbf{q}_t$ with its pseudo-inverse $J_t^{\#}$, β_l, β_d are weights for the secondary tasks, $(I - J^{\#}\,J)$ is the null-space projector, ∇f_t the joint limit gradient for $\mathbf{q}_t$, and ∇g_t the gradient of the function $g(\mathbf{q}_t)$ that punishes deviations from the reference joint configuration $\mathbf{q}_r$, where

$$g(\mathbf{q}) := (\mathbf{q} - \mathbf{q}_r)^\intercal\,(\mathbf{q} - \mathbf{q}_r). \tag{29}$$

Such resolved-rate IK solvers suffer from problems such as singularities and local minima. Therefore, providing a good seed configuration is essential. To this end, we developed an approach, the so-called reachability tree (see Fig. 5), to precompute a set of joint configurations (similar to [18]), along with the corresponding beater pose. These configurations are stored for efficient lookup in a kd-tree. In particular, we use the kd-tree to lookup the configurations $\{\mathbf{q}_n\}$ closest to the target pose for each IK query. The closest configuration might not be the best for the IK computation, as the gripper location or robot calibration potentially changed since the tree creation. We therefore pick the configuration $\mathbf{q} \in \{\mathbf{q}_n\}$

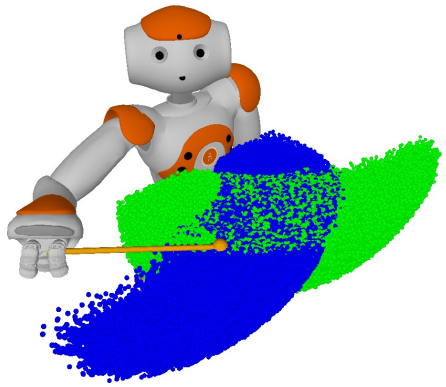

Fig. 5. Illustration of the reachability tree. The blue dots indicate locations that the beater's head can reach with the beater in the right hand. The green dots indicate the same for the beater in its left hand. The configurations are sampled from forward kinematics and used in the IK computation. The samples are restricted to the robot's typical workspace.

as seed that minimizes $\|\mathbf{e}(\mathbf{q})\|$ with $\mathbf{e}$ as defined in Eq. (26) from the current kinematic state and the pose estimates.

After adapting the joint configuration, our system checks whether the beater's head is at the desired pose. Otherwise, the pose estimates are updated and the IK computation is restarted. When the desired pose is reached, the robot performs a predefined beating motion to hit the sound bar, which is then further analyzed from auditory feedback, as described in the following.

VI. Auditory Feedback

To identify whether the executed beating motion successfully hit the sound bar, we rely on auditory feedback, as visual feedback would required a high speed camera which is still uncommon for robots. To this end, our system analyzes the audio signal of the integrated microphones, sampled at 48 kHz. The system considers a window of circa 2 s after the beating motion is executed. To analyze this signal, we follow a straight-forward approach, based on the Fast Fourier Transform (FFT) [19]. Let $\mathbf{a}$ be a discrete, normalized audio signal of length N_a, with $\mathbf{a}_i \in [-1, 1]$ for $i \in \{1, \cdots, N_a\}$. Our algorithm first computes

$$\mathbf{y} = \frac{1}{N_a}\mathsf{FFT}(\mathbf{a}) \tag{30}$$

$$\text{and} \quad \mathbf{m} = \operatorname{abs}\mathbf{y}, \tag{31}$$

where $\mathbf{m}$ is the magnitude of $\mathbf{y}$. The algorithm then searches for local maxima in $\mathbf{m}$ based on a sliding window. The peaks are ordered according to the magnitude values, beginning with the strongest magnitude, i.e., $\mathbf{m}(i_j) \geq \mathbf{m}(i_{j+1})$. We assume that the frequency $f(i_1)$ with the maximum magnitude $\mathbf{m}(i_1)$ is the note that was supposedly played, where $f(i) = \frac{i}{N_a - 1}\,48\text{kHz}$. Our approach employ four different heuristics to identify whether the note was played correctly.

1) The ratio of $\mathbf{m}(i_1)$ to the median magnitude is larger than a threshold.
2) The ratio $\frac{\mathbf{m}(i_1)}{\mathbf{m}(i_2)}$ is larger than a threshold.
3) The deviation of $f(i_1)$ from the frequency f_r of the closest reference note for the instrument in frequency space is smaller than a threshold. The deviation is specified in cent as $1200 \log \frac{f}{f_r}$.
4) The detected frequency f matches the frequency f_d of the desired note.

Only if all heuristics are satisfied, the note is considered successfully played. If only the fourth heuristic fails, the robot repeats the complete IK computation, as the beater is obviously positioned wrongly, e.g., due to wrong pose estimates. Because the pose estimates are continously updated, repeating the IK computation can compensate for temporary errors in the pose estimate. If any of the other heuristics fail, the audio signal is likely to contain only noise, and the robot first tries to beat harder prior to restarting the complete IK computation. This way, the robot learns the strongness for beating a note, and we achieve a robust self-calibration and positioning for playing the instrument.

