Implementing Tactile Behaviors Using FingerVision

Akihiko Yamaguchi*,** and Christopher G. Atkeson**

***Abstract*— We explore manipulation strategies that use vision-based tactile sensing. FingerVision is a vision-based tactile sensor that provides rich tactile sensation as well as proximity sensing. Although many other tactile sensing methods are expensive in terms of cost and/or processing, FingerVision is a simple and inexpensive approach. We use a transparent skin for fingers. Tracking markers placed on the skin provides contact force and torque estimates, and processing images obtained by seeing through the transparent skin provides static (pose, shape) and dynamic (slip, deformation) information. FingerVision can sense nearby objects even when there is no contact since it is vision-based. Also the slip detection is independent from contact force, which is effective even when the force is too small to measure, such as with origami objects. The results of experiments demonstrate that several manipulation strategies with FingerVision are effective. For example the robot can grasp and pick up an origami crane without crushing it. Video: `https://youtu.be/L-YbxcyRghQ`**

I. Introduction

We explore the use of a vision-based tactile sensor FingerVision [1] in robotic manipulations. FingerVision is a simple, physically robust, and inexpensive tactile sensor. Although some sensing modalities of FingerVision are inferior to that of humans, FingerVision provides other modalities that humans cannot perceive. We develop four manipulation strategies with FingerVision: gentle grasping (grasping with a small force), holding (controlling a gripper to avoid slip), handover (opening a gripper when passing an object to a human), and in-hand manipulation (changing the orientation of an object without releasing it).

The conceptual illustration of FingerVision is available in [2]. It consists of a transparent soft layer with markers on the surface, a transparent hard layer, and cameras. By tracking the markers on the surface of the skin, we can estimate the contact force distribution. The camera can see through the skin since the skin is transparent. Analyzing the image gives us information about nearby objects (proximity sensing). We present methods for proximity sensing that provide slip detection, object detection, and object pose estimation. Slip is detected as a set of moving points on the object in an image. Since our slip detection is vision-based, it can sense slip even when the object is very lightweight, such as grasping origami objects.

We also emphasize that in contrast to expensive tactile sensing technologies, such as the BioTac sensor [3], FingerVision is simple (making a sensor is easy and inexpensive, and force estimation and proximity vision are done by combining OpenCV (`http://opencv.org/`) functions), but it is still useful for robotic manipulations. We are working on making our technology open source [2] so that many projects can consider FingerVision as a tactile sensor.

*Yamaguchi is with Graduate School of Information Sciences, Tohoku University, 6-6-01 Aramaki Aza Aoba, Aoba-ku, Sendai 980-8579, Japan `info@akihikoy.net`

**Yamaguchi and Atkeson are with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213, United States

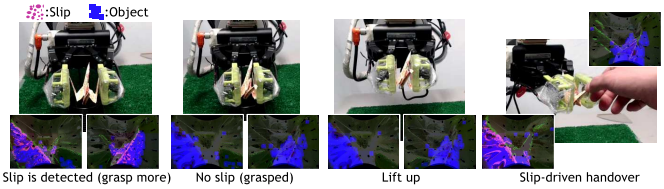

Fig. 1. Examples of tactile behaviors using FingerVision. Three images from left: picking up an origami crane. Right: handover activated by slip.

Recently using deep learning for robotic grasping has become popular [4], [5], [6]. In these studies, tactile sensing was not used. Vision-based grasping was possible because there is a consistent relation between the state (visual information) before grasping including the grasping parameters and the outcome of grasping. Tactile sensing is intermediate information, which is not necessary to use in learning grasping behavior. However tactile perception is useful in many manipulation scenarios, such as, grasping a container whose contents are unknown, and manipulating objects whose surface friction is unknown. Tactile sensing should make robotic manipulation more robust and reduce the number of samples needed to learn.

We tested several manipulation strategies with many different objects. The behaviors worked well. Remarkable results are: grasping a paper business card on edge and passing it to a human without bending it, holding a marshmallow being pulled on by a human, and picking up an origami crane without crushing it (Fig. 1).

Related Work

Slip detection has decades of research. An approach to use acoustic signals caused by slip was explored in [7]. A popular approach is using the vibration caused by slip [8], [9], [10], [11], [12], [13]. Some vibration approaches used accelerometers [9], [12]. Approaches to create a mechanism for making slip-detection easier are considered in [9] (soft skin with a texture), [8] (soft skin covered with nibs), and [11] (a flexible link structure). In [14], [15], [16], they analyzed an observed force (and torque) to detect slip. Many studies detect slip by using a distributed sensor array [17], [18], [19]. In [19], 44x44 pressure distribution is converted to an image, and slip is detected by an image processing. In [20], a multi-sensor fusion approach was proposed where they combined stereo vision, joint-encoders of the fingers, and fingertip force and torque sensors. In [21], they developed slip detection using center-of-pressure tactile sensors. Some researchers use the BioTac sensor [3]. In [22], two BioTac sensors are used and several strategies to detect slip are compared experimentally. BioTac sensors are also used in [23], where they developed three types of tactile estimation: finger forces, slip detection, and slip classification. Similar to ours, [24], [25], [16] take a vision-based approach to

detect slip. The sensor proposed in [16] provides a high-resolution force array, and slip detection with it. Unlike ours, [16] covered the skin with an opaque surface. Our approach is simpler than many of above studies, but is robust since proximity vision gives a direct measurement of the grasped object pose and movement. Our slip detection does not depend on contact force estimation, but uses a direct measurement of object movement obtained by vision. This is possible because our sensor has a highly transparent skin unlike other vision-based tactile sensors (e.g. [26], [16]). Therefore it is applicable even when the contact force is too small to measure. Relative object pose estimation is also a unique feature of our approach, and we have found it useful for many behaviors.

There is a huge literature on human grasping and slip detection. Recent surveys are [27] and [28]. An important early work on the role of slip in human grasping is [29] which describes a holding behavior where grip force is increased when slip is detected. In addition to launching the field, this work eventually led to the design of the BioTac sensor [3]. Recent papers following up on this work include [30], which discusses how to hold very light objects, and [31], which discusses how grip force is increased during arm movements. [32] discusses the effects of object inertia and gravity on the selection of grip force. [33] reviews human grip responses to perturbations of objects during precision grip. [34] studies human to human handover behaviors.

