Vision Augmented Robot Feeding

Alexandre Candeias[1](✉), Travers Rhodes[2], Manuel Marques[1], João P. Costeira[1], and Manuela Veloso[2]

[1] ISR - IST Universidade de Lisboa, Lisbon, Portugal
alexandre.candeias@tecnico.ulisboa.pt, {manuel,jpc}@isr.tecnico.ulisboa.pt

[2] Carnegie Mellon University, Pittsburgh, USA
traversr@andrew.cmu.edu, mmv@cs.cmu.edu

Abstract. Researchers have over time developed robotic feeding assistants to help at meals so that people with disabilities can live more autonomous lives. Current commercial feeding assistant robots acquire food without feedback on acquisition success and move to a preprogrammed location to deliver the food. In this work, we evaluate how vision can be used to improve both food acquisition and delivery. We show that using visual feedback on whether food was captured increases food acquisition efficiency. We also show how Discriminative Optimization (DO) can be used in tracking so that the food can be effectively brought all the way to the user's mouth, rather than to a preprogrammed feeding location.

Keywords: Assistive technologies · Manipulation aids Computer vision · Feeding assistance

1 Introduction

Disabilities that can affect control of the arms, including paralysis, Parkinson's disease, and cerebral palsy, may prevent or hinder someone from feeding themselves. In that case, the person may require a human caretaker to assist in the feeding task. To help people with disabilities live more autonomous lives, researchers have over time developed and tested how robotic feeding assistants may help at the difficult task of having a meal [1]. A feeding robot is a way of achieving greater independence at meals and can make the mealtime a more social event. Also, compared to a human caretaker, a robotic feeding assistant can have more time and patience in soliciting user requests for what type of food to acquire and in following the user's desired pace of the meal [2].

A. Candeias and T. Rhodes—Contributed equally.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-11024-6_4) contains supplementary material, which is available to authorized users.

There are several commercial assistive feeding robots, including the Obi [3], Bestic [4], and Meal Buddy [5]. All of these robots allow the user or a caretaker to program the desired feeding location to which the robot will bring the food. However, that feeding location remains constant throughout the meal. Current feeding systems are not equipped with a perception scheme and are not able to perceive the changes in the pose of the user or in the environment. People without sufficient head control to bring their mouth to the programmed feeding location might be better accommodated by a robot that can bring the spoon all the way to the current location of the user's mouth.

To solve this problem we propose in this paper to create a complete robotic feeding assistant system that incorporates a visual perception module to locate the user's face and mouth. Our system setup is shown in Fig. 1.

We propose a real-time system that uses depth images to track the user's mouth in 3D space and a separate vision system to provide useful feedback for how much food was acquired on the robotic spoon. Real experiments show that the visual feedback module significantly improves the feeding system's performance.

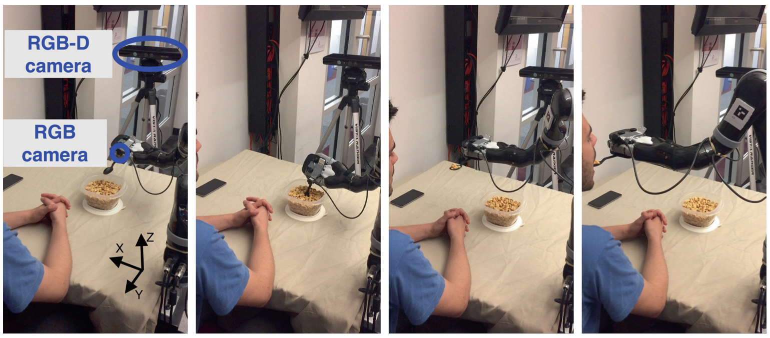


Fig. 1. Our feeding system includes a MICO robot arm, an RGB-D camera, and an RGB camera. The first image also labels the axis orientations of the robot coordinate system. (Color figure online)

2 Related Work

Several approaches have been used previously in detecting mouth location for robotic feeding tasks. In [6], an ARTag was affixed to the user's forehead. In [7], Park et al. extend their system to localize the user's mouth using a RGB-D camera, but do not specify or evaluate the algorithm used. Hawkins et al. [8] also focus on head pose estimation including user's feedback to help the head detection. Visual servoing was used to find the user's mouth in [9], but that system was unable to ascertain the distance to the user's mouth along the principal axis, as it did not use a depth camera. Their system takes roughly 10 s to identify the mouth location [9], whereas our system tracks mouth location in real time.

In [10], the user's face is tracked using three specific face points (one on the forehead, and one on each cheek), and those points are used to infer the mouth location. Like their system, the proposed approach is robust to occlusions of the mouth. However, unlike [10], our tracking algorithm is able to track faces even in profile.[1]

Discriminative Optimization (DO) [11] was applied in the work of Silva et al. [12] to track the user's face for robotic feeding, however they do not provide results about the performance of the tracking or the feeding system.

