International Journal of Humanoid Robotics
Vol. 16, No. 3 (2019) 1940002 (40 pages)

DOI: 10.1142/S0219843619400024

Tactile Behaviors with the Vision-Based Tactile Sensor FingerVision

Akihiko Yamaguchi*,‡ and Christopher G. Atkeson†,§

*Graduate School of Information Sciences,
Tohoku University,
6-6-01 Aramaki Aza Aoba, Aoba-ku,
Sendai, Miyagi 980-8579, Japan

†The Robotics Institute,
Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh PA 15213-3890, USA
‡info@akihikoy.net
§cga@cs.cmu.edu

Received 21 November 2018
Accepted 2 May 2019
Published 1 July 2019

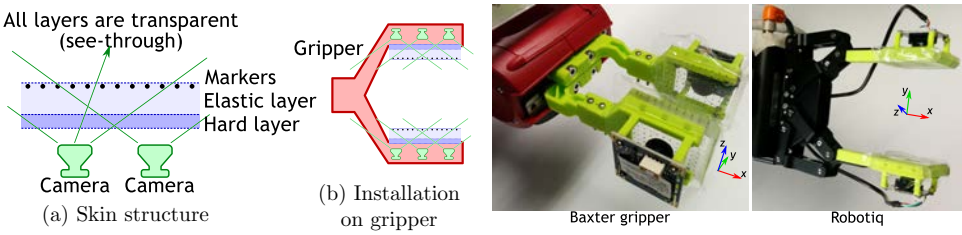
This paper introduces a vision-based tactile sensor FingerVision, and explores its usefulness in tactile behaviors. FingerVision consists of a transparent elastic skin marked with dots, and a camera that is easy to fabricate, low cost, and physically robust. Unlike other vision-based tactile sensors, the complete transparency of the FingerVision skin provides multimodal sensation. The modalities sensed by FingerVision include distributions of force and slip, and object information such as distance, location, pose, size, shape, and texture. The slip detection is very sensitive since it is obtained by computer vision directly applied to the output from the FingerVision camera. It provides high-resolution slip detection, which does not depend on the contact force, i.e., it can sense slip of a lightweight object that generates negligible contact force. The tactile behaviors explored in this paper include manipulations that utilize this feature. For example, we demonstrate that grasp adaptation with FingerVision can grasp origami, and other deformable and fragile objects such as vegetables, fruits, and raw eggs.

Keywords: FingerVision; tactile sensor; tactile behavior.

1. Introduction

We are exploring a vision-based tactile sensor FingerVision.[1] Unlike other vision-based tactile sensors such as TacTip[2] and GelSight,[3] FingerVision has a transparent skin that enables an embedded camera to see through the skin. This feature increases the modalities obtained by the sensor, including vision of nearby objects and slip distribution. We explore if FingerVision is a promising approach to overcome the

‡Corresponding author.



(a) Skin structure (b) Installation on gripper

Fig. 1. Conceptual design of FingerVision (a) and an installation sketch on a robotic gripper (b). Right two images are prototypes of FingerVision installed on a Baxter electric parallel gripper and a Robotiq gripper.

issues of existing tactile sensors, and how we effectively use FingerVision in designing robotic behaviors.

FingerVision consists of a transparent elastic material, a transparent hard layer, and cameras. On the surface of the elastic material, small dots are placed to track the deformation of the material with computer vision. The conceptual diagram is shown in Fig. 1. By processing the video data from the cameras of FingerVision, we can obtain tactile sensation and vision of nearby objects (*proximity vision*). The features of FingerVision can be summarized as follows:

(1) Multimodal: It can sense force distribution, high-resolution slip distribution, object distance, location, pose, size, shape, texture, and other information obtained from proximity vision.
  (1.1) Slip can be detected regardless of the force on objects. It can sense slippage even when the object is too light to generate measurable force (e.g. origami).
  (1.2) Cameras can sense objects before collision. With this feature, we can create safe interactive robots that are aware of nearby humans and fragile objects.
(2) Easy to fabricate: Because of its simple structure, its fabrication is easy.
(3) Low cost: The most expensive component is the camera, which is a low cost webcam. Other components are also inexpensive.
(4) Physically strong: External force is applied to the skin and frame, and does not reach the camera. Thus, it is physically strong.
(5) Easy to repair: Even if the skin is damaged, replacing it is inexpensive.
(6) Using wide-angle lenses (fisheye lenses), we can place the cameras sparsely distributed under the surface where we want to install tactile sensing.
(7) Sensor parameters are adjustable: We can adjust the dynamic range of force (hardness and thickness of the skin), size (small cameras miniaturize the sensor size), spatial resolution (camera resolution, marker allocation, etc.), and temporal resolution (high speed cameras).
(8) Other types of sensing components can be used, such as range finders and thermal cameras.

(9) Open source: The fabrication process including CAD files of frames and molds, software, and tutorials are available online[4] in order to encourage people to reproduce FingerVision for their own robots and projects.

Compared to the other vision-based tactile sensors, the features (1.1), (1.2), (8) are unique to FingerVision because of its transparent skin.

In this paper, we demonstrate the use of FingerVision, especially tactile manipulation with FingerVision, in order to show its usefulness. The tactile behaviors presented in this paper are simple to program. Even so, some behaviors are dramatically improved because of the advantages of FingerVision. For example, the grasp adaptation with FingerVision is very sensitive because of (1.1). FingerVision can adapt a grasp to a range of objects, including lightweight, deformable, and fragile ones.

Parts of this paper were published in conference papers; an early prototype,[1] tactile behaviors,[5,6] and grasp adaptation.[7] The purpose of this paper is providing a comprehensive understanding of tactile behaviors with FingerVision.

The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 introduces FingerVision. Section 4 describes the tactile behaviors. Section 5 reports the results of experiments. Section 6 is a discussion section, and Sec. 7 concludes the paper.

2. Related Work

2.1. *Tactile sensors in general*

There are many different approaches of tactile sensing, such as capacitive sensors, piezoresistive sensors, magnetic sensors, piezoelectric sensors, optical and proximity sensors, and vision-based sensors. Some of them are commercialized such as BarrettHand,[8] PR2,[9] and ReFlex Hand.[10] More comprehensive reviews are available.[11,12] When we design multimodal tactile sensing for robot fingers, we need to deal with issues of fabrication, wiring, power, size, installation, expense, and physical robustness. We think the vision-based approach is a good approach, since: (1) achieving high resolution (superhuman resolution) is not difficult, (2) the sensor structure can be simple and fabrication is not difficult, (3) wiring is not problematic by using well-established network infrastructure, (4) buying the parts and fabrication equipment is affordable, (5) the sensing device (camera) is becoming smaller, cheaper, reliable, and better in resolution and speed, due to the markets for smart phones and endoscopic surgery, and (6) physically robust since the sensing device can be isolated from skin deformation.

2.2. *Vision-based tactile sensors*

The idea of using imaging sensors for tactile sensing is decades old. An initial attempt was measuring the frustration of total internal reflection within a waveguide on a

sensor surface caused by contact.[13–16] The research trend has shifted to measuring displacement of markers placed on the sensor surface with computer vision, such as using a lattice pattern,[17] two-colored dots,[18] a single dot,[19] and single-colored dots.[2,20–25] Marker displacements are proportional to the external force as the displacements are directly caused by the external force. The resolution of the contact force field is decided by the camera resolution and the marker density. The dynamic range of the force measurement can be controlled by changing the hardness of the elastic material (softer is more sensitive; Ref. 26).

Similar to the above work, GelSight was developed by Johnson and Adelson.[27] It consists of a transparent elastomer covered with a opaque skin, which is sensitive to the surface texture and shape of a contacting object. There has been an application to robotic manipulation tasks,[3] shear force and slip estimation with markers,[28] and a slenderized fingertip (GelSlim).[29]

Most of previous vision-based sensors including GelSight occluded the view beyond the sensor itself. This simplifies computer vision since the background of the image is simplified. It makes sensing robust against external lighting conditions and object appearance. Some studies used functional membranes. For example, GelSight used a reflective membrane,[27] which was effective to sense the object texture in high resolution. An exception was proposed by Patel and Correll[30] where a completely transparent skin was used. However, it used an array of range finders to measure the distance to an object and the skin deformation rather than using an imaging optical device.

In contrast, FingerVision uses cameras to view objects of interest and relies on computer vision to separate objects from the background. Although it could be a disadvantage, there are good computer vision functions in publicly available libraries such as OpenCV.[a] More importantly, making all of the skin transparent gives FingerVision another modality, *proximity vision*. The sensitivity of measuring slip is much improved with this approach as discussed below.

2.3. *Slip detection with tactile sensors*

Slip detection has been studied for decades. An early approach used a mechanical roller to detect slip.[31] An approach using acoustic signals caused by slip was explored.[32] A popular approach is using the vibration caused by slip.[33–38] Some vibration approaches used accelerometers.[34,37] Approaches to create a mechanism for making slip-detection easier are considered, such as soft skin with a texture,[34] soft skin covered with nibs,[33] and a flexible link structure.[36] In Refs. 28, 39 and 40, they analyzed an observed force (and torque) to detect slip. Many studies detect slip by using a distributed sensor array.[41–43] In Ref. 43 a 44×44 pressure distribution is converted to an image, and slip is detected by image processing. In Ref. 44, a multisensor fusion approach was proposed where they combined stereo vision, joint-encoders of the fingers, and fingertip force and torque sensors. In Ref. 45, they

[a] http://opencv.org/.

developed slip detection using center-of-pressure tactile sensors. In Ref. 46, two BioTac[47] sensors are used and several strategies to detect slip are compared experimentally. BioTac sensors are also used in Ref. 48, where they developed three types of tactile estimation: finger forces, slip detection, and slip classification.

Similar to ours, Refs. 22, 28 and 49 developed methods to detect slip for vision-based tactile sensors. In Refs. 22 and 49, slip was estimated from the stick ratio (a ratio of areas of stick and contact regions). In Ref. 22, the stick ratio was estimated from the displacement of dotted markers. The GelSight work[28] developed a method to detect slip by thresholding the entropy of shear (marker) displacement distribution.

In contrast, FingerVision estimates slip by directly analyzing the video from fingertip cameras. Unlike other vision-based tactile sensors mentioned above, FingerVision does not rely on marker displacement. FingerVision can estimate slip even if there are no markers on its surface. This feature makes FingerVision special: it can sense slip of very lightweight objects such as origami whose contact force is too small to measure.

2.4. *Tactile behaviors*

Robotic manipulation with tactile sensing is also studied. A popular task is grasping. Sometimes grasp execution with tactile sensing is referred to as grasp adaptation. There are heuristic behavior designs of grasp adaptation,[50,51] and a human-inspired grasp strategy[40] which was based on the study of grasp strategy of humans.[52] Grasp adaptation is also called re-grasping.[53,54] Grasp adaptation is sometimes designed with a grasp stability estimator that estimates a quality of grasp from tactile sensor readings.[55–57] Typically machine learning approaches are used to construct such estimators, which requires training samples. Using slip sensation to adapt grasp is also a popular approach.[15,33,45,48,49,58–61] For example, the slip detection of an optical tactile sensor was used in grasp adaptation[15] where the grasping force of a robot hand was controlled to avoid slip. An experiment of grasping a paper cup was conducted, where water was poured into it. It was demonstrated that the robot adapted the grasp against the increasing weight of water without breaking the paper cup.

In this paper, we follow a grasp adaptation strategy with slip estimation since it is simple to implement, and we can emphasize the advantage of slip detection sensitivity with FingerVision. It is especially remarkable that grasp adaptation with FingerVision can adapt grasp to a light-weight fragile object (e.g., origami) where the contact force is too small to measure.

Other manipulation studies with tactile sensors are removing a cap of a bottle[15] where slip detection was used, rotating a cylinder with a single-finger robot[62] where an optical tactile sensor[2] was used, in-hand manipulation of a cylinder,[63] peg-in-hole with slip detection,[44] inserting a USB connector into a socket[3] where GelSight was used to estimate the pose of the USB connector in the gripper, and in-hand manipulations.[43] In Ref. 64, contour-following control was learned with tactile sensors and reinforcement learning.

Comparing tactile behaviors with other approaches is difficult since we do not have the same robot (especially the same robotic hand) and the same tactile sensors. Constructing a common baseline would be difficult. Therefore, in this paper, we demonstrate tactile behaviors that are enabled with FingerVision.

3. FingerVision

This section introduces the vision-based tactile sensor FingerVision, its fabrication, and the data processing.