VII. Joint Trajectory Generation

Our system parses single-track MIDI files to obtain the sequence of notes to play. It converts the notes into a joint trajectory using the beating configurations obtained from IK as control points. The timestamps for the control points are extracted from the MIDI file as well. The trajectory is then computed using Bezier interpolation in joint space by the manufacturer-provided API and send to the the robot controller for execution. This way, the robot plays in-time with the song.

VIII. Experiments

A. Qualitative Evaluation

First, we evaluate the performance of our system as a whole. The results are also shown in the accompanying video as well as in Fig. 6. Initially, our system performed the kinematic calibration procedure as outlined in Sec. III. Afterwards, we placed the beaters in the the robot's grippers, as autonomous grasping is not in the scope of this work. We then manually initialized the localization system and moved the instrument closer to the robot, such that it can reach all the bars with the beaters. As can be seen in the video, the tracker was following the motion of the instrument closely. The robot started the automatic beating calibration, i.e., it estimated the pose of the heads of the beaters, moved the head to the first sound bar, calibrated the beating motion, and proceeded to the next bar. Each sound bar was hit correctly on the first try. As can be seen in the video, the robot was then immediately able to play complete songs such as Beethoven's *Ode to Joy* and *Jingle Bells* provided as MIDI files. Each note in both songs was hit correctly. In the video, one can hear some dissonances that occur when the previous note still sounds while another note is hit and both notes are dissonant. Such effects can be prevented by advanced players by stopping the previous note at the same time as playing the current one. In future work, we will investigate such techniques in our system.

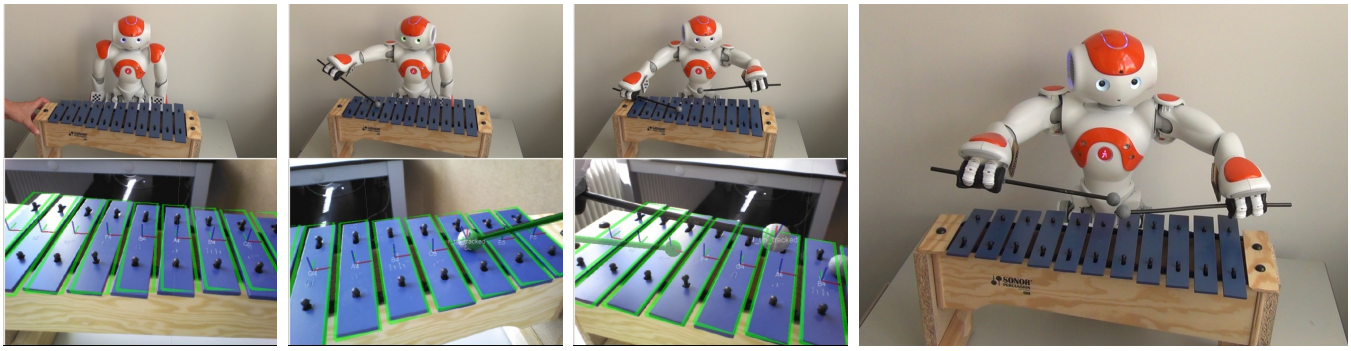

Fig. 6. Screenshots from the beating calibration procedure shown in the accompanying video. The top row shows an external camera video for reference, the lower the robot's onboard perspective along with the tracking results. 1st *left*: The instrument is moved in front of the robot, while it keeps track of the instrument. 2nd *left*: The robot calibrates the beating configuration for the D-sound bar from the visual pose estimation. 3rd *left*: The robot calibrates the beating motion for another sound bar with its left arm. The green stick indiactes the determined IK solution. *Right*: The robot plays a song using the learned motions.

B. Pose Estimation and Calibration

Fig. 1 shows an example for the pose estimation of the instrument and one beater's head. The edge model closely fits the camera image, and the estimated center of the beater's head (indicated by the coordinate frame) aligns with the head in the image as well. Furthermore, one can see the result of the kinematic calibration, as the coordinate frame indicating the center of the checkerboard marker, nicely overlays with the checkerboard in the camera image. Furthermore, in the accompanying video, a sequence showing tracking results for the beater and the instrument can be seen.

The localization performance is generally good, but the markerless localization approach can fail, once it loses track of the instrument. This is because there is a strong symmetry in the appearance of the object. If the pose estimate is shifted by one (or more) sound bar plus a small deviation in depth, the appearance is very similar and hence, the likelihood function will also return similar values. This can be avoided by additional markers on the object, which we chose to avoid. In practice, the localization performance has proven sufficient for our application.