Creating robotic manipulations with tactile sensing is also studied. A human-like gentle grasp without slip and crush was proposed in [15]. A grasping strategy using slip detection is studied in [35]. [23] created a behavior to gently pick up objects where they combined finger force estimates, and slip detection and classification. Some work focused on grasp control for avoiding slip [8], [24] where they used slip detection. A more complicated task was considered in [20] where they used slip detection in a peg-in-hole task. [19] made a method to estimate a contact area and slip, and in-hand manipulation with it. Our study presents different approaches of manipulations with tactile sensing. It is especially remarkable that the use of slip to grasp a light-weight fragile object (e.g. origami crane) where the contact force is too small.

II. Grasping Strategies with FingerVision

We have created several manipulation strategies using force estimation and proximity vision of FingerVision: **Gentle grasping**: Grasping an object gently by using force estimation. This is useful when grasping a fragile object. **Holding**: Controlling the gripper to avoid slip. This is especially useful when grasping a deformable and fragile object. It is also effective for grasping light-weight fragile objects. **Handover**: Opening the gripper when a force change or slip is detected. This is useful when passing an object to humans. **In-hand manipulation**: Change the orientation of a grasped object by repeatedly relaxing and tightening the gripper based on the slip estimate. In order to demonstrate the usefulness of FingerVision, we present examples of these manipulations. For simplicity, we use position control of our grippers. In the following behaviors, a small movement of the grippers means a position command to create a minimum movement.

A. Gentle Grasping

The behavior is closing the gripper until one of the FingerVision sensors on the fingers senses a sufficient contact force. FingerVision provides an array of forces (each marker gives 3-dimensional force estimate $[f_x, f_y, f_z]$). Rather than using an average force or torque to detect a small force, detecting a small force on each marker is better in this scenario. For robustness against marker tracking noise, we programmed force tracking as follows: We categorize $|f_y|$ (norm of the normal force) into 4 types: noise level, sufficient contact force, medium force, large force, and give scores 0, 1, 3, 5 respectively. Manually defined thresholds are used in this categorization. We defined the condition to stop closing the gripper as that the sum of the scores of the array exceeds a threshold (7 worked well in our experiments).

B. Holding

The behavior is that the robot slightly closes the gripper when the FingerVision sensors sense slippage, otherwise no action is performed by the gripper. For slip detection we use the number of moving points on the object in the image. If the number exceeds a threshold, it is recognized as a slip event.

Note that the holding strategy enables a robot to grasp very light-weight fragile objects such as origami. The idea is that if there is not enough friction between the object and the fingers, the object will slip when the robot moves the hand. Using the holding strategy until there is no slip, the robot will be able to move the object without slip. This approach is applicable even when force estimation cannot sense the contact force from the object. Thus this could be a strategy to grasp light-weight fragile objects.

C. Handover

We assume that the gripper already grasps an object, i.e. there are forces applied to the FingerVision sensors. For force change detection, we compare the force estimate on each marker with its initial value. We count the number of markers where a difference between those two values exceeds a threshold. The force change is defined as a flag which is "on" when the number is greater than a threshold (e.g. 5). The slip detection is the same as that used in the holding behavior.

D. In-hand Manipulation

We assume that the gripper already grasps an object. The robot repeats the following process until the target angle is achieved. The robot slightly opens the gripper until if senses a small slip. Since there would be a small delay between the gripper motion and slippage, we insert a short waiting time (0.1 s) after each gripper command. The method to detect slip is the same as that in the holding behavior, but the threshold is halved (i.e. more sensitive). After a short waiting time or when slip is detected, the robot closes the gripper until slip is not detected.

III. Proximity Vision

Proximity vision processes an image to obtain information about nearby objects, such as object colors, textures, shape, position and orientation, movement including slippage, and deformation. This paper focuses on approximate detection of an object and its movement. Simple approaches to detecting movement are optical flow and background subtraction.

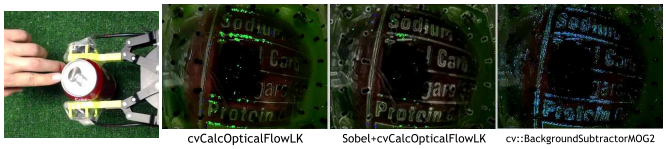

Fig. 2. An example of the comparison of three functions.

Movement detection involves detecting movement of the environment and the robot body. For example when moving the robot arm, the camera in FingerVision will capture background change. Operating the gripper also causes background change. We need to distinguish the movement of an object from background change. We developed a detection and tracking method for an object, as well as movement detection.

For simplicity, we model an object with a histogram of colors. In most grasping scenarios, a robot gripper approaches an object, or another agent passes an object to the gripper. In both cases, the object is seen as a moving object in the cameras of FingerVision. Thus, we design the object detection and tracking as follows. First we create a background model as a histogram of colors. At the beginning of grasping, we detect moving blobs in the image, compute a histogram of colors of the moving pixels, and subtract the background histogram. The remaining histogram is added to the object model. In the tracking phase, we apply the back projection of the object histogram to the current frame, and thresholding to detect the object. We describe more details in the following.

A. Movement Detection

We found that optical flow and background subtraction are good at detecting changes in a sequence of images. We compared three implementations based on functions in OpenCV, applying `cvCalcOpticalFlowLK` to raw images, `cvCalcOpticalFlowLK` to edge images detected by the Sobel filter, and `cv::BackgroundSubtractorMOG2` to raw images. In many cases, the three approaches provided similar results. In some cases, `cv::BackgroundSubtractorMOG2` was slightly better than the others (Fig. 2). We used the background subtraction approach for movement detection.

B. Object Model Construction

The object model construction consists of two phases. One is the construction of a background model, which is performed at the beginning of the experiments. The other is the construction of an object model, which is performed during each grasping action. Both background and object models are histograms of colors. We use the hue and saturation components of the HSV color space to construct the histograms, where the number of bins of hue and saturation components are 100 and 10 respectively.