Research into anomaly detection during feeding tasks was performed in [7], which is used to detect errors during the feeding motion to the user's mouth. However, anomalies related to food acquisition were not analyzed in that work. The anomalies we investigate in our work are related to failures in acquiring a sufficient amount of food. Herlant [13] also uses feedback on the state of the fork in order to detect an anomalously low mass of food acquired. They use an expensive 6DOF force-torque sensor, while we use an inexpensive vision system as our feedback sensor, in the hope that a cheaper system will accessible to a broader range of people.

The work of Ragusa et al. [14] focuses on food/non-food detection on plate images. In that work, they benchmark different deep-learning based approaches. They used pre-trained convolution neural networks (CNNs) to capture features and trained a support vector machine (SVM) and Soft-Max classifier to classify food/non-food images. That work also analyses the impact of fine tuning the CNNs to capture more relevant features to the food/non-food detection. Although those approaches are successful, in this work we are interested in classification of specific spoon images. We also go beyond classification and compute a continuous measure of how much food is present in the image of the spoon.

3 Feeding System

The developed feeding system is composed of two parts: a vision system that is responsible for tracking the user and also for detecting if there is food on the spoon and a control system that is responsible for transforming visual perception into tasks executed by the robot arm. These tasks are food acquisition from the plate and delivering the food to the user's mouth.

In the next subsections, we describe each component in the system diagram presented in Fig. 2.

3.1 Vision System

Our vision system is composed of two parts. One is responsible for tracking the person's face using an RGB-D sensor, and the other is responsible for detecting what is on the spoon using a small RGB camera mounted on the robot's end-effector.

[1] A video of our system working is available at https://www.youtube.com/watch?v=X7McqWk1AK8.

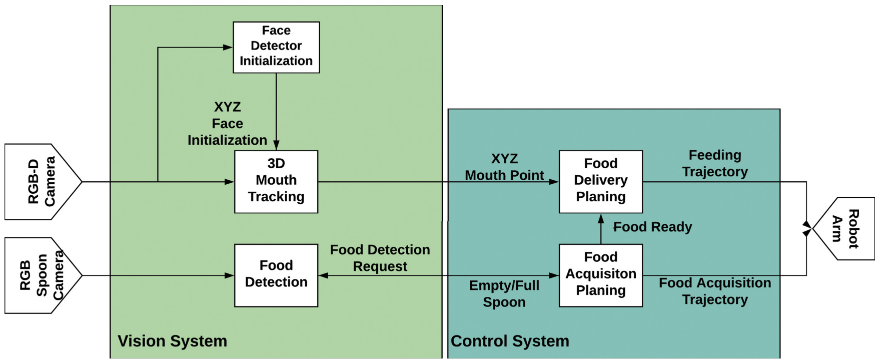


Fig. 2. Feeding system architecture diagram

Tracking a person's face is a challenging task because faces are non-rigid. However, since the face movements are constrained by the head movements, we can, in a first approximation, track the person's head and assume that the mouth is just a point in the rigid body shaped by the head.

Given a front view 3D model of the person's head (P_M) and a 3D point cloud of the scene given by the RGB-D sensor (P_S), we can formulate the problem of tracking the person's head as a 3D point registration problem. This can be formulated as the following optimization problem:

$$\begin{gathered}(R^*, t^*, C^*) = \underset{R,t,C}{\operatorname{argmin}} \sum_{i=1}^{N_{P_S}} \sum_{j=1}^{N_{P_M}} C_{ij} \left\| p_S^i - R \cdot p_M^j - t \right\|^2 \\ \text{subject to } \det(R) = 1, \quad R^T R = I, \quad C_{ij} \in \{0, 1\}\end{gathered} \tag{1}$$

where we want to fit the rotation R^* and translation t^* that align the model point cloud with the scene point cloud. p_S^i and p_M^j are, respectively, points from the point clouds P_S and P_M. C_{ij}^* represents the optimal point correspondences between the two point clouds. C_{ij}^* is equal to 1 if p_S^i and p_M^j are corresponding points and the model point p_M^j is visible on the scene. Otherwise, C_{ij}^* will be 0. Since we do not know the correspondences C^* between the two point clouds, we have to include them in the optimization problem. This leads to a non-convex combinatorial problem.

When the correspondences between the model and scene point clouds are not known, it is a common practice to solve 3D registration problems using ICP [15]. Although ICP is a standard procedure to use, it tends to get trapped in local minima near the initialization provided and to have problems when dealing with a high percentage of outliers.

To overcome these problems we use Discriminative Optimization (DO) [11] to solve the model-based 3D registration tracking problem presented in Eq. 1. In [11] it is shown that DO is better than ICP in terms of computation time and more robust to outliers.