3.1. *Overview*

FingerVision consists of a transparent elastic material made with silicone, a transparent hard layer made with acrylic, and a camera underneath. Some dots (markers) are placed on the surface of the elastic material, that are made with plastic beads. The conceptual diagram is shown in Fig. 1. Unlike other research,[2,18,22] we do not place an opaque material on the surface. The whole skin is transparent except for the markers, and the cameras can see the external scene through the skin. This paper presents a prototype with an RGB camera, but it can be extended with multiple cameras and other types of cameras such as thermal and depth cameras.

The markers are captured by the cameras and tracked. This gives us a 3-axis (x, y, z) force measurement at each marker point. By combining multiple marker measurements, we can estimate torque information. The marker size affects the accuracy of tracking. In general, a bigger marker is easier to detect. The density of the markers determines the resolution of the contact force field. There is a trade-off between the resolution and the surface transparency. The hardness and the thickness of the elastic layer affect the marker movement caused by contact force (a softer layer is more easily deformed by a small force), and determine the dynamic range of the contact force measurement. The hard layer is assumed to be fixed on the gripper so that external force is applied to the elastic and hard layers only and does not affect the cameras. The physical robustness of the FingerVision sensor is decided by the elastic and the hard layers. The camera resolution affects the accuracy of the marker detection and tracking. The camera frame rate affects the sensing frame rate. These properties (the marker size and density, the hardness and the thickness of the elastic and the hard layers, and the camera properties) should reflect the purpose (task) of each part of the skin. Multiple layers of different materials allow us to create a "hardening" spring or nonlinear compliance.

3.2. *Specification of prototype*

As the elastic material, we use silicone, Silicones Inc. XP-565 that has A-16 Shore hardness after cure. The effective thickness of the elastic material is 4 mm. The thickness of the acrylic plate is 2 mm.

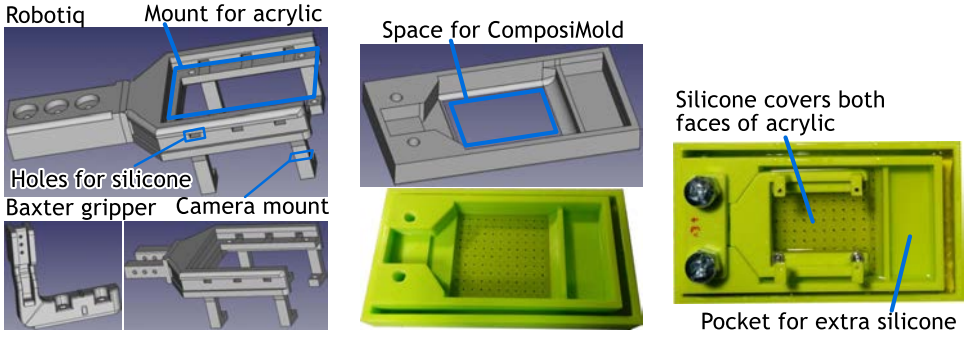
For markers, we use black micro plastic beads that are spheres of around 1 mm diameter. The size varies from 0.5 mm to 1.5 mm. The markers are placed on a 5 mm grid.

We use a fisheye lens camera ELP Co. USBFHD01M-L180 that has a USB interface. It can capture at many different resolutions. We use the mode of 320×240 with MJPG compression.

3.3. *Fabrication of FingerVision*

Fabrication of the FingerVision sensor consists of the following processes: (1) making base frames, molds, and an acrylic plate, (2) placing markers on the mold, (3) mixing silicone resin and pouring it into the mold (note: degassing with vacuum is necessary to remove air bubbles before pouring), (4) inserting the base frame with the acrylic plate into the resin in the mold, (5) fixing the base frame on the mold, (6) waiting for the silicone to cure, (7) removing the silicone and the frame from the mold, and finally (8) attaching the camera. For the stability of the fabrication and installation on robotic hands, we design the frames and the molds with a 3D printer. Figure 2 shows the 3D printed frames, molds, and after casting the silicone (at the beginning of (6)).

We design 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 create two versions: one is for the electric parallel gripper of a Baxter robot (a standard gripper), and the other is for the Robotiq 2-finger adaptive robot gripper-85 (legacy version). Since we have CAD data for 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. We use a 3D printer (LultzBot Mini, Aleph Objects, Inc.) for producing the frames.



(a) Frames (b) Mold (c) Casting silicone

Fig. 2. (a) CAD of frames for a Baxter electric parallel gripper and a Robotiq gripper. (b) Mold for silicone casting. (c) After pouring silicone into the mold.

The soft layer is made by casting silicone. We make a mold for casting using a 3D printer to achieve consistent fabrication. However we noticed that the surfaces 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 increase the durability of the soft layer from peeling, 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. 2(c)).

Figure 1 shows our Baxter robot with the FingerVision sensors installed.

3.4. *Computer vision for processing FingerVision data*

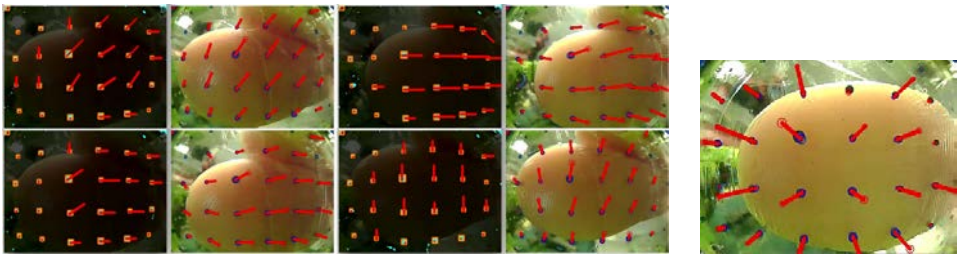
Since the sensing element of FingerVision is a camera, the raw data from FingerVision is an RGB video stream. We use computer vision methods to process the video. There are two types of computer vision for FingerVision video. One is marker tracking that estimates the marker displacements from the initial positions. The displacements of markers are used to estimate the force distribution. The other is proximity vision that consists of nearby object detection and movement detection. The object detection and tracking provides information about a manipulated object, such as location, pose, area, and texture. It is also used to distinguish the movement of a manipulated object and the background.

3.4.1. *Marker tracking*

We consider two approaches for marker tracking. One is using the mean shift method to track marker movement. Initial marker positions are obtained by blob detection. For 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, marker tracking with mean shift was robust. However, it turned out that this approach does not provide good marker position accuracy 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. 3(a)). Thus, we chose 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



(a) (b)

Fig. 3. (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.

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 due to environmental noise, they are removed from the marker candidates as they are noisy points. Typically only a few points are removed. The remaining blobs are considered as initial markers. Note that during the calibration, we do not move the robot and the white sheet is fixed on the sensor surface. If we do not remove these points, they will be observed as noisy movements, which affects the accuracy of force estimate.

Tracking: Starting from the initial marker positions, we track each marker frame by frame. We consider a small (e.g., 30×30) region of interest (ROI) around the previous marker position. First we count the nonzero pixels in the ROI and compare it with the nonzero points of the initial marker. If there is a 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 a failure. We compare the previous and current blob positions and sizes, and if their difference is large, it is considered a failure. Otherwise, the blob is considered as the new marker location.

Post Processing: 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, 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. 3(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

$$[f_x, f_y, f_z] = [c_x d_x, c_y \sqrt{d_x^2 + d_y^2}, c_z d_y], \tag{1}$$

where c_x, c_y, c_z denote constant coefficients. Note that f_y is the normal force (see Fig. 1 for the coordinate system). We also define an average force and a torque estimate as

$$\mathbf{f} = \frac{1}{N} \sum [f_x, f_y, f_z], \tag{2}$$

$$\boldsymbol{\tau} = \frac{1}{N} \sum \mathbf{r} \times [f_x, f_y, f_z], \tag{3}$$

where N denotes a number of markers, and $\mathbf{r}$ denotes a position of a marker from the center of the image.

3.4.2. *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. 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 what follows.

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

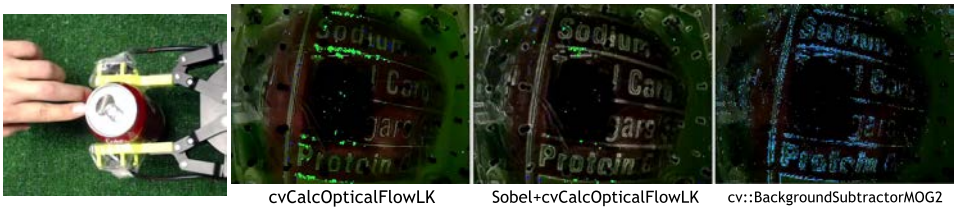


Fig. 4. An example of the comparison of three functions.

approaches provided similar results. In some cases, `cv::BackgroundSubtractorMOG2` was slightly better than the others (Fig. 4). We used the background subtraction approach for movement detection.

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 average the histograms of all frames. Let us denote the background histogram model as $H_{\mathrm{bg}}(h, s)$ where h and s denote 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

$$\begin{aligned} H'_{\mathrm{obj}}(h, s) = \; & \min(255, H_{\mathrm{obj}}(h, s) \\ & + f_{\mathrm{gain}} \max(0, H_{\mathrm{mv}}(h, s) - f_{\mathrm{bg}} H_{\mathrm{bg}}(h, s))), \end{aligned} \tag{4}$$

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.

Object Tracking: In 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.

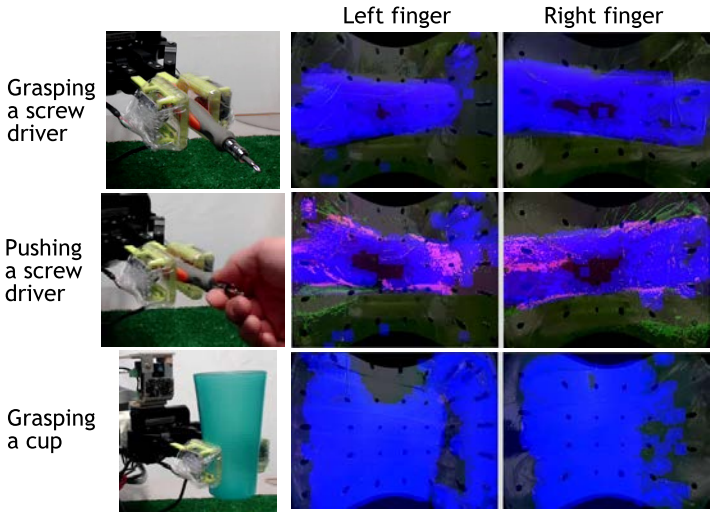


Fig. 5. Examples of proximity vision. In each case, the detected object is shown as blue. In pushing a screw driver, a human pushed the object which caused slip. The detected slip is emphasized by the purple color. We can also see green particles that are pixels detected as moving. They are considered as background movement since they are outside the detected object region.

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. 5.

4. Tactile Behaviors with FingerVision

We have created several tactile behaviors with FingerVision (cf. Fig. 6). In the following, ⟨Behavior⟩ denotes a behavior.

⟨**Gentle Grasp**⟩**:** 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.
⟨**Grasp Adaptation**⟩**:** Automating a lifting-up motion with the slip avoidance control (i.e., ⟨Holding⟩). It enables the robot to pick up a range of objects.
⟨**Handover**⟩**:** Opening the gripper when a force change or slip is detected. This is useful when passing an object to humans.
⟨**Automatic Placing**⟩**:** Placing an object grasped by the robot.
⟨**Automatic Cutting**⟩**:** Automating a part of cutting motion with tactile sensing.

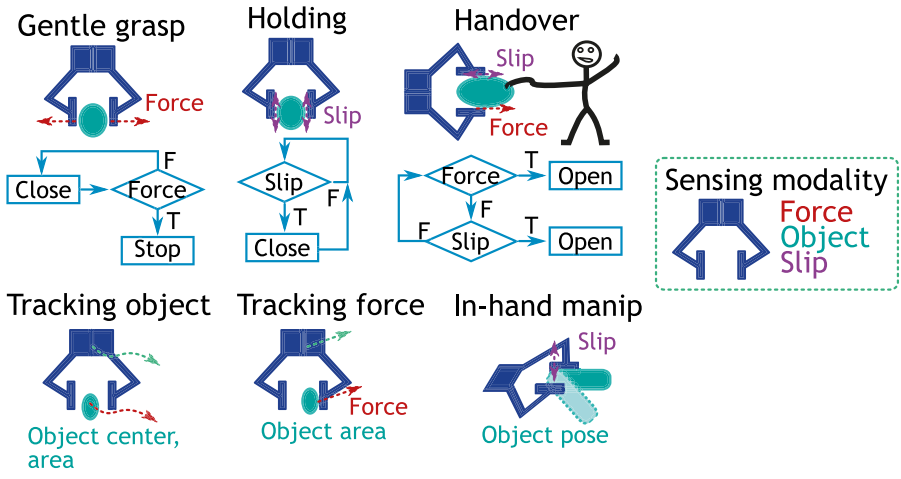


Fig. 6. Conceptual diagrams of tactile behaviors where the modalities used by the behavior and the brief processes are illustrated.