The background model is constructed with several adjacent frames (e.g. 3). We simply make it as an average of histograms of all frames. Let us denote the background histogram model as $H_{\mathrm{bg}}(h, s)$ where h and s denote indexes of hue and saturation bin respectively.

During construction of an object model, the object is assumed to be moving in the image as we described above. At each frame, we detect the moving points with the background subtraction method, and calculate the histogram of colors as $H_{\mathrm{mv}}(h, s)$. We update the object histogram model by:

$$H'_{\mathrm{obj}}(h,s) = \min(255, H_{\mathrm{obj}}(h,s) + f_{\mathrm{gain}} \max(0, H_{\mathrm{mv}}(h,s) - f_{\mathrm{bg}} H_{\mathrm{bg}}(h,s))) \quad (1)$$

where $H_{\mathrm{obj}}(h, s)$ and $H'_{\mathrm{obj}}(h, s)$ are the current and the updated object histogram models. At the beginning, $H_{\mathrm{obj}}(h, s)$ is initialized to be zero. The component $\max(0, H_{\mathrm{mv}}(h, s) - f_{\mathrm{bg}} H_{\mathrm{bg}}(h, s))$ computes the remaining histogram after subtracting the background histogram from the color histogram of moving points. The $\min(255, \dots)$ operation is for normalization. f_{bg} and f_{gain} are constant values, for example 1.5 and 0.5 respectively.

In order to simplify the timing to start and stop object model construction, we use an object model made with the latest 200 frames. We stop object model construction when the robot starts closing the gripper.

C. Object Tracking

At each frame, we track an object by detecting the pixels similar to the object model. Concretely, we apply a back projection method (`cv::calcBackProject`) with the histogram of the object $H_{\mathrm{obj}}(h, s)$, and threshold the result to remove the uncertain pixels. The remaining pixels are the detected object. These pixels are used in two ways. One is removing the background change from the moving points obtained from the background subtraction. For this purpose we apply an erosion (size 2) and a dilation (size 7) to remove noise and expand the boundary of the object. The other is computing the position and the angle (orientation) of the object. This is done by computing the moment of the object pixels. Examples of proximity vision are shown in Fig. 3.

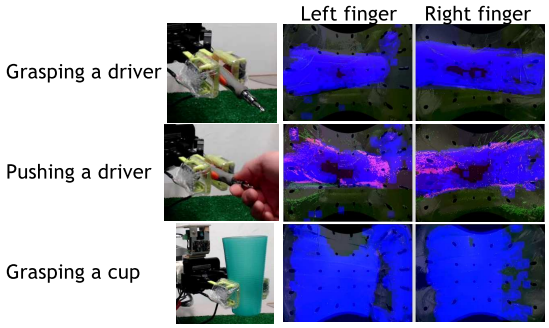

Fig. 3. Examples of proximity vision. In each case, the detected object is shown as blue. In pushing a screw driver, a human pushed the object which caused slip. The detected slip is emphasized by the purple color. We can also see green particles that are pixels detected as moving. They are considered as the background movement since they are outside the detected object region.

IV. Force Estimation By Marker Tracking

In [1] where we introduced a prototype of FingerVision, a simple approach to track the markers was used. We applied a blob detection method to the entire image in each frame, and found the matching blobs between initial and current frames to track the markers. When some blobs were not detected, matching failed or large blob movements were shown. In this paper we present an improved marker tracking approach.

We consider two approaches for marker tracking. One is using the mean shift method to track the marker movement. Initial marker positions are obtained by blob detection. For

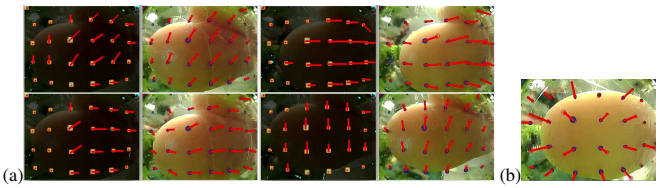

Fig. 4. (a) Comparison of blob tracking based on `cv::meanShift` (left of each pair) and `cv::SimpleBlobDetector` (right of each pair). (b) An example of marker movements when a normal force is applied.

each marker, we apply mean shift starting from the previous marker position to obtain the current marker position. The other approach is applying blob detection locally for each marker. We consider a small region around the previous marker position, and apply blob detection to obtain the current marker position.

Both methods are implemented in OpenCV. The mean shift method is available as the `cv::meanShift` function, and blob detection is available as the `cv::SimpleBlobDetector` class. We thought the mean shift approach would be better since it is a common tracking method. According to our preliminary test, the marker tracking with mean shift was robust. However it turned out that this approach does not provide good accuracy of marker position since `cv::meanShift` returns an updated object location as integer values. Since the marker movement on the image is small (a few pixels), the movement was jumpy. On the other hand, `cv::SimpleBlobDetector` provides the detected blob position as floating-point values. The obtained marker position movement was smooth (see the comparison in Fig. 4(a)). Thus we choose the blob detection-based approach.

The actual procedure consists of two phases: calibration and tracking. In both phases, we preprocess the image by rectifying the distortion caused by the fisheye lens, and thresholding to extract black colors as the current markers are black. We also apply a dilation and an erosion to remove noise.

Calibration: The sensor is covered with a white sheet to remove the background. We apply blob detection method to an entire image. Then we apply the tracking method to several frames (e.g. 10); if some markers are moving, they are removed from the marker candidates as they are noisy points. The remaining blobs are considered as initial markers.

Tracking: Starting from the initial marker positions, we track each marker frame by frame. We consider a small (e.g. 30x30) region of interest (ROI) around the previous marker position. First we count the non-zero pixels in the ROI and compare it with the non-zero points of the initial marker. If there is large difference, we do not perform marker tracking (i.e. a detection failure). Otherwise we apply the blob detection method to the ROI. Only one blob is expected; otherwise it is considered as a failure. We compare the previous and current blob positions and sizes, and if their difference is large, it is considered as a failure. Otherwise the blob is considered as the new marker location.