Discriminative Optimization (DO). We want to develop a vision system that is capable of locating the person's mouth in real time. We also want that system to be robust to occlusions of the face, because the robot arm will move between the person's face and the camera.

Discriminative Optimization (DO) [11] is a method that can solve the registration problem of Eq. 1 given a 3D model. DO is a fast method compared to other approaches like ICP and has a wider convergence region. Moreover, since we want to deal with occlusions, DO is a good approach because it can handle a high percentage of outliers. Finally, DO has lighter computational requirements than approaches based on deep learning and can be run directly on the CPU. DO is a learning-based methodology to tackle problems that are formulated as optimization problems but for which the function is unknown. From training data, DO learns a vector field, represented by a series of linear maps, that "emulate" the gradient of a function. During testing, given unseen data, DO follows this "gradient" leading to a stationary point that solves the optimization problem. Specifically in our case, the algorithm is given a 3D model of the rigid body that we want to track and generates a set of synthetic training data by applying random rotations and translations to the 3D model. This augmented training data is used to discover a sequence of linear maps that represent the descending directions. Those maps are used during inference, which is an iterative procedure that repeatedly updates the translation and rotation of the 3D model of the head to better align it with the point cloud of the scene.

Initialization. 3D registration algorithms, like ICP and DO, require an initialization step that places the model of the face close to the face in the scene. We use a face detector as our initial guess for where the face is in the RGB-D image of the scene.

A popular face detector is the Viola Jones face detector [16]. Though this face detector is fast, it is not invariant to different face poses. MTCNN [17] is more robust to different face poses and also provides landmarks of the mouth, nose, and eyes. Though MTCNN is based on deep learning, it can still run on a CPU at a lower speed. Since initialization does not happen frequently, the speed of the initialization step is not an important concern, and we remove the need for a GPU in our system by running initialization on a CPU.

We initialize our face tracker using the average 3D location of the facial landmarks given by MTCNN. The 3D locations of these landmark points can be computed because we are using an RGB-D camera. With these 3D points, we approximate the initial translation $\tilde{t}$ of the face model in the scene as:

$$\tilde{t} = \frac{1}{L}\sum_{i=1}^{L} p_S^i - \frac{1}{N}\sum_{j=1}^{N} p_M^j, \tag{2}$$

where L is the number of landmarks and N is the number of points in the model. For the initial rotation of the model, we assume $R = I$, where I is the 3×3 identity. The MTCNN landmarks are also used to compute the mouth 3D location in the model's point cloud.

Food Detection. To see what is on the spoon, we place a tiny RGB camera on the end-effector of the robot arm. After a simple calibration step of holding a light-colored background behind the spoon, we compute a mask for the image that only contains the spoon. We use this masked image for two purposes, to detect if there is enough food to serve the user, and to detect if the user has eaten the food off the spoon. To detect if there is enough food on the spoon we use a detection algorithm that we can tune to specify the required amount of food. To detect if the user has eaten the food, we use a classifier with two classes: "food" and "no food."

Calibration is performed by acquiring an image with a white sheet of paper under the spoon. Otsu's method [18] is used to select a threshold and segment the image which provides a spoon mask. This method works well because the colored spoon contrasts sharply with the white sheet of paper. In the calibration step, we also record a histogram, H_s, of the H channel of the HSV image of the empty spoon after applying the calibration mask.

The recorded histogram, H_s, is compared with the histograms of subsequent images of the spoon, H_a. To compare the histograms, we used the normalized correlation between the two histograms,

$$d(H_s, H_a) = \frac{\sum_{i=1}^{N}(H_s(i) - \bar{H}_s)(H_a(i) - \bar{H}_a)}{\sqrt{\sum_{i=1}^{N}(H_s(i) - \bar{H}_s)^2 \sum_{i=1}^{N}(H_a(i) - \bar{H}_a)^2}}, \tag{3}$$

where $\bar{H} = \frac{1}{N}\sum_{i=1}^{N} H(i)$ is the histogram mean. If the value of the correlation is below a certain threshold, then the algorithm reports that there is enough food on the spoon.

To tune the correlation threshold, we gathered a dataset containing 387 data points of correlations and the corresponding weight of food present on the spoon. Example images collected are shown in Fig. 3. We use peanuts and fried rice as our test foods. We train a linear regression to discover the relationship between the histogram correlation value and the weight of the food on the spoon. This regression can be used to inform the choice of the threshold for "enough food" on the spoon.