C. Auditory Feedback

To test the auditory feedback, we had a novice human player play the notes of the instrument while the robot ran the audio analysis. We evaluated whether the detected notes were the actually played one. The human played notes randomly, with a pause of circa 5 s in between two notes. We placed the instrument at 3 different locations (left, right, center) relative to the robot. Each of the 11 notes were played in each position. From 33 played notes, the system was able to recognize 30 correctly, for the 3 failures, the notes were determined correctly, but heuristic 2 was not satisfied. Consequently, in the beating calibration context, the robot would unnecessarily have tried to solved the IK again. Further, we tested whether the analyzer is able to identify whether two notes were hit at the same time. In practice, this is useful, as the robot sometimes hit inbetween two notes, due to imprecise motion execution or minor pose estimation errors. To this end, the novice player was asked to hit two neighboring notes at the same time, thereby simulation the inbetween-sound-bars effect (as hitting inbetween on purpose proved to be difficult). In all three positions, all of the 11 notes were correctly identified as unacceptable.

D. Beating Calibration Accuracy

To test the accuracy of the beating motion calibration, we let the robot perform the beating calibration five times, similar to the one shown in the video and described in Sec. VIII-A. For each trial, the instrument's location was shifted slightly within the reachable limits for the robot. For each location, the robot managed to find a configuration for all of the 11 sound bars and beat each note correctly as part of the auditory feedback. Therefore, one can conclude, that the accuracy of the visual tracking is sufficiently good for the task of calibrating the beating motions.

IX. Conclusions

In this paper, we presented a novel approach to enable a humanoid robot to play the metallophone. To realize this task, we developed a technique for self-calibration of the robot kinematic model, a particle filter framework for model-based pose estimation of the instrument and the beaters, and a method for exploiting auditory feedback to automatically adapt the beating motion. As we showed in the experiments in the paper and in the accompanying video, we achieve good accuracy in tracking the relevant object poses and in the kinematic calibration that allows for clean playing of complete songs. Hereby, the robot adapts its motions if necessary based on visual and auditory feedback. In future work, we plan to update the beating motions while the robot is playing, which relies on a constant update of the instrument's pose using our particle filter framework.

Our methods are not restricted to the Nao humanoid but could easily be adopted for other robots as well. The employed techniques such as visual tracking using a model-based approach are also general and can be used in the context of other tasks such as stair climbing or estimating the pose of other tools held by the robot.

REFERENCES

[1] K. Chida, I. Okuma, S. Isoda, Y. Saisu, K. Wakamatsu, K. Nishikawa, J. Solis, H. Takanobu, and A. Takanishi. Development of a new anthropomorphic flutist robot WF-4. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.

[2] G. Weinberg, T. Mallikarjuna, and A. Ramen. Interactive jamming with Shimon: A social robotic musician. In *Proc. of the Int. Conf. on Human Robot Interaction (HRI)*, 2009.

[3] T. Mizumoto, H. Tsujino, T. Takahashi, T. Ogata, and H. Okuno. Thereminist robot: Development of a robot theremin player with feedforward and feedback arm control based on a theremin's pitch model. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[4] A. M. Batula and Y. Kim. Development of a mini-humanoid pianist. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2010.

[5] I. Fujimoto, T. Matsumoto, P. R. S. De Silva, M. Kobayashi, and M. Higashi. Study on an assistive robot for improving imitation skill of children with autism. In *Proc. of the Int. Conf. on Social Robotics (ICSR)*, 2010.

[6] D. Ricks and M. Colton. Trends and considerations in robot-assisted autism therapy. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[7] U. Hubert, J. Stückler, and S. Behnke. Bayesian calibration of the hand-eye kinematics of an anthropomorphic robot. In *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2012.

[8] O. Birbach, B. Bäuml, and U. Frese. Automatic and self-contained calibration of a multi-sensorial humanoids upper body. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[9] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[10] D. Maier, S. Wrobel, and M. Bennewitz. Whole-body self-calibration via graph-optimization and automatic configuration selection. 2015. Submitted.

[11] C. Choi and H. I. Christensen. Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *Int. Journal of Robotics Research (IJRR)*, 31(4), 2012.

[12] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade. GPU-accelerated real-time 3D tracking for humanoid locomotion and stair climbing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[13] D. Gonzalez-Aguirre, M. Vollert, T. Asfour, and R. Dillmann. Robust real-time 6d active visual localization for humanoid robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

[14] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[15] M. Agrawal and L. Davis. Camera calibration using spheres: a semi-definite programming approach. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2003.

[16] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[17] S. R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. `http://euclid.ucsd.edu/~sbuss/ResearchWeb/ikmethods/index.html`, 2004.

[18] N. Vahrenkamp, T. Asfour, and R. Dillmann. Robot placement based on reachability inversion. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

[19] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics Computation*, 1965.