A. Force Estimation

From the marker movement, we estimate an array of forces. The blob detection provides a position and a size of each blob. The position change is caused by a horizontal (surface) force, while the size change is caused by a normal force. However, since the size change is subtle compared to the position change as reported in [1], the normal force estimate based on the size change is noisy and unreliable. An alternative approach approximates the normal force at each marker with a norm of marker position change. This approximation is useful especially when taking an average of all the forces. When a normal force is applied to the center of the skin surface, the markers around the point move radially (Fig. 4(b)). An average of the horizontal forces in such a case will be close to zero, while an average of the approximated normal forces will have a useful value. Let d_x, d_y denote the horizontal marker movement from the initial position. The force estimate at each marker is given by: $[f_x, f_y, f_z] = [c_x d_x, c_y \sqrt{d_x^2 + d_y^2}, c_z d_y]$ where c_x, c_y, c_z denote constant coefficients. We also define an average force and a torque estimate as: $\mathbf{f} = \frac{1}{N} \sum [f_x, f_y, f_z]$, $\boldsymbol{\tau} = \frac{1}{N} \sum \mathbf{r} \times [f_x, f_y, f_z]$ where N denotes a number of markers, and $\mathbf{r}$ denotes a position of a marker from the center of the image.

V. FingerVision

This section describes an improvement of FingerVision in manufacturing (using a 3D printer for molds for silicone casting, and frames to install the sensor on a gripper), and installation of four sensors on a robot (sensor network). See [1] for the design parameters such as the marker arrangement.

A. Making FingerVision

The prototype of FingerVision proposed in [1] was manufactured with hand-made molds whose dimensions were inaccurate, and manufacturing was inconsistent. Moreover, the soft layer was easily peeled from the hard layer. Installation on the robot gripper was also handcrafted, which caused mechanical play during use.

We designed a frame to attach the hard layer on the finger of a gripper. The frame has a place to attach the hard layer made with transparent acrylic, and a connection structure to the gripper. The latter part depends on the gripper. We created two versions: one is for the electric parallel gripper of a Baxter robot (i.e. a standard gripper), and the other is for the Robotiq 2-finger adaptive robot gripper-85 (legacy version). Since we have CAD data of the fingers, we can easily connect to the frame. The Robotiq gripper has a mount on the fingertip under the original finger pad. We made a structure to attach the frame to the mount. The frame also has a mount for a camera (ELP Co. USBFHD01M-L180, fisheye lens USB camera). We use a 3D printer (LultzBot Mini, Aleph Objects, Inc.) for producing the frames where we use PolyLite PLA filament.

The soft layer is made by casting silicone (Silicones Inc. XP-565). We make a mold for the casting using a 3D printer for consistent manufacturing. However we noticed that the surface of 3D printed objects are not smooth enough to make optically clear skin even after smoothing with sandpaper. Thus we use a 3D printed mold except for the surface part of the soft layer. For the surface part, we use ComposiMold.

In order to reduce the peeling of the soft layer, we create depressions and holes on the sides of the frames so that the silicone locks into them, and we cover the hard layer on both top and bottom with the silicone (see Fig. 5(c)).

The manufacturing process is as shown in Fig. 5. Fig. 6(a) shows our Baxter robot with the FingerVision sensors installed.

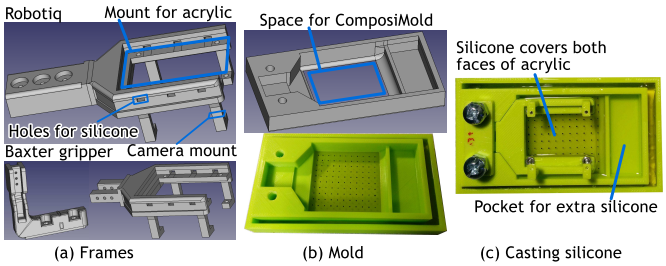

Fig. 5. (a) CAD of frames for Baxter electric parallel gripper and Robotiq gripper. (b) Mold for silicone casting. (c) After pouring silicone into the mold.

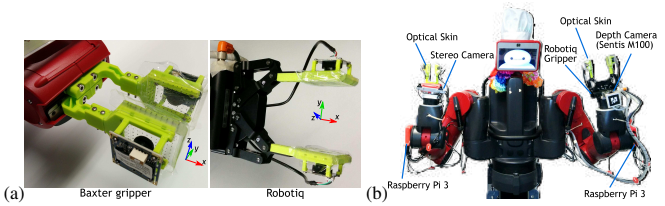

Fig. 6. (a) FingerVision installed on Baxter electric parallel gripper and Robotiq gripper. (b) Our Baxter system with four FingerVision sensors.

B. Sensor Network

The cameras we use have a USB 2.0 interface. The local computers send images obtained from cameras to a central computer, and the central computer processes all the images. In our particular case, we use Raspberry Pi 3s as the local computers, and transmit data through a Gigabit Ethernet network. We use MJPG-streamer[1] installed on each Raspberry Pi to capture images from cameras, and transmit them using a motion JPEG format. We use two Raspberry Pi 3 computers each of which has two camera connections. In our test, they could transmit 320x240 images at 63 FPS from four cameras simultaneously. Fig. 6 shows our Baxter system.

VI. Experiments

We test the FingerVision sensors with force estimation and proximity vision in the proposed manipulation behaviors. Although they work with both the Baxter electric parallel gripper and the Robotiq gripper, we use the Robotiq gripper here. We use the cameras at 30 FPS with resolution 320x240. Most of the scenes are shown in Fig. 7 with the force estimation and the proximity vision views. See also the accompanying video.

A. Evaluating Force Estimation

We evaluate the force estimation using a scale. First we let the robot push the scale vertically to evaluate f_y. Second we let the robot hold a stick and push the scale with it in order to evaluate f_z. Similarly we evaluate f_x by changing the stick and pushing direction. In each case we discretely increase the pushing force from around 1 [N] to 20 [N]. Fig. 8 shows the results. The values of the weight scale are linearly scaled and offset. We noticed that there was hysteresis. There were two sources of noise: marker tracking and the robot control.