We can detect if there is food on the spoon after the user's bite using a classifier. We tried two different classifiers: a logistic regression and a linear support vector machine (SVM). The input feature to the logistic regression is the value of the histogram correlation, and the input features to the SVM are the

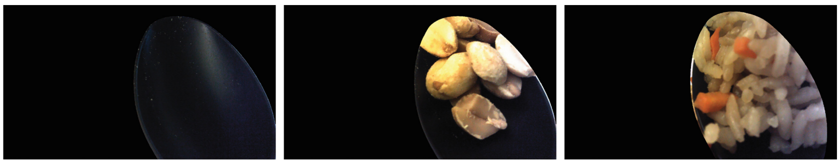


Fig. 3. Example masked images of the plastic spoon, showing the spoon empty, with nuts, and with rice.

bins of the H color channel histogram of the spoon image. Both classifiers were trained using a dataset containing 119 empty spoon images and 118 non-empty spoon images.

3.2 Control System

Gradient-Based Planning. The goal of our planning algorithm is to move the tip of the spoon from its current location to the user's mouth. Because the mouth of the user may be moving, we need our planning algorithm to be able to quickly re-compute its plan based on updated mouth-target positions. For this reason, we choose a planning algorithm that iteratively moves the tip of the spoon one step in a straight line toward the current location of the mouth. We also want to ensure that the spoon does not dump its contents during transit, so we control the orientation of the spoon during transit.

A flexible approach to allow quick re-planning and to constrain orientation is a gradient-based approach. At each timestep, we compute an intermediate Cartesian end-effector target that is a certain distance, parameterized by a parameter "translationStepSize," from the current location. The intermediate Cartesian target is the mouth location itself if the mouth location is within "translationStepSize."

We similarly compute an intermediate orientation target by computing the rotation necessary to convert from the current orientation of the end-effector to the target orientation. We define the target orientation to along the y-axis direction shown in Fig. 1. We set our intermediate orientation target to be the orientation that is at most a certain angle rotation, parameterized by a parameter "rotationStepSize," away from the current orientation. During transit from the plate to the user's mouth, the orientation is never more than a very small angle away from the desired final feeding orientation, so the target orientation is always the desired final orientation.

We use OpenRave [19] to compute the Cartesian Jacobian, which is a $3 \times \mathrm{DOF}$ matrix showing how changes in each joint angle locally moves the end-effector in Cartesian space. We also use OpenRave to compute the angular velocity Jacobian, which is a $3 \times \mathrm{DOF}$ matrix showing how changes in each joint angle locally rotates the end-effector, where rotations are given in angle-axis representation.

We write our desired translation as a length three vector. We concatenate that vector with the angle-axis representation of our desired rotation to give us a length 6 vector representing our desired change in end-effector pose. Our "translationStepSize" and "rotationStepSize" parameters guarantee that our intermediate target pose is "close" to our current pose, so we can reasonably linearize our problem to be Eq. 4, where J_c and J_r are the Cartesian and rotation Jacobian defined above, and $\varDelta_c$ and $\varDelta_r$ are the desired translation and rotation to move the end-effector to the intermediate target. In our experiments, we use a 6DOF robot, so we can use ordinary least-squares regression to compute the required joint changes necessary to satisfy that equation. For a robot with more degrees of freedom, regularization could be used to prefer solutions with small joint angle changes.

$$\begin{bmatrix} J_c \\ J_r \end{bmatrix} \Theta = \begin{bmatrix} \Delta_c \\ \Delta_r \end{bmatrix} \tag{4}$$

We find that this simple architecture works for the majority of our feeding tests. Since this gradient-based algorithm is greedy, we do note that it is possible for the robot arm to follow trajectories into local minima where joint constraints prevent the robot arm from continuing all the way to the user's mouth. In our setup, we find that this would generally happen if the trajectory brings the robot end-effector too close to the robot base. Placing the user and plate so that the path between them stays more than a certain distance from the robot base circumvents this issue and leads to successful trajectories, though we also implement a more robust solution that modifies planned trajectories to keep away from the robot base while moving.

Learning from Demonstration. In addition to defining trajectories from the plate to the mouth and back, we also need to train the robot to acquire food from the plate. To do so, we use Learning from Demonstration to imitate human utensil trajectories. In this manner, we do not need to perform the complex task of modeling different food types in order to plan a food acquisition strategies. The ability of Learning from Demonstration to plan trajectories without an explicit model of the domain dynamics [20] is an attractive benefit for us in this context. "Learning from demonstration" in the robotics context usually means some mechanism of automatically acquiring knowledge from human demonstration, but for this work it was sufficient to pick and imitate a single trajectory from a collection of trajectories for assistive feeding. In [21], Bhattacharjee et al. collected detailed fork trajectories in a simulated assistive feeding environment. We visualize several of those trajectories in Rviz and select and truncate a single fork trajectory that follows a simple scooping motion in acquiring coleslaw. Our robot imitates that single fork trajectory to acquire food with a spoon. We find that single scooping trajectory to be sufficient for scooping up the different food types we tested, but we expect that there are food types and serving plate configurations for which more nuanced trajectory learning is required.