⟨**In-hand Manipulation**⟩**:** Change the orientation of a grasped object by repeatedly relaxing and tightening the gripper based on the slip estimate.
⟨**Tracking Object**⟩**:** Centering an object between the robot fingers.
⟨**Tracking Force**⟩**:** Operating the robot by pushing with a small force.

For simplicity, we use position control on our grippers. In the following behaviors, a small movement of the grippers means a position command to create a minimum movement.

4.1. ⟨*Gentle Grasp*⟩

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 four 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).

4.2. ⟨*Holding*⟩

The behavior is that the robot slightly closes the gripper when the FingerVision sensors detect 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. This strategy is also considered as feedback control of slip.

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.

4.3. ⟨*Grasp Adaptation*⟩

⟨Grasp Adaptation⟩ is a control to adapt grasp to an unknown object where the gripper is controlled to avoid slip, i.e., activating the ⟨Holding⟩ strategy introduced above. Grasping is considered as a control to prevent slip. With a sensor that can detect slip, we can create a control strategy to prevent slip by adjusting the grasping force. As explained in the ⟨Holding⟩ section, this ⟨Grasp Adaptation⟩ will work with a range of objects, from heavy rigid objects to lightweight, deformable, and fragile objects.

In the implementation, a lifting-up motion is executed with slip feedback control (⟨Holding⟩). The robot tries to lift up an object with the slip feedback control for the gripper. If the grasping force is not enough to hold the object, the slip feedback control adjusts the grasp. We refer to this controller as the ⟨Grasp Adaptation⟩ controller. Figure 7 shows the control scheme of ⟨Grasp Adaptation⟩. First, the robot tries to bring up the object (`BringTest`) with the slip feedback control (`slipavd`). `BringTest` is performed slowly so that the gripper can adapt the grasp to the object.

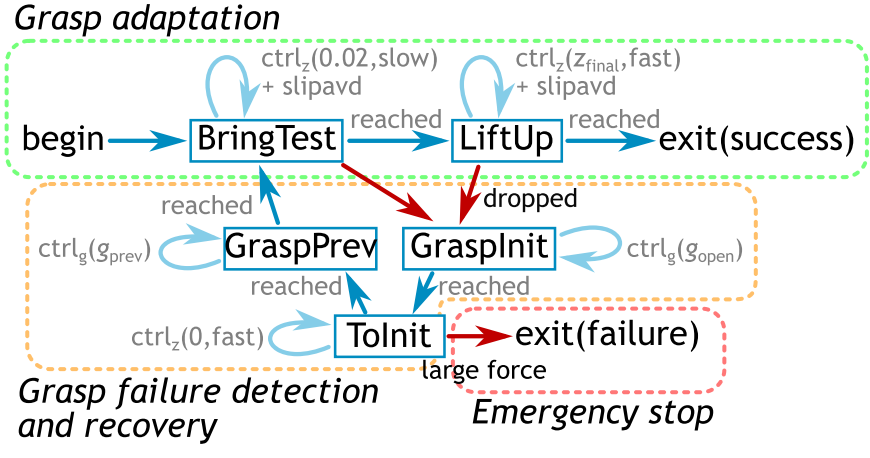


Fig. 7. Control scheme of the ⟨Grasp Adaptation⟩. The detection of grasp failure and recovery, and emergency stop are unified.

Then the robot lifts it up to the final height (`LiftUp`). `LiftUp` is faster than `BringTest`, while the slip feedback control is still active to adjust the grasp.

The above behavior is a one-way procedure normally. The implementation (Fig. 7) has an error recovery. During the above two motions, the robot starts a recovery motion when it detects that the object is considered to be dropped (`dropped`). The detection of object drop is done by thresholding the ratio of the object area over its initial value as `BringTest` starts. If the drop condition is satisfied, the robot opens the gripper (`GraspInit`), moves the gripper to the initial pose (`ToInit`), and closes the gripper to the previous value when the recovery motion started (`GraspPrev`). Then the robot restarts from `BringTest`.

In the state machine Fig. 7, there is another block (Emergency stop), which is activated when FingerVision detects a large force. Such an event is considered to be an exception since the robot grasps nothing and the gripper width is the initial value which should be greater than the object size. An example scenario of such an event is when the robot drops the object and it rolls under the finger. Although the robot finger pushes the object vertically in that case (i.e., only the fingertip of FingerVision contacts the object), FingerVision can still measure the force. This is possible because the camera of FingerVision has a fisheye lens and the elastic material propagates the deformation at the fingertip toward the middle part. In such an event is detected, the state machine is designed to stop immediately.

4.4. ⟨*Handover*⟩

We assume that the gripper already grasps an object, i.e., there are forces applied to the FingerVision sensors. FingerVision is used as a trigger to open the gripper. Both force change estimation and slip detection are used as the trigger: if one of them is detected, the gripper is opened. Combining two modalities increases its applicability. When grasping an object strongly, force tends to be detected. When grasping a light weight object such as an origami crane, slip tends to be detected.

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. When the number exceeds a threshold (e.g., 5), it is considered as the trigger. The slip detection is the same as that used in the ⟨Holding⟩ behavior.

4.5. ⟨*Automatic Placing*⟩

The purpose of this behavior is placing an object grasped by the robot. Tactile sensing is useful to detect an event when the object touches with the ground. Such an event could be estimated with external vision with a model of the object, but there will be uncertainty in estimating the distance between the object and the ground. The approach to use tactile sensing can handle such uncertainty. During the placing motion (the robot moving the gripper downward), the robot stops the movement and

opens the gripper when contact is detected. The implementation of this event detection can be the same as the ⟨Handover⟩ strategy.

4.6. ⟨*Automatic Cutting*⟩

We implement the ⟨Automatic Cutting⟩ motion of fruits with FingerVision. In this paper, we simply use tactile sensing to detect a large force applied to the knife. This detection is useful in two cases. One is detecting the event when the knife reaches the cutting board. Since the knife is often occluded from the robot vision, estimating such an event contains uncertainty. The event detection with tactile sensing can handle such uncertainty. The second is detecting too large force for the gripper to hold the knife. This situation happens when cutting hard materials such as pumpkins.

We create a cutting controller which (a) starts from a state where the knife held by the gripper is put above the material, (b) moves the knife downward (cutting vertically), and then (c) slightly pulls the knife (cutting horizontally). The controller moves the knife to the initial position in order to repeat the motion several times when it cannot cut off the material at once. Event detection is used to determine the transition from (b) to (c). It is implemented as follows: (A) if $-(f_{\mathrm{L}x} - f_{\mathrm{L}x0})(f_{\mathrm{R}x} - f_{\mathrm{R}x0}) > 10$, or (B) if $|\tau_{\mathrm{L}y}| + |\tau_{\mathrm{R}y}| > 4$, where $f_{\mathrm{L}x}$ and $f_{\mathrm{R}x}$ indicate the x-value of the average force of the left and the right sensors, $f_{\mathrm{L}x0}$ and $f_{\mathrm{R}x0}$ indicate their initial values (right before cutting), and $\tau_{\mathrm{L}y}$ and $\tau_{\mathrm{R}y}$ indicate the y-value of the average torque of the left and the right sensors. The condition (A) is defined to detect a large force. The threshold is decided from a preliminary experiment. The condition (B) is introduced to avoid rotational slip of the knife. Figure 8(A) shows the state machine of the ⟨Automatic Cutting⟩ motion.

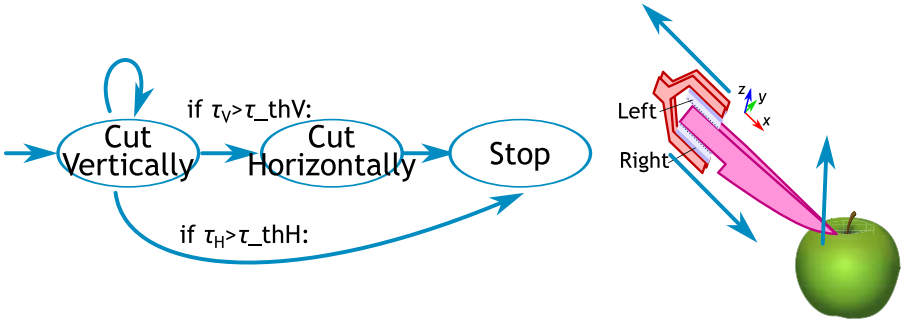
When too large force is applied to the knife, it may deform the grasp by moving the fingers (cf. Figs. 8(B)(a)) or the knife may slip in the fingers (cf. Fig. 8(B)(b)). We emphasize that FingerVision can be used to detect these situations. For example, look at the camera view of Fig. 8(B)(b); we can find that the angle of the knife is different from its initial grasp position.

4.7. ⟨*In-hand Manipulation*⟩

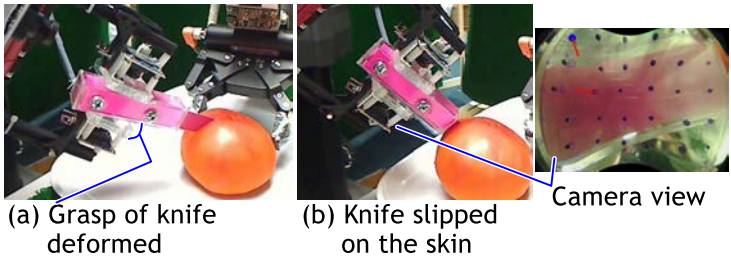
We assume that the gripper already grasps an object. The robot repeats the following process until the target angle is achieved. The robot slightly opens the gripper until if senses a small slip. Since there is a small delay between the gripper motion and slippage, we insert a short waiting time (0.1 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.

4.8. ⟨*Tracking Object*⟩

The goal of this behavior is centering an object between the robot fingers. We control the robot arm to achieve this purpose. We use object detection and pose estimation.



(A) State machine of the ⟨Automatic Cutting⟩ motion.



(B) Difficulties in cutting.

Fig. 8. ⟨Automatic Cutting⟩.

The position in the camera image plane is estimated; we control the robot arm to center the object on the image. For controlling the height of the object from the camera, we use the area of the object on the image. From two FingerVision sensors on two fingers, we obtain two estimates of object areas on the images. By controlling the robot to equalize the areas, the object locates at the center of the fingers.

This strategy is a demonstration of proximity vision of FingerVision; the robot responds to an object that is not in contact with the tactile sensors. Such a function is only possible with transparent skin. This control will be useful in centering an object before grasping it. Another application would be inspecting fruits before picking them.

4.9. ⟨*Tracking Force*⟩

The goal of this behavior is operating the robot by pushing with a small force. We use the force estimate and control the robot to move in the pushed direction. We also use object detection as a trigger to activate the control, which increases safety since the robot does not move when no object is between the fingers. We compared two variations: one uses the force estimate of Baxter (estimation from joint torque sensors), and the other combines the force estimate of Baxter and FingerVision. In the latter case, the robot was operated with smaller force. This control is a demonstration of using FingerVision in a physical human-robot interaction (HRI) scenario.

5. Experiments

We conduct some experiments to demonstrate how the proposed tactile behaviors work with FingerVision. The robots we use here are the Baxter robot of Rethink Co. and UR3 of Universal Robots. Some of the scenes are shown in Fig. 9 with the force estimation and the proximity vision views. The videos of experiments are available online:

`https://youtu.be/L-YbxcyRghQ` Force estimation, pouring water into a grasped container, test of proximity vision, ⟨Gentle Grasp⟩, ⟨Handover⟩, ⟨Holding⟩, slip-based grasping, and ⟨In-hand Manipulation⟩.
`https://youtu.be/uy32tO9e7O4` ⟨Grasp Adaptation⟩ of a flower, an origami crane, and a hairy rubber toy.
`https://youtu.be/OsAkec5bpu4` ⟨Grasp Adaptation⟩ of more than 30 kinds of objects.
`https://youtu.be/TAA4YJqEOqg` Tracking a feather with proximity vision (no touch).
`https://youtu.be/FQbNV549BQU` ⟨Tracking Force⟩ behavior (playing *Tai Chi* with the robot).
`https://youtu.be/VOrwJRv2jdk` ⟨Automatic Placing⟩ (from 0:21), and emergency stop (from 0:43).
`https://youtu.be/ifOwQdy9gDg` Early test of FingerVision, and ⟨Automatic Cutting⟩ (from 1:33).