[1]We use a forked version: `https://github.com/jacksonliam/mjpg-streamer`

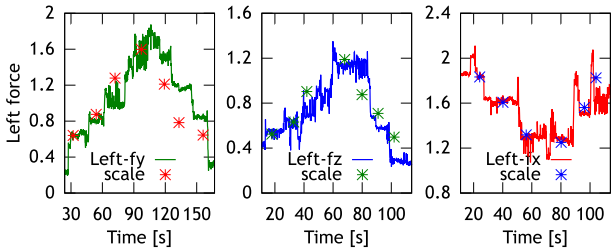

Fig. 8. Average force trajectories in evaluating f_y (left), f_z (middle), and f_x (right) respectively. The $*$ mark the scale readings (linearly scaled, and offset). The unit of force is omitted as it is not calibrated as engineered units.

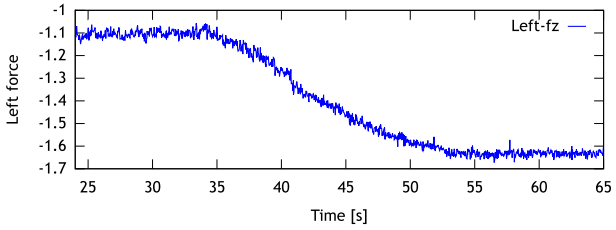

Fig. 9. Vertical component (z-axis) of the force estimate during the pouring-water experiment. Actual pouring is from 35 to 52 [s]. The unit of force is omitted as it is not calibrated as engineered units.

B. Pouring Water into a Grasped Container

We have the gripper grasp a container, and then pour water into the container manually. Fig. 9 shows the vertical component (z-axis) of the force estimate. Pouring was performed from 35 [s] to 52 [s]. The force gradually increased. This would be accurate enough to estimate the poured amount of water during a pouring task.

C. Proximity Vision

We explore basic results from proximity vision. We let the gripper grasp a screw driver weakly, and move it in the gripper manually. Then we let the gripper grasp an empty Coke can, and poke the can 4 times. Fig. 10 shows the result of rotating the screw driver. We can see that the object angle changes from zero to negative, to positive, and goes back to zero again. The object angle measured by an external camera is also plotted in the figure. Around the peaks, the object angle is different from the estimate by proximity vision. This was because around these angles, a part of the object was out of the camera view. During rotating the screw driver, there are positive movement values that are capturing the slippage. The torque estimate sensed the external torque that rotated the screw driver. Fig. 11 shows the result of poking the Coke can. Since the Coke can was light weight, the human poked very weakly. The force and torque estimates did not capture the poke. However the proximity vision detected the movement as we can see four peaks in the graph that correspond with the four pokes.

D. Gentle Grasp

We test the gentle grasp strategy. We have the robot grasp an empty Coke can, and grasp a paper business card on edge. Both objects are soft and will be damaged with even small forces. Fig. 12 shows the force and torque estimate, and the proximity vision output during the gentle grasp. The actual grasp happened at 34.5 [s]. We can see a small

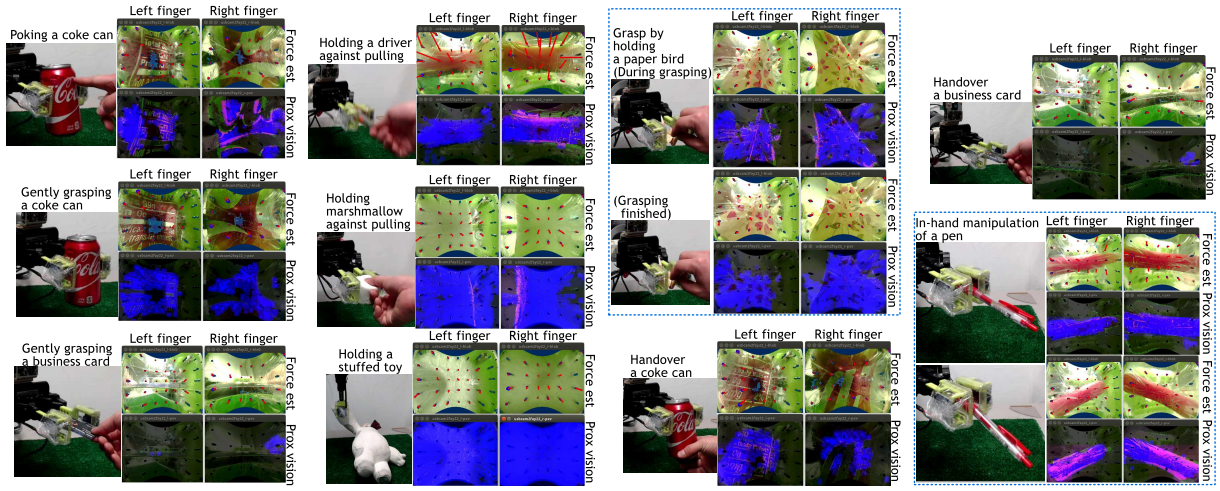

Fig. 7. Scenes of experiments. "Force est" are views of force estimation (red lines show estimated forces), and "Prox vision" are views of proximity vision (blue regions are detected objects, and purple points are detected movements).

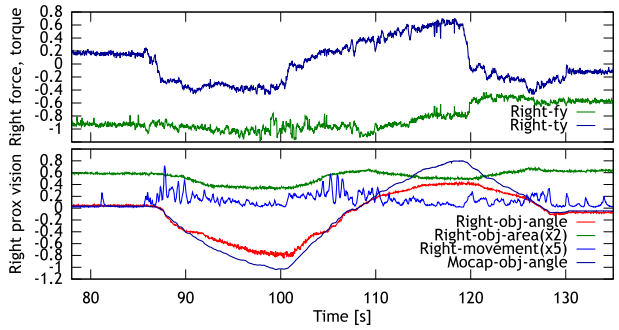

Fig. 10. Force and torque estimate (top) and proximity vision (bottom) during rotating the screw driver. In the proximity vision graph, there are plots of the object angle (radian) and area obtained from the moment of object pixels, and the total number of moving pixels (normalized by the image size). Object angle obtained by an external camera is also plotted (Mocap-obj-angle). The units of force and torque are omitted as they are not calibrated as engineered units.