To have our robot spoon replicate the training trajectories, we want the robot to move its arm in such a way that the tip of the spoon follows the demonstrated tip of the fork in both position and orientation. In addition, we want the robot to be able to translate the demonstration trajectory target by various offsets so that the robot can scoop food from different parts of the serving plate. To accomplish this, we use the same gradient-based planner described above, where now the target position and orientation of our gradient-based controller are generated by playing back the recorded utensil tip poses, offset by the desired translation. We slow down the played-back demonstration to one fifth the demonstration speed due to speed constraints of the robot arm and to ensure safety.

4 Experimental Setup and Results

4.1 Hardware Setup

To test our feeding algorithm in a real robotic setting, we use a 6 degree-of-freedom Kinova MICO robot affixed rigidly to a dining table. We place a serving bowl of food in front of the robot and have the robot hold a spoon in its end-effector. We attach an iDS-xs RGB camera to the end-effector pointed toward the bowl of the spoon. Our user sits at the table, and across from the user we place a Kinect V1 RGB-D camera on a tripod so that it can see the robot arm and the user's face. The two cameras publish ROS topics which are used in our planning architecture.

4.2 Calibration of Camera to Robot

In order for the robot to use the information from the RGB-D camera in a meaningful way, we have to calibrate the robot base frame with the RGB-D camera frame. We want to do this calibration in an automatic way so that we can use the feeding system quickly after changing the position of the camera or the robot arm.

To automatically calibrate the robot frame with the camera frame, we attach an orange ball to the robot arm end-effector and then we move the robot arm to different positions. These positions are known in the robot base frame because we know the robot kinematics. For each of the positions, we acquire the corresponding RGB-D images from the camera. We use color segmentation to detect the ball in those images and get its corresponding 3D points. We fit a sphere to the 3D points of the ball using RANSAC to compute the 3D center of the ball in the camera frame. Finally, using these 3D correspondences between the camera frame and the robot base frame, we fit a rotation and a translation by solving an Orthogonal Procrustes Problem [22].

4.3 Mouth Tracking Accuracy Over Time

Setup. To test the performance of our tracking system, we acquired an RGB-D video of a person moving his head in front of the camera. In the first part of the video, we ask the user to look in different directions but to keep his torso still. In the second part, we ask the user to move his head to different places in addition to looking in different directions.

Since DO requires a 3D face model, we acquire the face model of the user using MTCNN to detect the face and DBSCAN [23] to filter the model point cloud.

The error of our tracking system was measured by annotating the mouth point in our dataset at a frame-rate of 2 FPS and then measuring the error between the output mouth point of our tracking and the annotated label in each frame.

Results. The error in each frame is presented in Fig. 4. The experiment was performed on a single thread on a Intel Core i7-6500U CPU 2.50 GHz with 8 GB of memory. Our tracking achieved speeds of 8–10 FPS.

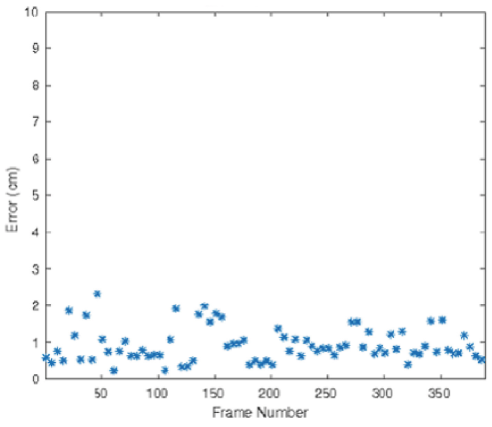


Fig. 4. Tracking error in each frame

Figure 4 shows that our tracking method can track the person's mouth with an error below 3 cm through the entire video. This is on the order of magnitude of the mouth's size, so the output of our system will not be far from the real location of the mouth.

To have a better visual insight about the output of our tracking method, we show in red on Fig. 5 the output mouth point of our algorithm.

As discussed previously, our tracking works well for rotations and translations of the user's head. With our method we can give an accurate position of the user's mouth.

4.4 Food Detection

Setup. We also analyze the performance of the food detector. In this section, we evaluate the performance of our two methods for food detection. To train the classifiers and the linear regression we split our dataset of spoon images into 70% train data and 30% test data.