5.1. *Robotic system*

5.1.1. *Sensor network*

The cameras of FingerVision have a USB 2.0 interface. In order to avoid long USB cables, we place local computers. The local computers send videos obtained from the FingerVision cameras to a central computer using Ethernet, and the central computer processes all the videos. In the experiments, we use Raspberry Pi 3Bs as the local computers, and transmit data through a Gigabit Ethernet network. We use MJPG-streamer[b] installed on each Raspberry Pi to capture videos from cameras, and transmit them using a motion JPEG format. In our test, the final output of FingerVision data processing (marker tracking and proximity vision) was at 63 FPS with 320×240 resolution from four cameras simultaneously. In the experiments, we reduced the FPS to 30 to reduce the computational load.

5.1.2. *FingerVision on Baxter*

The Baxter robot has two 7 degrees of freedom (DoF) arms. We use its velocity control mode commanded at 500 Hz. Our Baxter robot has two different grippers. One is the electric parallel gripper of a Baxter robot (a standard gripper) on the right hand, and the other is the Robotiq 2-finger adaptive robot gripper-85 (legacy

[b]We use a forked version: https://github.com/akihikoy/mjpg-streamer.

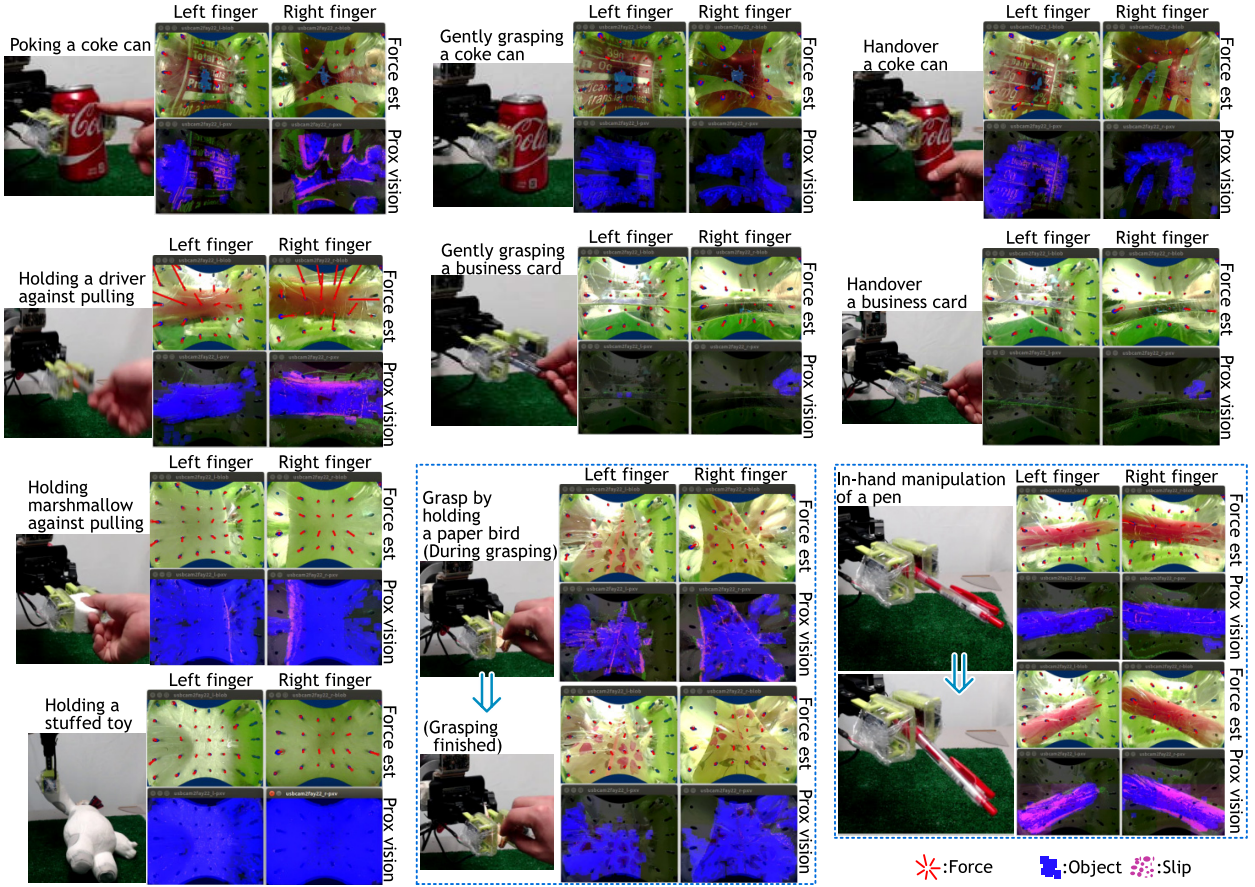
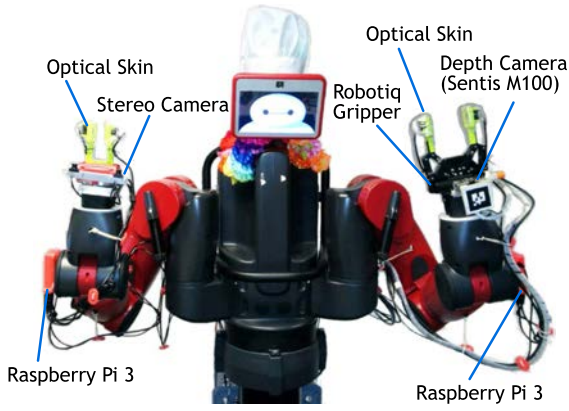
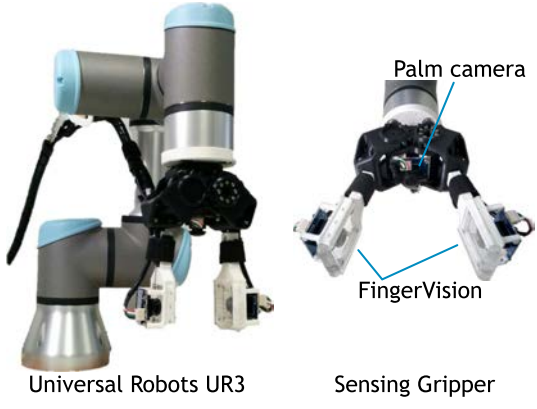


Fig. 9. Scenes of experiments. "Force est" are views of force estimation (red lines show estimated forces), and "Prox vision" are views of proximity vision (blue regions are detected objects, and purple points are detected movements).



(a) The Baxter system with four FingerVision sensors.



(b) Universal Robots UR3 with FingerVision.

Fig. 10. Robotic systems.

version) on the left hand. Two FingerVision sensors are attached on the fingers of each gripper. We use two Raspberry Pi 3B computers each of which has two FingerVision camera connections. Figure 10(a) shows the Baxter system.

5.1.3. *FingerVision on UR3*

The UR3 robot has 6 DoF and is driven by joint position or velocity commands. The robot accepts the joint velocity commands at 125 Hz. A 3D printed gripper actuated by a Dynamixel servo is mounted on the wrist of the robot that has 1 DoF. The servo is operated using the position control mode at 60 Hz, while the state is observed at 40 Hz. Two FingerVision sensors are attached on the fingers of the gripper. These devices are integrated with the control box of UR3, a Raspberry Pi 3B, and a central computer. Figure 10(b) shows the UR3 system.

5.2. *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]. We record the force in static situations. For each measurement, we wait for a few seconds for recording. Figure 11 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.

5.3. *Pouring water into a grasped container*

We have the Robotiq gripper of Baxter grasp a container, and then pour water into the container manually. Figure 12 shows the gravity-direction component (z-axis) of the force estimate. Pouring was performed from 35 [s] to 52 [s]. The force gradually

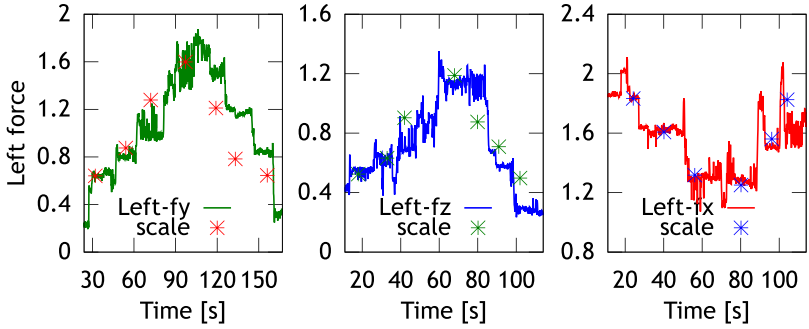


Fig. 11. Average force trajectories in evaluating f_y (left), f_z (middle), and f_x (right) respectively. The $*$ mark the scale readings (linearly scaled, and offset). The unit of force is omitted as it is not calibrated as engineered units.

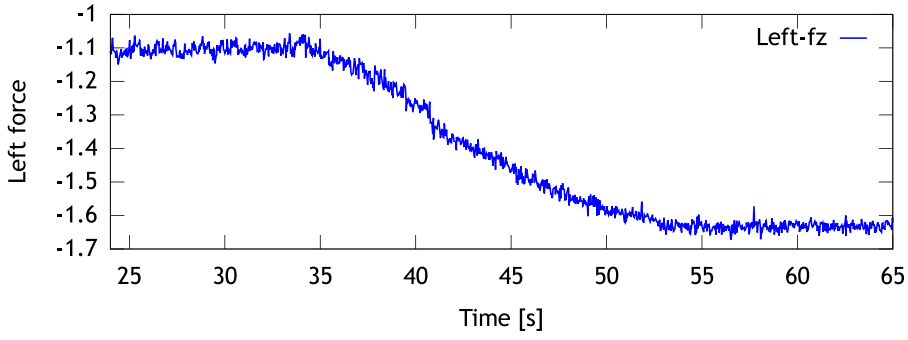


Fig. 12. Gravity-direction component (z-axis) of the force estimate during the pouring-water experiment. Actual pouring is from 35 [s] to 52 [s]. The unit of force is omitted as it is not calibrated as engineered units.

increased. This would be accurate enough to estimate the poured amount of water during a pouring task.

5.4. *Test of proximity vision*

We explore basic results from proximity vision. We let the Robotiq gripper of Baxter 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. Figure 13 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. Figure 14 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.

5.5. ⟨*Gentle Grasp*⟩

We test the ⟨Gentle Grasp⟩ strategy with the Robotiq gripper of Baxter. We have the robot grasp an empty Coke can, and grasp a paper business card on edge. Both

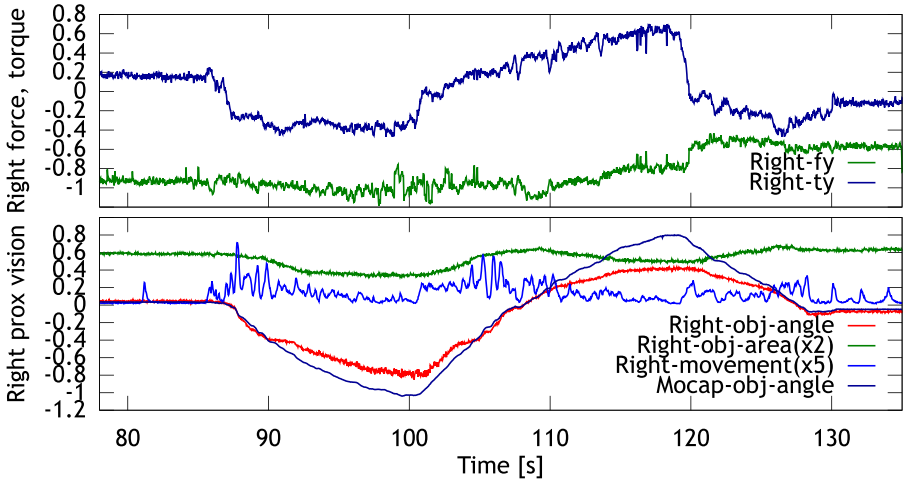


Fig. 13. Force and torque estimate (top) and proximity vision (bottom) during rotating the screw driver. In the proximity vision graph, there are plots of the object angle (radian) and area obtained from the moment of object pixels, and the total number of moving pixels (normalized by the image size). 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.