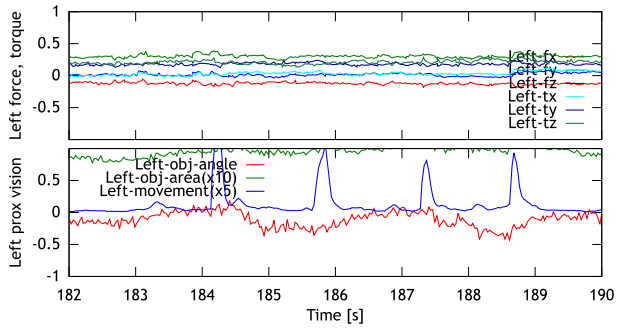

Fig. 11. Force and torque estimate (top) and the proximity vision (bottom) during poking a Coke can. See the caption of Fig. 10 for the plots.

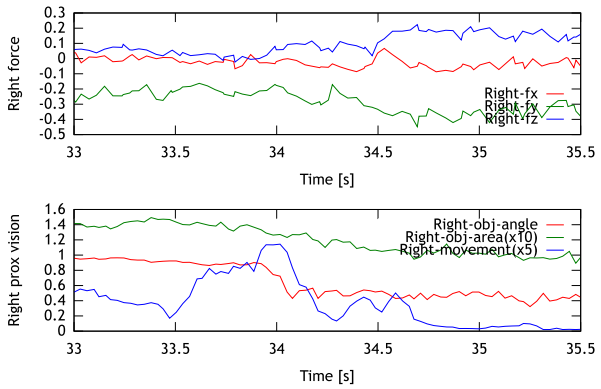

Fig. 12. Force and torque estimate (top) and the proximity vision (bottom) during gently grasping a Coke can. See the caption of Fig. 10 for the plots.

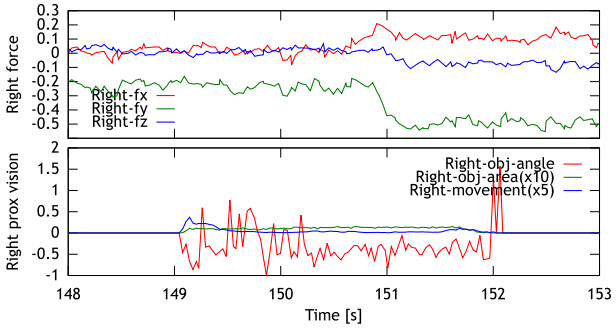

Fig. 13. Force and torque estimate (top) and the proximity vision (bottom) during gently grasping a business card. See the caption of Fig. 10 for the plots.

change of force around that time. We can also see movement detection before and during grasping. This was caused by the approaching motion before the grasp. Similar results are found in the card case as shown in Fig. 13. The actual grasping happened at 151 [s]. The movement detection is less than that of the Coke can case. Since the robot grasped the card on edge, it appeared only in a small region of the image.

E. Holding Strategy

We demonstrate the holding strategy by grasping a screw driver. We compare two patterns: (A) the gentle grasp strategy, and (B) the holding strategy. During grasping with

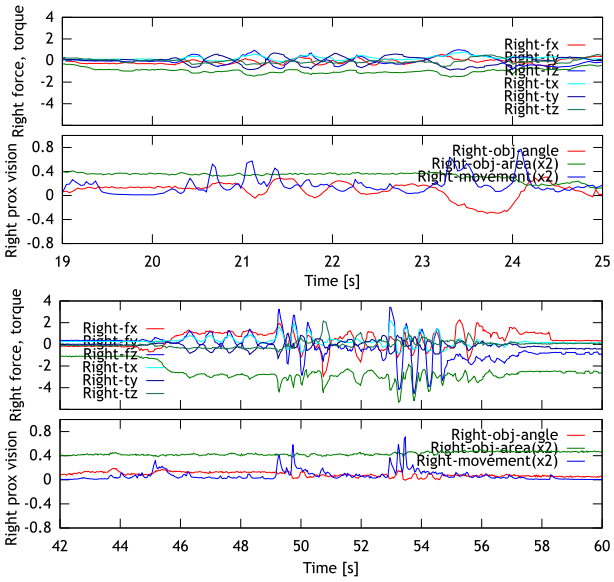

Fig. 14. Results of the holding strategy. Top two graphs are results of the gentle grasp, and bottom two graphs are ones of the holding strategy. In each pair, force and torque estimate (top) and the proximity vision (bottom) are plotted. See the caption of Fig. 10 for the plots.

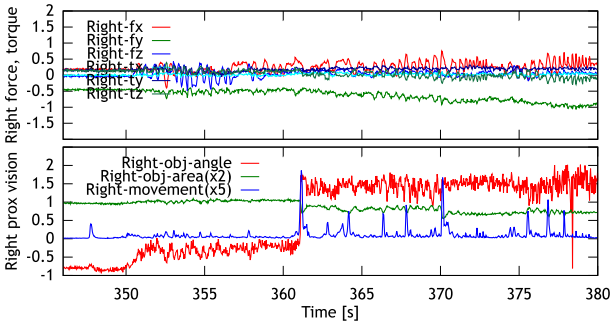

Fig. 15. Force and torque estimate (top) and the proximity vision (bottom) during holding and moving a marshmallow. See the caption of Fig. 10 for the plots.

each pattern, a human pushes the driver several times. The results are shown in Fig. 14. From the graphs of force and torque estimates, we can see that stronger external force was applied in (B). The orientation of the object is changing more in (A). Thus the holding strategy could reduce slip.

We apply the holding strategy to grasp a marshmallow where a human pulls the marshmallow. Fig. 15 shows the result. We can see many slip detections (peaks in Right-movement) from the bottom graph, and the magnitude of grasping force ($|f_y|$) is increasing accordingly in the top graph.