Results. Figure 6 shows the linear regression fit modeling the weight of food in the spoon with the correlation of histograms between the current image of the spoon and the image of the empty spoon. The regression has a validation mean squared error of 1.37 g and a correlation coefficient (r^2) of 0.75 on the test set. There is a strong negative correlation between the weight of the food that is on the spoon and the correlation between the histograms. Even if the model is not

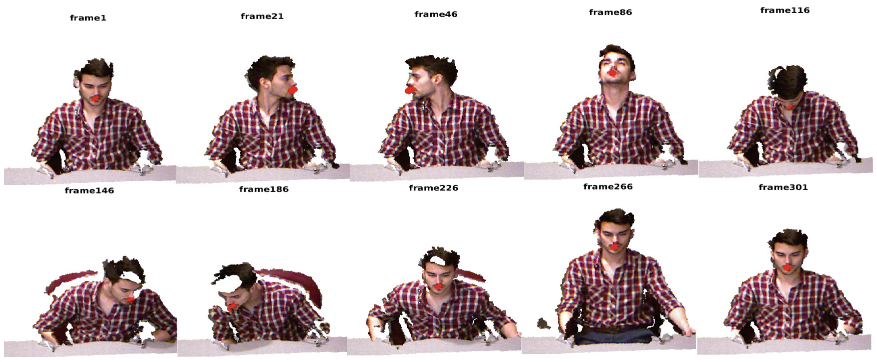


Fig. 5. Tracking output in red for the user's mouth. (Color figure online)

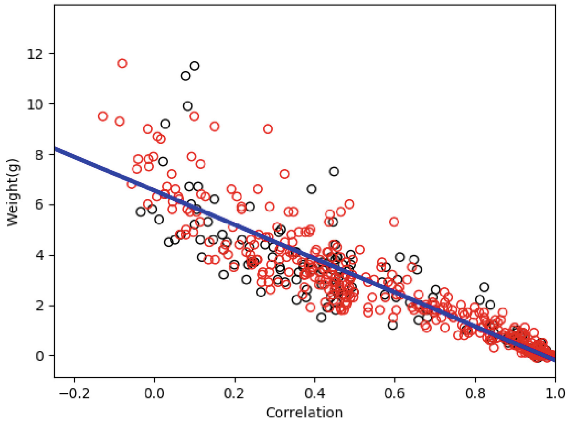


Fig. 6. Relationship between the correlation measure and the weight of food on the spoon. The correlation measure is computed as shown in Eq. 3 by taking the histogram of the image of the spoon with food on it and correlating that with the histogram of the baseline image of the spoon without food on it. The data points used for training are shown in red, the data used for test is shown in black, and the linear regression is shown in blue. (Color figure online)

perfect in measuring the amount of food that is on the spoon, we can use this model to calibrate our correlation threshold to require more or less food per bite.

In the classification task, we achieved an accuracy of 95.8% with the logistic regression and 98.6% with the linear SVM. The SVM was trained using the whole histogram and the logistic regression using only the correlation between each sample of the train data and an empty spoon histogram example. We present, in Figs. 7 and 8, the confusion matrix results for the two classification methods. These results validate our classification approach to discover if there is still food on the spoon.

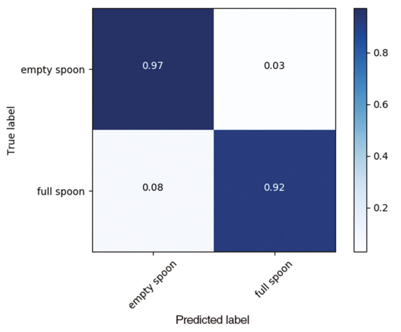


Fig. 7. Confusion matrix for logistic regression

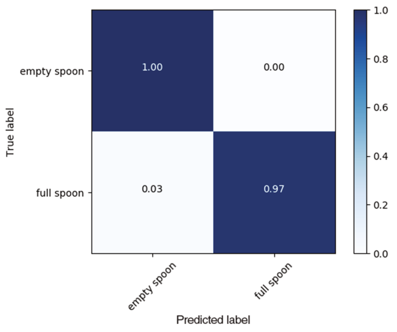


Fig. 8. Confusion matrix for SVM

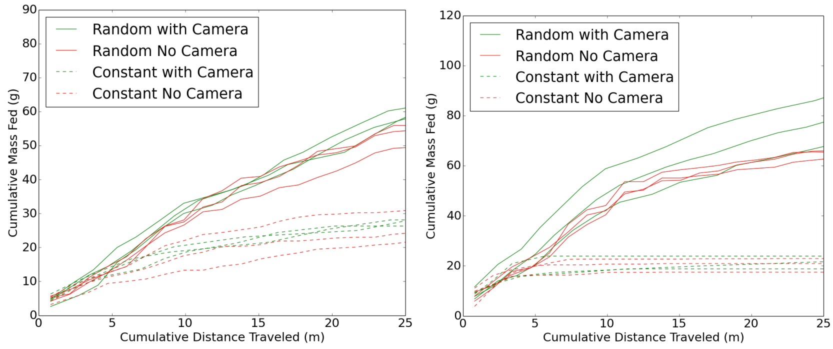
4.5 Ablation Study for Food Acquisition

Setup. To analyze the importance of various components of the food acquisition system, we perform an ablation study in the feeding task where we measure the efficiency of the feeding robot when it does and does not use certain food acquisition strategies. We define the efficiency to be the mass of food that is delivered to the feeding location as a function of the total distance that the tip of the spoon has traveled. We report the mass as a function of distance traveled rather than the time taken because we want to negate any effect that setting the robot to a faster or slower speed would have on the results.