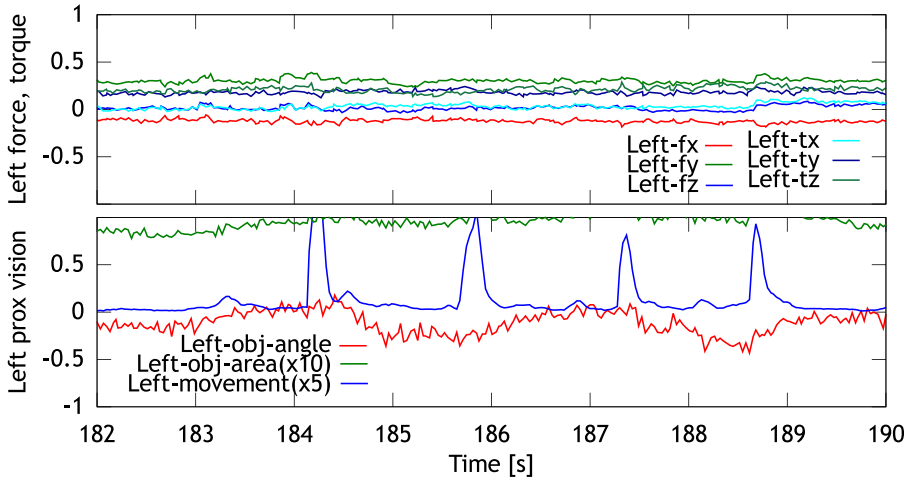


Fig. 14. Force and torque estimate (top) and the proximity vision (bottom) during poking a Coke can. See the caption of Fig. 13 for the plots.

objects are soft and will be damaged with even small forces. Figure 15 shows the force and torque estimate, and the proximity vision output during ⟨Gentle Grasp⟩. The actual grasp happened at 34.5 [s]. We can see a small 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. 16. 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.

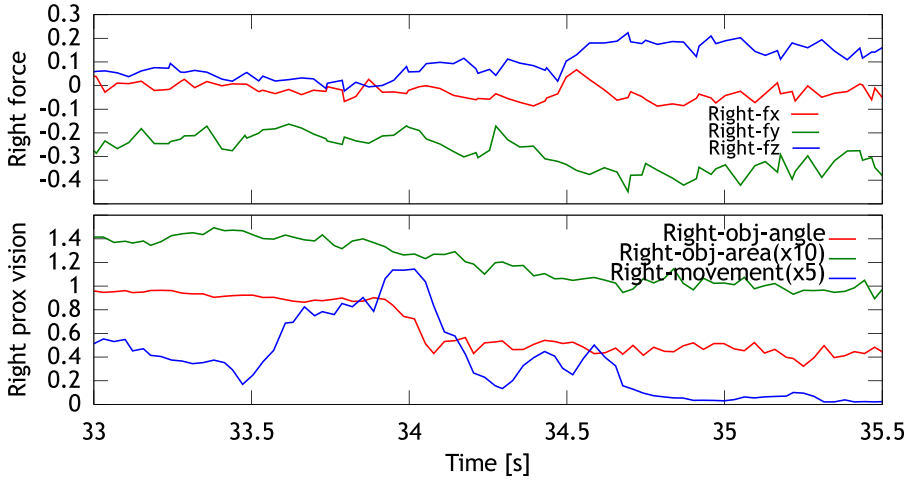


Fig. 15. Force and torque estimate (top) and the proximity vision (bottom) during gently grasping a Coke can. See the caption of Fig. 13 for the plots.

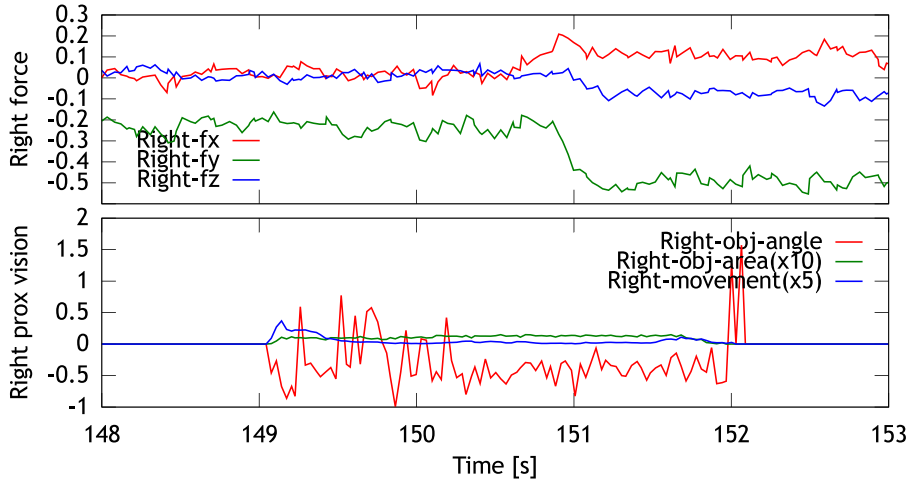


Fig. 16. Force and torque estimate (top) and the proximity vision (bottom) during gently grasping a business card. See the caption of Fig. 13 for the plots.

5.6. ⟨*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 each pattern, a human pushes the driver several times. The results are shown in Fig. 17. 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. Figure 18 shows the result. We can see many slip detections (peaks in Right-movement) from the bottom graph, and the magnitude of grasping force ($|fy|$) is increasing accordingly in the top graph.

Next we let the robot move a stuffed toy. Moving with the ⟨Gentle Grasp⟩ strategy, the robot dropped the toy due to a slip. However by activating the ⟨Holding⟩ strategy, the robot could hold and move the toy. Figure 19 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.

5.7. *Grasping a fragile object with the* ⟨*Holding*⟩ *strategy*

We verify our concept that by 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 Baxter robot uses the ⟨Holding⟩

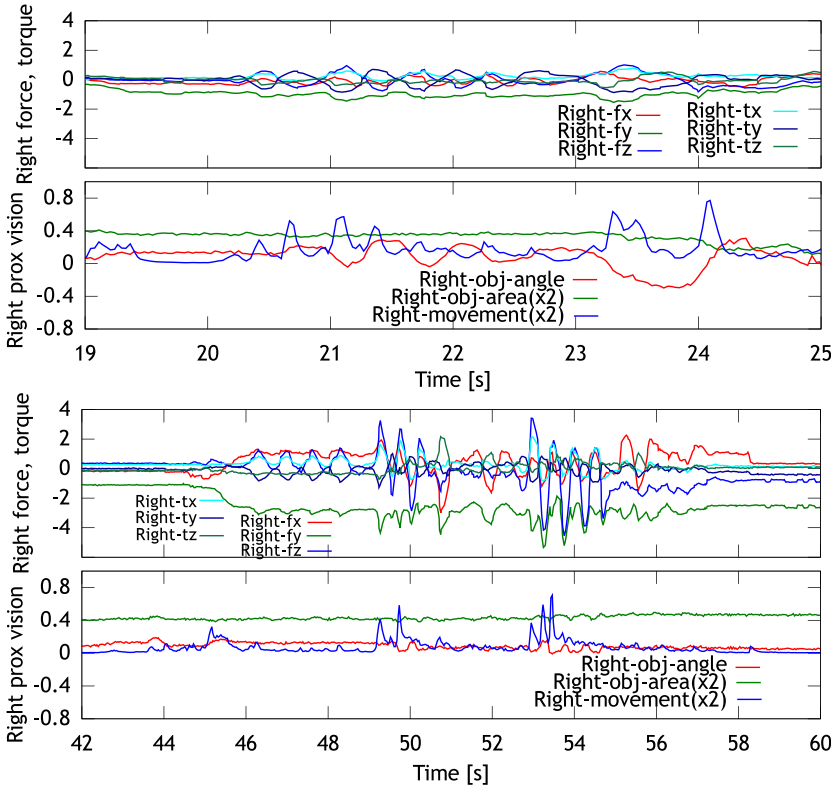


Fig. 17. Results of the ⟨Holding⟩ strategy. Top two graphs are results of ⟨Gentle Grasp⟩, and bottom two graphs are ones of the ⟨Holding⟩ strategy. In each pair, force and torque estimate (top) and the proximity vision (bottom) are plotted. See the caption of Fig. 13 for the plots.

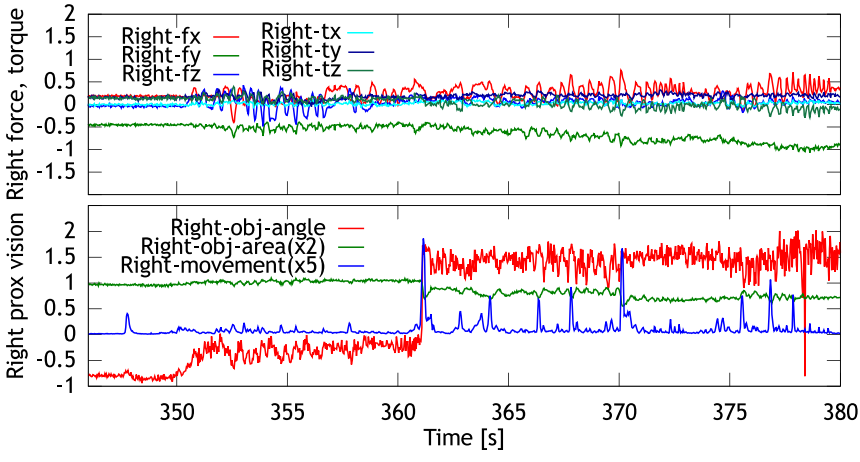


Fig. 18. Force and torque estimate (top) and the proximity vision (bottom) during holding and moving a marshmallow. See the caption of Fig. 13 for the plots.

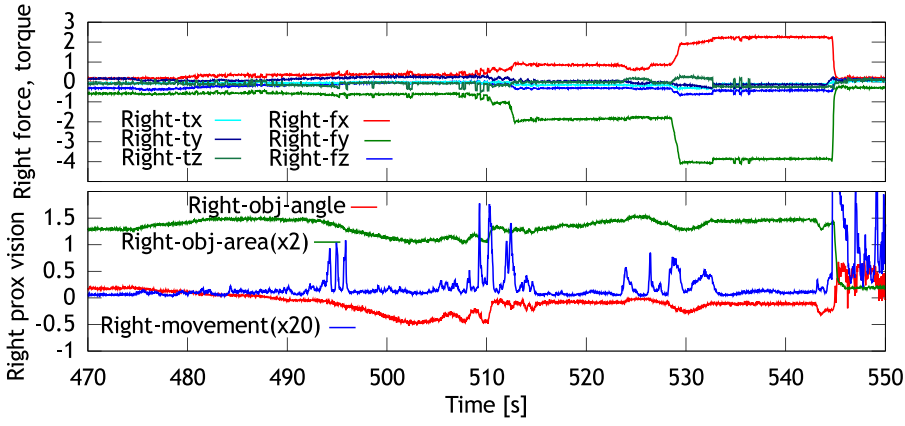


Fig. 19. Force and torque estimate (top) and the proximity vision (bottom) during holding and moving a stuffed toy. See the caption of Fig. 13 for the plots.

strategy. After grasping it without slip, the robot swings its arm to see if the ⟨Holding⟩ strategy is effective. Figure 20 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).

5.8. ⟨*Grasp Adaptation*⟩

The previous experiment showed our concept works. Next, we conduct a further test of the ⟨Grasp Adaptation⟩ controller. We verify that when an adequate grasp pose

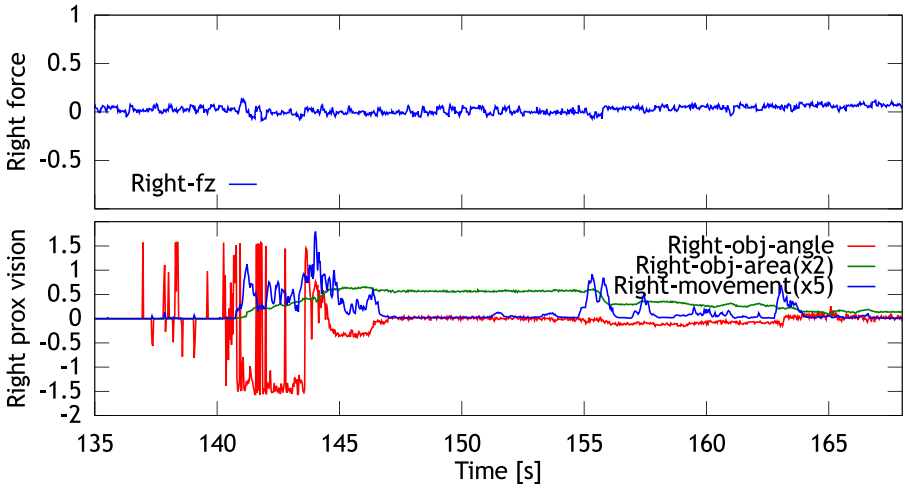


Fig. 20. Force estimate (top) and the proximity vision (bottom) during grasping a paper bird. See the caption of Fig. 13 for the plots.

for an object is given, the ⟨Grasp Adaptation⟩ controller can adapt the grasp to the object robustly regardless of the object and its properties. We let a human operator decide a good grasp pose for a given object with a joystick controller, and then run the ⟨Grasp Adaptation⟩ controller to pick up the object. We do not tune the parameters of the controller for each object.