Next we let the robot move a stuffed toy. Moving with the gentle grasping strategy, the robot dropped the toy due to a slip. However by activating the holding strategy, the robot could hold and move the toy. Fig. 16 shows the force and torque estimates and the proximity vision output during the motion. We find that there are several discrete events of slippage, and after each of them, the grasping force (see f_y) was increased. At 545 [s], the robot passed the object to the human. The area of the object in the image, and the force and torque estimates became zero after that.

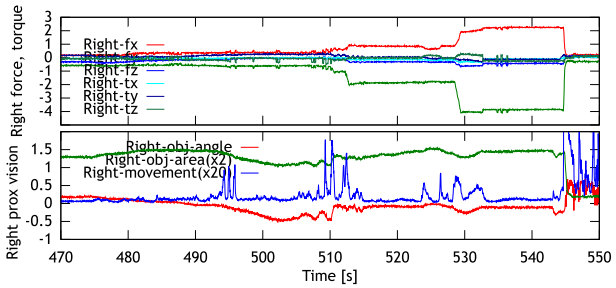

Fig. 16. Force and torque estimate (top) and the proximity vision (bottom) during holding and moving a stuffed toy. See the caption of Fig. 10 for the plots.

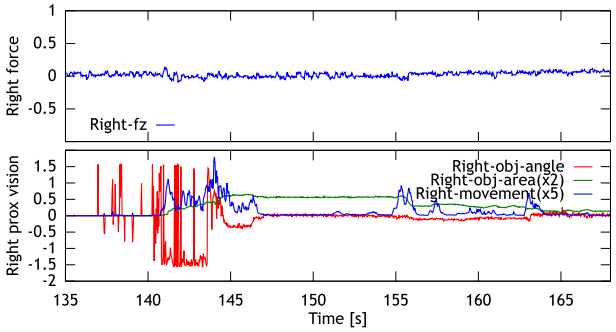

Fig. 17. Force estimate (top) and the proximity vision (bottom) during grasping a paper bird. See the caption of Fig. 10 for the plots.

F. Grasping A Fragile Object with the Holding Strategy

We demonstrate that using the holding strategy the robot can grasp a very light-weight fragile object. As such an object, we use an origami crane. A human passes an origami crane to the gripper, and the robot uses the holding strategy. After grasping it without slip, the robot swings its arm to see if the holding strategy is effective. Fig. 17 shows the result. The robot performed grasping from 142 [s] to 145 [s]. We can see slip detection around 155 [s] and so on, but the object was kept inside the gripper. From the force and torque estimates, we cannot see informative changes. This was due to the small weight of the object (1.7 g).

G. Handover

We demonstrate the handover strategy by applying it to a Coke can and a business card. Both objects are grasped by the gentle grasping strategy, and the card is grasped on edge. Fig. 18 shows the result. In the Coke can case, the robot started to open the gripper triggered by the slip detection at 25.1 [s]. In the business card case, the opening gripper was triggered by the force change detection at 160 [s]. The reason could be that the Coke can is slippery, while the slip detection does not work well with the card when it is grasped on edge. We also investigated other object cases, and found that when the robot grasped an object strongly, the force-trigger was often used since the slip rarely happened with such grasps.

H. In-hand Manipulation

We apply the in-hand manipulation strategy to rotating a pen. The target angle is 20 degrees from the current angle. Fig. 19 shows the results of eight runs. In most cases the

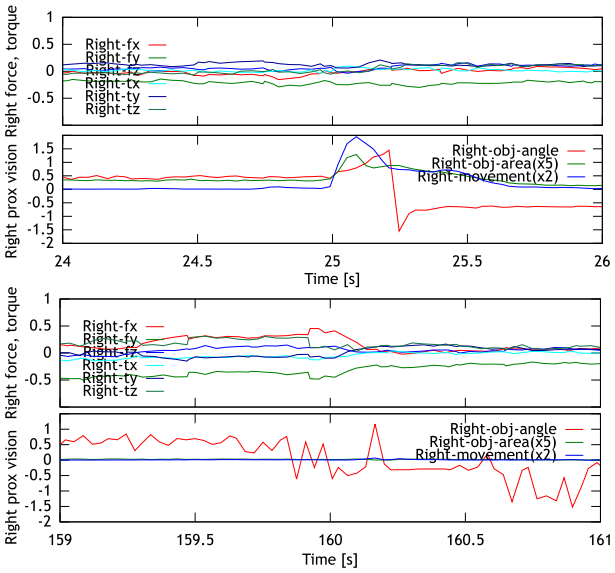

Fig. 18. Results of the handover strategy. Top two graphs show the result of the Coke can case, and bottom tow graphs show the result of the card case. Each of them have force and torque estimates (top) and the proximity vision (bottom). See the caption of Fig. 10 for the plots.

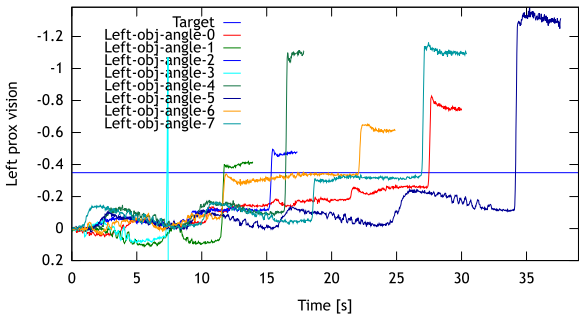

Fig. 19. Result of in-hand manipulation. The object orientations obtained by the proximity vision are plotted per time. The initial orientation is set to be zero.

achieved angles exceeded the target. This was because the object movement caused by gravity was fast and the sensing and processing frame rate was not enough to respond to that, and the gripper response was not fast enough.

VII. Conclusion

We explored four manipulation strategies that used tactile sensing: gentle grasping, holding, handover, and in-hand manipulation. We used a simple vision-based approach, FingerVision, an optical multimodal-sensing skin for fingers that we proposed in [1]. We developed image processing methods for FingerVision (proximity vision) to provide slip detection, object detection, and object pose estimation. We improved the hardware design of FingerVision, and force estimation. The results of experiments demonstrated that the manipulation strategies with FingerVision were effective.

Acknowledgment

This material is based upon work supported in part by the US National Science Foundation under grant IIS-1717066.