The system components that we alter in our ablation study are (1) whether or not we re-scoop if the spoon-facing camera detects that not enough food was acquired and (2) whether the robot always scoops from the same position on the serving plate or whether it scoops from a uniformly random position within a 6 cm $\times$ 3 cm rectangle on the plate. We perform the ablation experiment on two different kinds of food: peanuts and fried rice. In this way, we can determine if the importance of system components depends on the type of food.

For consistency in results, we use the same random seed for all trials that randomize the scooping location on the plate. To speed up data collection, the robot dumps the food directly onto the scale after food acquisition. Then, in our data analysis we add in the distance to the user's mouth and back. Cutting out the travel time between the plate and the user's mouth cuts the data acquisition time by a factor of three. On average, the distance traveled by the spoon tip in a single scooping motion is 32 cm and the average distance to and from the user's mouth is 49 cm in each direction. We use these average distance values when representing the amount of food served as a function of distance.

Results. In our ablation study for food acquisition, we find that vision feedback significantly increases the amount of food brought to the user in each bite. For the spoon-facing vision feedback system, we use a histogram correlation cutoff of 0.5 for both rice and nuts. For rice, when randomizing the scooping location, the average amount of rice served per bite in the first 20 bites across three trials is 4.8 g with camera feedback and 3.2 g without camera feedback (paired t-test p-value 9e−6). Likewise for nuts it is 3.8 g with camera feedback and 2.7 g without (paired t-test p-value 1e−5). There is an added distance that the end-effector travels in re-scooping when using camera feedback. Despite this added cost in distance, Fig. 9 shows that even when looking at the amount of food served as a function of the total distance that the spoon tip travels (including the added distance required to re-scoop), after 25 m, random scooping with camera feedback consistently outperforms random scooping without camera feedback.



(a) Feeding Nuts (b) Feeding Rice

Fig. 9. Charts showing the amount of food fed (in grams) as a function of the distance the tip of the spoon has traveled, including the distance traveled in re-scooping. The chart includes results for scooping location randomization (solid lines) compared to a constant scooping location (dotted lines), and it includes results with (green lines) and without (red lines) camera feedback on whether food was successfully acquired. We ran three trials for each test type. (Color figure online)

For both rice and nuts we find that randomization of the scooping location is a useful technique in food acquisition. Figure 9 also shows that randomization coupled with visual feedback on the success of a scoop is the most effective. Most interestingly, we find that the importance of our system components depends on the type of food used. We find that randomizing the location of scoops is more important when serving rice instead of nuts. This could be related to the fact that nuts were observed to "settle" after each scoop, whereas fried rice will tend not to fill in the hole left after scooping. Thus, scooping the same place for fried rice will quickly result in very little (almost no) mass acquired by the spoon in subsequent scoops.

5 Conclusion and Future Work

In this work, we explored how vision can be introduced in a robotic feeding platform. By introducing a vision-based approach we can make the system more intelligent and also more autonomous. The developed system is fully capable of feeding a user using a spoon with different types of food like rice or peanuts, and uses visual feedback to ensure spoons are full of food when presented to the user.

Our results show that DO approach can be used effectively for tracking a user's face. However, the method still needs improvements when generalizing to different users from the one for which we have the 3D face model. In future work, we are planning to test DO against other head-pose estimation approaches like OpenFace [24]. We also want to incorporate the user's facial expression information to detect the user's intention to be fed.

Using a camera that can see what is on the spoon, we are capable of ensuring there is an adequate amount of food per bite. We can also use the proposed classifiers to see if there is food on the spoon after the user's bite. Given the effectiveness in the spoon-facing vision feedback in identifying the mass of food on the spoon, we also plan to investigate using that vision feedback system to control the depth that the spoon digs into the food, so that the feeding system can alter its serving strategy to accommodate the gradually decreasing level of food in the serving bowl.

Using this vision framework, we hope in future work to personalize our system to different users and make our system learn with individual user experience. We plan to test our system in real-world situations with people living with disabilities and plan to use a broader range of food types, potentially needing to improve our food detection and acquisition algorithm. We hope to quantitatively and qualitatively analyze the efficacy of our entire system for several users. We will also compare our algorithm to the type of baseline algorithm found in current commercially-available robots.