We tested with 30 deformable and fragile objects shown in Fig. 21 including vegetables, fruits, origami objects, and a raw egg. We use the Robotiq gripper of Baxter. Initially each object is placed on a table.

We conducted 36 trials: Origami box, Origami crane, Badminton ball, Hairy rubber toy, Cup cake, Chocolate, Strawberry, Tomato-medium-1, Eggplant@1, Eggplant@2, Zucchini-yellow, Mushroom-1@1, Mushroom-1@2, Egg(raw), Pepper-red-1, Oyster mushroom-1, Peach-1, Mushroom-2, Potato-1, Kiwi-1, Tomato-medium-2, Broccoli@1, Broccoli@2, Oyster mushroom-2, Green pepper-1, Kiwi-2, Pepper-red-2, Tomato-big, Banana-1@1, Banana-1@2, Banana-1@3, Green pepper-2@1, Green pepper-2@2, Peach-2, Potato-2, Banana-2. A label with @N denotes an

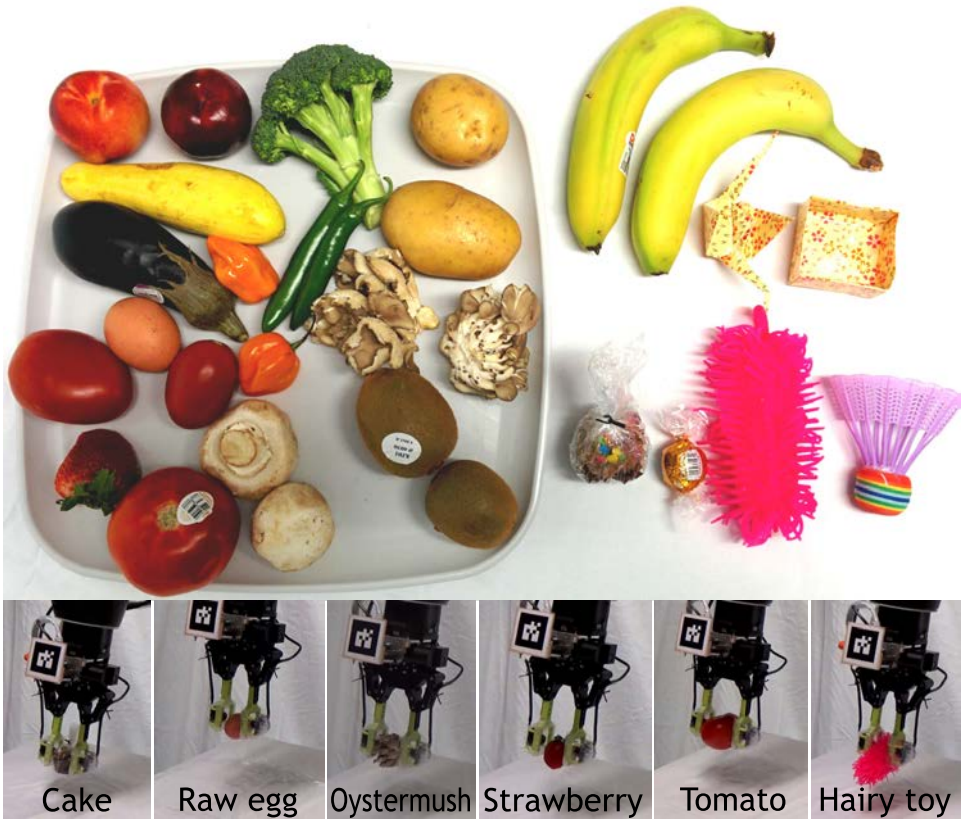


Fig. 21. Top: 30 objects used in the experiment. Bottom: Examples of grasp (cup cake, raw egg, oyster mushroom, strawberry, tomato, hairy rubber toy).

Nth trial of the same object. Examples of successful grasping are shown in Fig. 21. There were several failures: (1) Dropped after bringing up: Oyster mushroom-1, Potato-1. (2) Slippage could not be detected due to a computer vision failure: Eggplant@1 (the skin was black), Broccoli@1 and Green pepper-2@1 (the color was similar to the fingers). (3) Closing gripper did not stop in Banana-1@1 because detecting the deformation of object as slip. Since the contact force from the table disappeared when bringing up the banana, the banana skin was deformed slightly. (4) In Banana-1@2, dropped during bringing up, and failed to re-grasp since the fingers got stuck at the edge of the object, and the passive joints of the gripper bent. Note that (2) was solved by grasping the green part (Eggplant@2), helping the object detection manually (Broccoli@2), and just trying again (Green pepper-2@2).

The issues of (2) and (3) will be solved by improving the computer vision method for: (A) a better object detection and (B) distinguishing slippage and deformation. The issues of (1) and (4) will be solved by improving the behavior. For example, testing the grasp stability by shaking the object after grasping will avoid (1). (4) can be solved by optimizing the trajectory of fingertip in re-grasping.

5.9. ⟨*Handover*⟩

We demonstrate the ⟨Handover⟩ strategy by applying it to a Coke can and a business card. Both objects are grasped by the ⟨Gentle Grasp⟩ strategy, and the card is grasped on edge. Figure 22 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.

5.10. ⟨*Automatic Placing*⟩, ⟨*Tracking Object*⟩, *and* ⟨*Tracking Force*⟩

We demonstrate the motions of the ⟨Automatic Placing⟩ with the UR3. The motion starts where the robot grasps an origami box. The grasp pose is displaced toward the fingertip on purpose. In the placing motion, the robot tries to move the gripper downward until its fingertip is around the table. With this setup, the grasped object will touch the table before the planned motion ends.

Figure 23 shows an example of the execution. The left image is the initial pose where we can see the displaced grasp of the object. At the fourth image, the object touched the table. Since slip was detected, the robot stopped the placing motion and opened the gripper. The views of FingerVision at the beginning and at slip detection are shown in the same figure. We can see the points in slip at the frame when slip was detected.

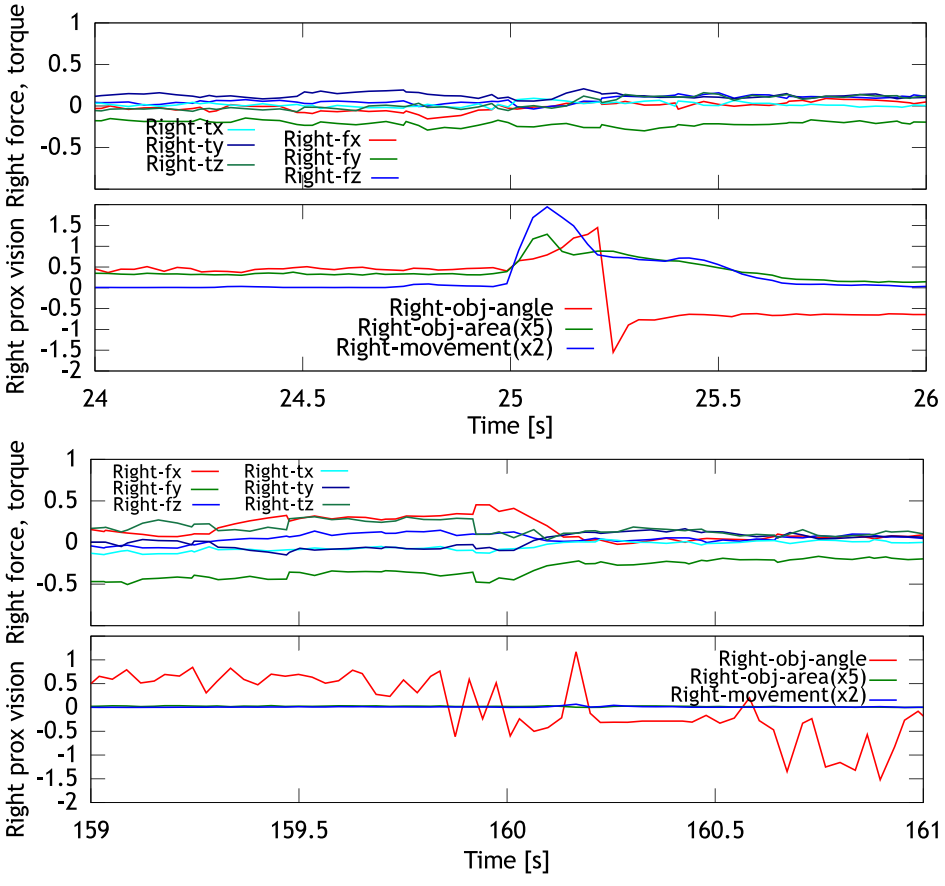


Fig. 22. Results of the ⟨Handover⟩ strategy. Top two graphs show the result of the Coke can case, and bottom tow graphs show the result of the card case. Each of them have force and torque estimates (top) and the proximity vision (bottom). See the caption of Fig. 13 for the plots.

5.11. ⟨*In-hand Manipulation*⟩

We apply the ⟨In-hand Manipulation⟩ strategy to rotating a pen. The target angle is 20 degrees from the current angle. Figure 24 shows the results of eight runs. In most cases, the 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.

5.12. ⟨*Automatic Cutting*⟩

We conduct an experiment of the ⟨Automatic Cutting⟩ with the electric parallel gripper of Baxter. The target objects are a banana and an apple. Figure 25 shows a scene of cutting the apple and the corresponding marker tracking result. Figure 26

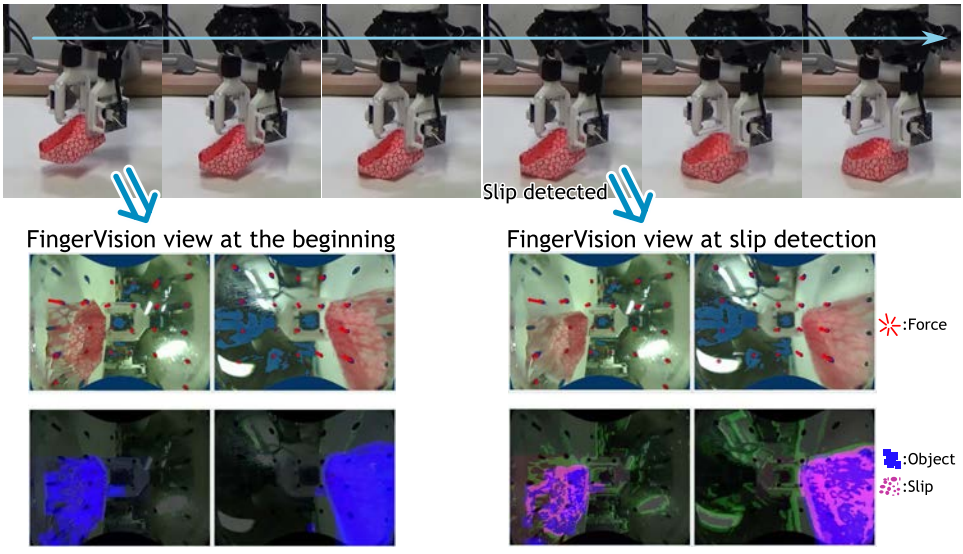


Fig. 23. An execution scene of placing origami box.