References

[1] A. Yamaguchi and C. G. Atkeson, "Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables," in *the 16th IEEE-RAS International Conference on Humanoid Robots*, 2016.
[2] A. Yamaguchi, "FingerVision," http://akihikoy.net/p/fv.html, 2017.
[3] R. Johansson, G. Loeb, N. Wettels, D. Popovic, and V. Santos, "Biomimetic tactile sensor for control of grip," Feb 2011, US Patent 7,878,075.
[4] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.
[5] L. J. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *the IEEE International Conference on Robotics and Automation (ICRA'16)*, 2016.
[6] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, 2017.
[7] D. Dornfeld and C. Handy, "Slip detection using acoustic emission signal analysis," in *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, vol. 4, 1987.
[8] M. R. Tremblay and M. R. Cutkosky, "Estimating friction using incipient slip sensing during a manipulation task," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 1993.
[9] R. D. Howe and M. R. Cutkosky, "Sensing skin acceleration for slip and texture perception," in *Proceedings, 1989 International Conference on Robotics and Automation*, 1989.
[10] R. Fernandez, I. Payo, A. S. Vazquez, and J. Becedas, "Micro-vibration-based slip detection in tactile force sensors," *Sensors*, vol. 14, no. 1, pp. 709–730, 2014.
[11] ——, *Slip Detection in Robotic Hands with Flexible Parts*. Cham: Springer International Publishing, 2014.
[12] A. A. S. Al-Shanoon, S. A. Ahmad, and M. K. b. Hassan, "Slip detection with accelerometer and tactile sensors in a robotic hand model," *IOP Conference Series: Materials Science and Engineering*, vol. 99, no. 1, 2015.
[13] R. Fernandez, I. Payo, A. S. Vazquez, and J. Becedas, *Slip Detection in a Novel Tactile Force Sensor*. Cham: Springer International Publishing, 2016.
[14] C. Melchiorri, "Slip detection and control using tactile and force sensors," *IEEE/ASME Transactions on Mechatronics*, vol. 5, no. 3, pp. 235–243, 2000.
[15] J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker, "Human-inspired robotic grasp control with tactile sensing," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1067–1079, 2011.
[16] W. Yuan, R. Li, M. A. Srinivasan, and E. H. Adelson, "Measurement of shear and slip with a GelSight tactile sensor," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
[17] E. G. M. Holweg, H. Hoeve, W. Jongkind, L. Marconi, C. Melchiorri, and C. Bonivento, "Slip detection by tactile sensors: algorithms and experimental results," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, 1996.
[18] N. Tsujiuchi, T. Koizumi, A. Ito, H. Oshima, Y. Nojiri, Y. Tsuchiya, and S. Kurogi, "Slip detection with distributed-type tactile sensor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2004.
[19] V. A. Ho, T. Nagatani, A. Noda, and S. Hirai, "What can be inferred from a tactile arrayed sensor in autonomous in-hand manipulation?" in *2012 IEEE International Conference on Automation Science and Engineering*, 2012.
[20] T. Hasegawa and K. Honda, "Detection and measurement of fingertip slip in multi-fingered precision manipulation with rolling contact," in *Conference Documentation International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2001.
[21] D. Gunji, Y. Mizoguchi, S. Teshigawara, A. Ming, A. Namiki, M. Ishikawa, , and M. Shimojo, "Grasping force control of multi-fingered robot hand based on slip detection using tactile sensor," in *2008 IEEE International Conference on Robotics and Automation*, 2008.
[22] J. Reinecke, A. Dietrich, F. Schmidt, and M. Chalon, "Experimental comparison of slip detection strategies by tactile sensing with the BioTac on the DLR hand arm system," in *2014 IEEE International Conference on Robotics and Automation*, 2014.
[23] Z. Su, K. Hausman, Y. Chebotar, A. Molchanov, G. E. Loeb, G. S. Sukhatme, and S. Schaal, "Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots*, 2015.
[24] A. Ikeda, Y. Kurita, J. Ueda, Y. Matsumoto, and T. Ogasawara, "Grip force control for an elastic finger using vision-based incipient slip feedback," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.
[25] M. Tada, M. Imai, and T. Ogasawara, "Development of simultaneous measurement system of incipient slip and grip/load force," in *Proceedings of 9th IEEE International Workshop on Robot and Human Interactive Communication*, 2000.
[26] K. Nagata, M. Ooki, and M. Kakikura, "Feature detection with an image based compliant tactile sensor," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1999.
[27] D. A. Nowak, *Sensorimotor control of grasping: physiology and pathophysiology*. Cambridge University Press, 2009.
[28] E. Chinellato and A. P. del Pobil, "The visual neuroscience of robotic grasping: Achieving sensorimotor skills through dorsal-ventral stream integration," Ph.D. dissertation, Springer, 2008.
[29] G. Westling and R. S. Johansson, "Factors influencing the force control during precision grip," *Experimental Brain Research*, vol. 53, no. 2, pp. 277–284, 1984.
[30] Y. Hiramatsu, D. Kimura, K. Kadota, T. Ito, and H. Kinoshita, "Control of precision grip force in lifting and holding of low-mass objects," *PLOS ONE*, vol. 10, pp. 1–19, 2015.
[31] G. P. Slota, M. L. Latash, and V. M. Zatsiorsky, "Grip forces during object manipulation: experiment, mathematical model, and validation," *Experimental Brain Research*, vol. 213, no. 1, pp. 125–139, 2011.
[32] O. White, "The brain adjusts grip forces differently according to gravity and inertia: a parabolic flight experiment," *Frontiers in Integrative Neuroscience*, vol. 9, p. 7, 2015.
[33] M. De Gregorio and V. J. Santos, *Human Grip Responses to Perturbations of Objects During Precision Grip*. Cham: Springer International Publishing, 2014.
[34] W. P. Chan, C. A. C. Parker, H. F. M. V. der Loos, and E. A. Croft, "Grip forces and load forces in handovers: Implications for designing human-robot handover controllers," in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2012.
[35] M. Kaboli, K. Yao, and G. Cheng, "Tactile-based manipulation of deformable objects with dynamic center of mass," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016.