Acknowledgements. We very specially thank Henny Admoni for generously making available the arm robots in her HARP lab for our work. We specially thank Jayakorn Vongkulbhisal for the help with the implementation of DO. We thank the CMU-Portugal program for their partial support of this work through an ERI grant, as well as for supporting the visit of Alexandre Candeias to CMU, which led to this exciting and successful joint collaboration and work. This work was also funded by FCT grant [UID/EEA/50009/2013] and FCT project IF/00879/2012 (Projecto Exploratório). Finally, the authors thank the anonymous reviewers for their comments. The views and conclusions of this document are those of the authors only.

References

1. Al-Halimi, R.K., Moussa, M.: Performing complex tasks by users with upper-extremity disabilities using a 6-DOF robotic arm: a study. IEEE Trans. Neural Syst. Rehabil. Eng. **25**(6), 686–693 (2017)
2. Topping, M.: Early experience in the use of the 'handy 1' robotic aid to eating. Robotica **11**(6), 525–527 (1993)
3. Obi: Obi, robotic feeding device designed for home care. https://meetobi.com. Accessed 19 May 2018

4. Camanio Care: Bestic. http://www.camanio.com/us/products/bestic/. Accessed 09 May 2018
5. Performance Health: Meal buddy. https://www.performancehealth.com/meal-buddy-systems. Accessed 18 Feb 2018
6. Park, D., Kim, Y.K., Erickson, Z.M., Kemp, C.C.: Towards assistive feeding with a general-purpose mobile manipulator. CoRR abs/1605.07996 (2016)
7. Park, D., Kim, H., Hoshi, Y., Erickson, Z., Kapusta, A., Kemp, C.C.: A multimodal execution monitor with anomaly classification for robot-assisted feeding. In: 2016 IEEE International Conference on Robots and Systems (IROS) (2017)
8. Hawkins, K.P., Grice, P.M., Chen, T.L., King, C.H., Kemp, C.C.: Assistive mobile manipulation for self-care tasks around the head. In: IEEE SSCI 2014–2014 IEEE Symposium Series on Computational Intelligence - CIR2AT 2014: 2014 IEEE Symposium on Computational Intelligence in Robotic Rehabilitation and Assistive Technologies, Proceedings, pp. 16–25 (2014)
9. Perera, C.J., Lalitharatne, T.D., Kiguchi, K.: EEG-controlled meal assistance robot with camera-based automatic mouth position tracking and mouth open detection. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 1760–1765. IEEE (2017)
10. Schröer, S., et al.: An autonomous robotic assistant for drinking. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 6482–6487. IEEE (2015)
11. Vongkulbhisal, J., De La Torre, F., Costeira, J.P.: Discriminative optimization: theory and applications to point cloud registration. In: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, pp. 3975–3983, January 2017
12. Silva, C., Vongkulbhisal, J., Marques, M., Costeira, J.P., Veloso, M.: Feedbot - a robotic arm for autonomous assisted feeding. In: Oliveira, E., Gama, J., Vale, Z., Lopes Cardoso, H. (eds.) EPIA 2017. LNCS (LNAI), vol. 10423, pp. 486–497. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65340-2_40
13. Herlant, L.V.: Algorithms, implementation, and studies on eating with a shared control robot arm. Ph.D. dissertation, Carnegie Mellon University (2016)
14. Ragusa, F., Tomaselli, V., Furnari, A., Battiato, S., Farinella, G.M.: Food vs. non-food classification. In: Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management - MADiMa 2016, pp. 77–81 (2016)
15. Besl, P.J., McKay, N.D.: A Method for Registration of 3-D Shapes (1992)
16. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, vol. 1, pp. I-511–I-518 (2001)
17. Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Sig. Process. Lett. **23**(10), 1499–1503 (2016)
18. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. **9**(1), 62–66 (1979)
19. Diankov, R.: Automated construction of robotic manipulation programs. Ph.D. thesis, Carnegie Mellon University, Robotics Institute, August 2010
20. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. **57**(5), 469–483 (2009)
21. Bhattacharjee, T., Song, H., Lee, G., Srinivasa, S.S.: Food manipulation: a cadence of haptic signals. arXiv preprint arXiv:1804.08768 (2018)
22. Eggert, D.W., Lorusso, A., Fisher, R.B.: Estimating 3-D rigid body transformations: a comparison of four major algorithms. Mach. Vis. Appl. **9**, 272–290 (1997)

23. Daszykowski, M., Walczak, B.: Density-based clustering methods. Compr. Chemom. **2**, 635–654 (2010)
24. Baltrusaitis, T., Robinson, P., Morency, L.P.: OpenFace: an open source facial behavior analysis toolkit. In: 2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016 (2016)