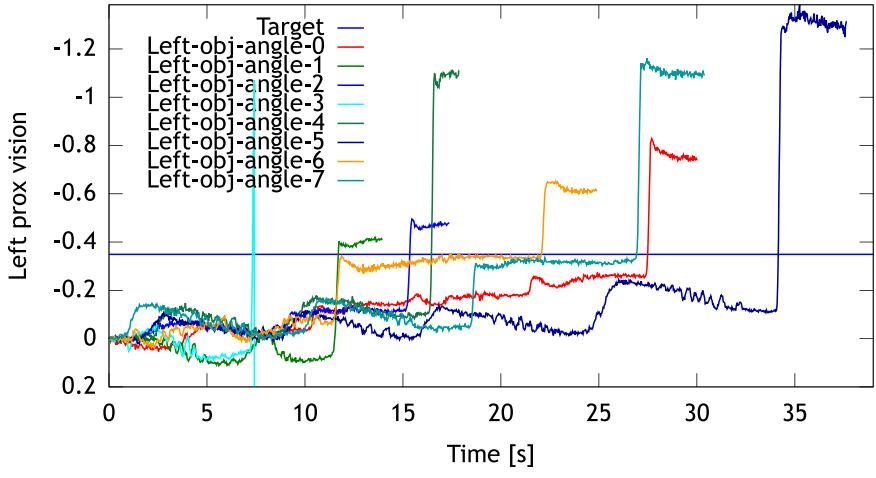


Fig. 24. Result of ⟨In-hand Manipulation⟩. The object orientations obtained by the proximity vision are plotted per time. The initial orientation is set to be zero.

and Fig. 27 shows the sensor values during cutting a banana and an apple, respectively, where the trajectories of the average force (x, z) and torque (y) of the left and the right FingerVision sensors are shown. Since cutting a banana requires only a small force, the robot cut it with a single trial, while the robot took four trials in cutting an apple. In the banana case, the robot stopped moving the knife because

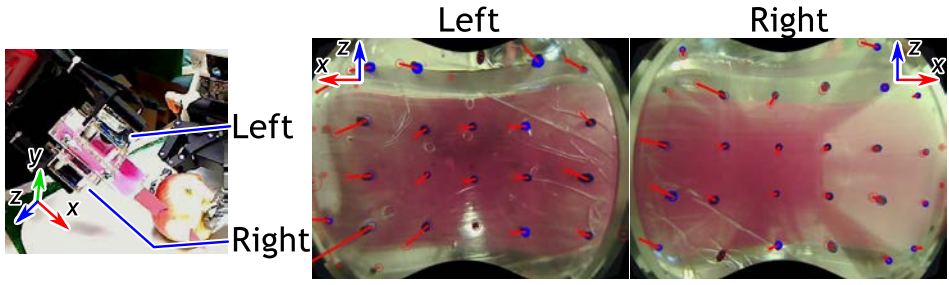


Fig. 25. Marker tracking result in cutting an apple by the robot. The left image is a view from an external camera.

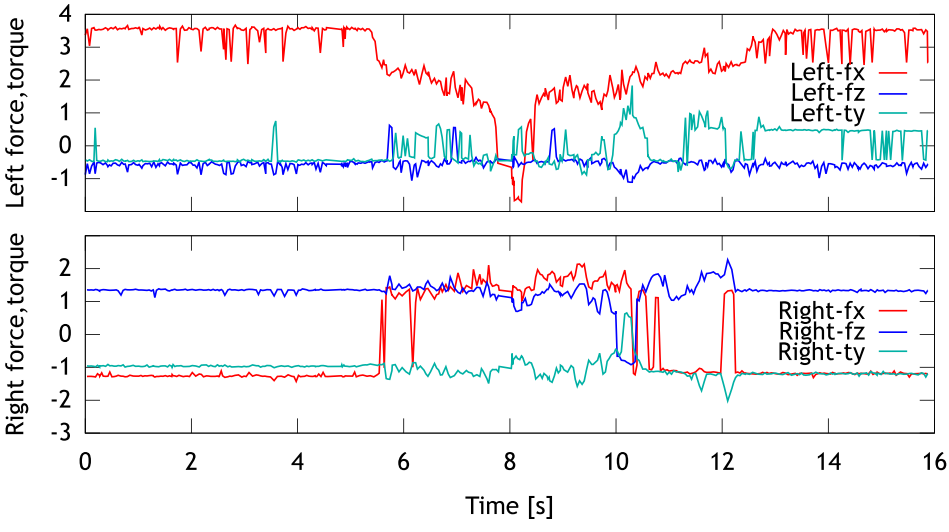


Fig. 26. Trajectories of the average force (x, z) and torque (y) of the left and the right FingerVision sensors during cutting a banana by the robot. There was a single cutting motion around the peak of left x-force.

the condition (A) of the ⟨Automatic Cutting⟩ motion was satisfied when the knife hit the cutting board. In the apple case, the condition (A) was mainly satisfied by the pressure from the cut edge of the apple fresh. In these graphs, the condition (B) was not satisfied, but it was useful when the initial knife orientation was not perpendicular.

5.13. ⟨*Tracking Object*⟩ *and* ⟨*Tracking Force*⟩

We demonstrate the motions of ⟨Tracking Object⟩ and ⟨Tracking Force⟩. Figure 28 shows the snapshots of 30 s of ⟨Tracking Object⟩ where the object is a pink feather

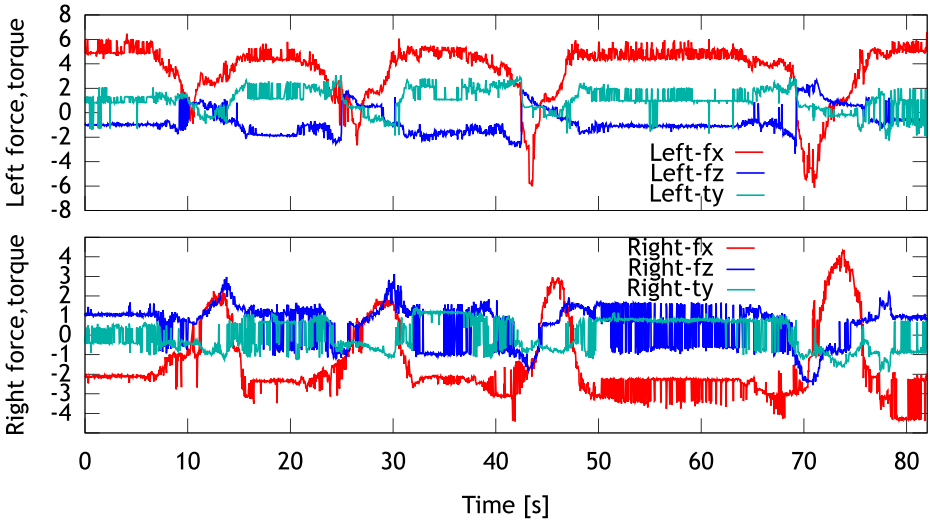


Fig. 27. Trajectories of the average force (x, z) and torque (y) of the left and the right FingerVision sensors during cutting an apple by the robot. The robot performed the cutting motion four times to cut the apple completely. The peaks of left and right x-force correspond with the cutting motion.

performed by Baxter. We can see that the feather is detected by the object detection algorithm in the FingerVision views. The robot tries to center the object on image by moving its arm.

Figure 29 shows the snapshots of 6 s of the ⟨Tracking Force⟩ behavior with the UR3. The human operator pushed the finger with a small force, and then the robot moved toward the pushed direction. Although this is a simple demonstration, it shows the capability of FingerVision in HRI applications.

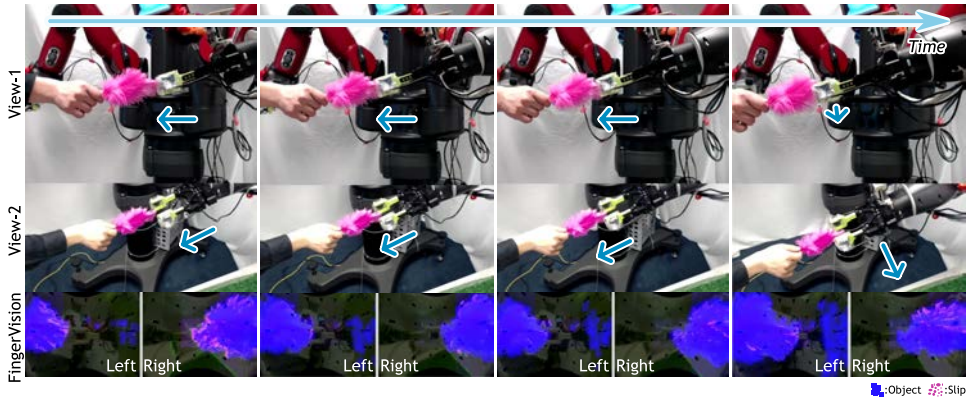


Fig. 28. Snapshots of ⟨Tracking Object⟩.

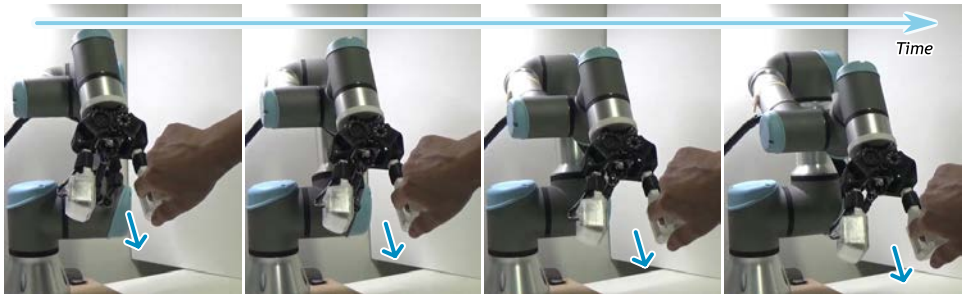


Fig. 29. Snapshots of ⟨Tracking Force⟩.

6. Discussion

6.1. *Why are tactile sensors not as popular as vision sensors?*

There are popular solutions in robot vision (RGB cameras and OpenCV; depth cameras and the Point Cloud Library), while there is no *de facto* standard solutions in tactile sensing for robots. The reasons vary: difficulty to physically install, low durability, poor compatibility among different tactile sensors, unclear usage, maintenance complexity, programming complexity, and expense. As a result, accumulating the knowledge of tactile sensing for robots is more difficult than that of robot vision.

6.2. *Improving software improves FingerVision*

This is an interesting feature of vision-based tactile sensors. Typically, we need to improve hardware to improve a sensor, while in vision-based tactile sensors, we can improve the sensing capability by improving the computer vision software. Of course high-level sensing capabilities can be improved by software with the other types of sensors, but vision-based tactile sensors have potential to increase low-level sensing capabilities. For example, one important part of computer vision for FingerVision is object detection and tracking. We found that there were some failures in the ⟨Grasp Adaptation⟩ experiments. These failures could be solved by improving the computer vision. It also should be mentioned that we can use the recent success of deep learning in computer vision with vision-based tactile sensors. Especially, with FingerVision, we can use RGB image processing with FingerVision such as object recognition, which would be an advantage of FingerVision.

6.3. *How does FingerVision improve manipulation?*

Through the implementation of tactile behaviors, we found there are some useful cases of FingerVision in manipulation. Slip detection is useful in many scenarios to avoid slip, ⟨Grasp Adaptation⟩, and ⟨In-hand Manipulation⟩ where the slip is created on purpose. It is also used as triggers, for example in ⟨Handover⟩ and placing

behaviors. Force estimation is used as triggers in ⟨Handover⟩ and ⟨Automatic Cutting⟩ behaviors. It is useful to detect an emergency (collision), which was used in the ⟨Grasp Adaptation⟩.

6.4. *Should we convert the force estimate to an engineering unit (e.g., Newtons)?*

This depends on the application. We are planning to use machine learning methods to learn dynamical models (e.g., relation between input gripper motion and output force changes) for example by using neural networks.[65] In this case, obtaining contact force information in engineering units is not necessary. If the force estimates are consistent enough, we could learn mappings from grasping-action parameters to a force distribution, and from a force distribution to grasping-quality measurements. These models would be useful to reason about grasping actions. However, using an engineering unit will be generalizable to other situations and other robots, so it is still beneficial to consider. Accurately estimating force will depend on the contact location and may require finite element modeling of the sensor structure.

6.5. *Accuracy, reliability, and hysteresis*

The marker movement in a horizontal direction is easier to track. The vertical force is more difficult to detect as the changes of the marker visual size are comparably smaller. A way to improve the vertical force sensitivity is increasing the thickness of the elastic layer, although it would make the skin heavier.

In the experiments, there were some false detections of the markers. These were mostly due to the external scene. Increasing the number of markers is helpful for removing outliers, although it will reduce the transparency accordingly.

We found hysteresis, especially when a strong force was applied. In the cutting vegetable experiments by the robot, the force estimate changed before and after cutting a hard material. This would be because the deformation of the soft layer remained. This was mostly reset after releasing the knife.

Although we demonstrated a basic study of force estimation, we did not identify the range of detectable force and the accuracy of force estimation in dynamic situations. Also, we did not evaluate the detection of slip. These are left for future work.

6.6. *Can FingerVision grasp objects in the dark or black/transparent objects?*

Installing structured light sources is helpful in dark scenes, although we would need to avoid reflections of light at the surface of the silicone skin and the boundary of the silicone and acrylic layers.

In case an object is black (the same color as the markers), it will become difficult to detect the markers. However, using multiple colored markers will make the marker

detection more robust against the object colors. Note that proximity vision still works with black objects.

On the other hand, proximity vision will not work with completely transparent objects. However, marker tracking may detect the reaction force from the objects.

Recent progress of computer vision also helps FingerVision to solve these issues.

6.7. *Can we distinguish slip and deformation?*

Since both slip and deformation appear as movement of pixels in the image, it is not straightforward to distinguish them. Our current implementation of slip detection detects both as slip. From our experience, we found that the patterns of the movement of pixels are different between slip and deformation. In the case of slip, the pixels move in the same direction, while in the case of deformation, the pixels move inward radially.

6.8. *Proximity vision versus functional membranes*

The key aspect of FingerVision is the transparent skin, while other vision-based tactile sensors use opaque skin. FingerVision also has markers to detect force distribution. A drawback of FingerVision would be the robustness to external light and object view. As discussed above, we think this drawback can be handled.

Some vision-based tactile sensors use functional membranes such as GelSight.[27] While GelSight provides high-resolution and sensitive texture and shape detection, FingerVision provides proximity vision and sensitive slip detection. FingerVision can also combine with other sensors such as depth and thermal cameras. Which is better? It would be difficult to answer now since it depends on the task.

7. Conclusion

We introduced a vision-based tactile sensor FingerVision and explored tactile behaviors implemented on a Baxter and a Universal Robots UR3 to show its usefulness. FingerVision consists of a transparent elastic material with black dots, and a camera, that is easy to fabricate, low cost, and physically robust. Unlike other vision-based tactile sensors, the complete transparency of the FingerVision skin provides a multimodal sensation. The modalities sensed by FingerVision include distributions of force and slip, and object information such as distance, location, pose, size, shape, and texture. The slip detection is very sensitive since it is obtained by computer vision directly applied to the output of the FingerVision camera. It provides high-resolution slip detection, which does not depend on the contact force, i.e., it can sense slip of a lightweight object that generates negligible contact force. The tactile behaviors explored in this paper include manipulations that emphasize this feature. For example, we demonstrated that grasp adaptation with FingerVision could

grasp origami, and other deformable and fragile objects such as vegetables, fruits, and raw eggs.

Acknowledgments

Yamaguchi was supported in part by NSK Foundation for the Advancement of Mechatronics, and JST START grant JPMJST1815. Atkeson was supported by award IIS-1717066 from the US National Science.

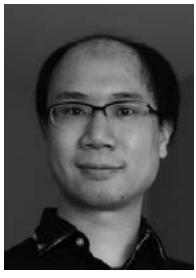
References

1. A. Yamaguchi and C. G. Atkeson, Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables, in *16th IEEE-RAS Int. Conf. Humanoid Robots (Humanoids'16)*, 2016.
2. N. F. Lepora and B. Ward-Cherrier, Superresolution with an optical tactile sensor, in *2015 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)* (2015), pp. 2686–2691.
3. R. Li, R. Platt, W. Yuan, A. Ten Pas, N. Roscup, M. A. Srinivasan and E. Adelson, Localization and manipulation of small parts using GelSight tactile sensing, in *2014 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014, pp. 3988–3993.
4. A. Yamaguchi, FingerVision, http://akihikoy.net/p/fv.html (2017).
5. A. Yamaguchi and C. G. Atkeson, Implementing tactile behaviors using FingerVision, in *17th IEEE-RAS Int. Conf. Humanoid Robots (Humanoids'17)* (2017).
6. A. Yamaguchi, FingerVision for tactile behaviors, manipulation, and haptic feedback teleoperation, in *4th IEEJ Int. Workshop on Sensing, Actuation, Motion Control, and Optimization (SAMCON2018)*, 2018.
7. A. Yamaguchi and C. G. Atkeson, Grasp adaptation control with finger vision: Verification with deformable and fragile objects, in *35th Annual Conf. Robotics Society of Japan (RSJ2017)*, 2017, pp. 1L3–01.
8. Barrett Technology, About BarrettHand, https://www.barrett.com/about-barrethand/ (2018) [accessed on 26 Oct 2018].
9. Willow Garage, PR2, http://www.willowgarage.com/pages/pr2/overview (2015) [accessed on 26 Oct 2018].
10. RightHand Robotics, "ReFlex hand," https://www.labs.righthandrobotics.com/reflex-hand (2018) [accessed on 26 Oct 2018].
11. U. Martinez-Hernandez, Tactile sensors, *Scholarpedia* **10**(4) (2015) 32398.
12. Z. Kappassov, J.-A. Corrales and V. Perdereau, Tactile sensing in dexterous robot hands review, *Robot. Auton. Syst.* **74** (2015) 195–220.
13. S. Begej, Planar and finger-shaped optical tactile sensors for robotic applications, *IEEE J. Robot. Autom.* **4**(5) (1988) 472–484.
14. H. Maekawa, K. Tanie, K. Komoriya, M. Kaneko, C. Horiguchi, and T. Sugawara, Development of a finger-shaped tactile sensor and its evaluation by active touch, in *Proc. 1992 IEEE Int. Conf. Robotics and Automation, 1992*, Vol. 2, 1992, pp. 1327–1334.
15. H. Yussof, J. Wada and M. Ohka, Sensorization of robotic hand using optical three-axis tactile sensor: Evaluation with grasping and twisting motions, *J. Comput. Sci.* **6**(8) (2010) 955–962.
16. T. Ikai, S. Kamiya and M. Ohka, Robot control using natural instructions via visual and tactile sensations, *J. Comput. Sci.* **12**(5) (2016) 246–254.

17. K. Nagata, M. Ooki and M. Kakikura, Feature detection with an image based compliant tactile sensor, in *1999 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1999, pp. 838–843.
18. K. Kamiyama, H. Kajimoto, N. Kawakami and S. Tachi, Evaluation of a vision-based tactile sensor, in *2004 IEEE Int. Conf. Robotics and Automation, 2004, Proc. ICRA '04*, Vol. 2, 2004, pp. 1542–1547.
19. J. Ueda, Y. Ishida, M. Kondo, and T. Ogasawara, Development of the NAIST-Hand with vision-based tactile fingertip sensor, in *Proc. 2005 IEEE Int. Conf. Robotics and Automation*, 2005, pp. 2332–2337.
20. C. Chorley, C. Melhuish, T. Pipe and J. Rossiter, Development of a tactile sensor based on biologically inspired edge encoding, in *Int. Conf. Advanced Robotics, 2009, ICAR 2009*, 2009, pp. 1–6.
21. D. Hristu, N. Ferrier and R. W. Brockett, The performance of a deformable-membrane tactile sensor: Basic results on geometrically-defined tasks, in *IEEE Int. Conf. Robotics and Automation, 2000, Proc. ICRA '00*, Vol. 1, 2000, pp. 508–513.
22. Y. Ito, Y. Kim and G. Obinata, Robust slippage degree estimation based on reference update of vision-based tactile sensor, *IEEE Sensors J.* **11**(9) (2011) 2037–2047.
23. T. Assaf, C. Roke, J. Rossiter, T. Pipe and C. Melhuish, Seeing by touch: Evaluation of a soft biologically-inspired artificial fingertip in real-time active touch, *Sensors* **14**(2) (2014) 2561.
24. T. Sakuma, F. Von Drigalski, M. Ding, J. Takamatsu, and T. Ogasawara, A universal gripper using optical sensing to acquire tactile information and membrane deformation, in *2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.
25. B. W. McInroe, C. L. Chen, K. Y. Goldberg, R. Bajcsy and R. S. Fearing, Towards a soft fingertip with integrated sensing and actuation, in *2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018, pp. 6437–6444.
26. OptoForce Co., White paper: Optical force sensors — introduction to the technology, Tech. Rep., January 2015. Available: http://optoforce.com/.
27. M. K. Johnson and E. H. Adelson, Retrographic sensing for the measurement of surface texture and shape, in *2009 IEEE Conf. Computer Vision and Pattern Recognition*, 2009, pp. 1070–1077.
28. W. Yuan, R. Li, M. A. Srinivasan, and E. H. Adelson, Measurement of shear and slip with a GelSight tactile sensor, in *2015 IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 304–311.
29. E. Donlon, S. Dong, M. Liu, J. Li, E. Adelson and A. Rodriguez, Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger, in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018.
30. R. Patel and N. Correll, Integrated force and distance sensing using elastomer-embedded commodity proximity sensors, in *Proc. Robotics: Science and Systems*, 2016.
31. M. Ueda, K. Iwata and H. Shingu, Tactile sensors for an industrial robot to detect a slip, in *2nd Int. Symp. Industrial Robots*, 1972, pp. 63–76.
32. D. Dornfeld and C. Handy, Slip detection using acoustic emission signal analysis, in *Proc. 1987 IEEE Int. Conf. Robotics and Automation*, Vol. 4, 1987, pp. 1868–1875.
33. M. R. Tremblay and M. R. Cutkosky, Estimating friction using incipient slip sensing during a manipulation task, in *1993 Proc. IEEE Int. Conf. Robotics and Automation*, Vol. 1, 1993, pp. 429–434.
34. R. D. Howe and M. R. Cutkosky, Sensing skin acceleration for slip and texture perception, in *Proc. 1989 Int. Conf. Robotics and Automation*, Vol. 1, 1989, pp. 145–150.
35. R. Fernandez, I. Payo, A. S. Vazquez and J. Becedas, Micro-vibration-based slip detection in tactile force sensors, *Sensors* **14**(1) (2014) 709–730.

36. R. Fernandez, I. Payo, A. S. Vazquez, and J. Becedas, *Slip Detection in Robotic Hands with Flexible Parts* (Springer International Publishing, Cham, 2014), pp. 153–167.
37. 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 Conf. Ser. Mater. Sci. Eng.* 99(1) (2015) 012001.
38. R. Fernandez, I. Payo, A. S. Vazquez and J. Becedas, *Slip Detection in a Novel Tactile Force Sensor* (Springer International Publishing, Cham, 2016), pp. 237–252.
39. C. Melchiorri, Slip detection and control using tactile and force sensors, *IEEE/ASME Trans. Mechatron.* **5**(3) (2000) 235–243.
40. J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta and K. J. Kuchenbecker, Human-inspired robotic grasp control with tactile sensing, *IEEE Trans. Robotics* **27**(6) (2011) 1067–1079.
41. 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 *Proc. IEEE Int. Conf. Robot. Automa.* Vol. 4, 1996, pp. 3234–3239.
42. 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 Int. Conf. Intelligent Robots and Systems (IROS)*, Vol. 1, 2004, pp. 331–336.
43. 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 Int. Conf. Automation Science and Engineering*, 2012, pp. 461–468.
44. T. Hasegawa and K. Honda, Detection and measurement of fingertip slip in multi-fingered precision manipulation with rolling contact, in *Int. Conf. Multisensor Fusion and Integration for Intelligent Systems (MFI 2001)*, 2001, pp. 43–48.
45. 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 Int. Conf. Robotics and Automation*, 2008, pp. 2605–2610.
46. 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 Int. Conf. Robotics and Automation (ICRA)*, 2014, pp. 2742–2748.
47. R. Johansson, G. Loeb, N. Wettels, D. Popovic and V. Santos, Biomimetic tactile sensor for control of grip, (2011), US Patent 7,878,075.
48. 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 Int. Conf. Humanoid Robots (Humanoids)*, 2015, pp. 297–303.
49. 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 Int. Conf. Intelligent Robots and Systems (IROS)*, Vol. 1, 2004, pp. 810–815.
50. P. K. Allen, A. T. Miller, P. Y. Oh and B. S. Leibowitz, Integration of vision, force and tactile sensing for grasping, *Int. J. Intelligent Machines* **4** (1999) 129–149.
51. K. Hsiao, S. Chitta, M. Ciocarlie and E. G. Jones, Contact-reactive grasping of objects with partial shape information, in *2010 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2010, pp. 1228–1235.
52. R. S. Johansson and J. R. Flanagan, Coding and use of tactile signals from the fingertips in object manipulation tasks, *Nat. Rev. Neurosci.* **10**(345) (2009).
53. R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson and S. Levine, More than a feeling: Learning to grasp and regrasp using vision and touch, *IEEE Robot. Autom. Lett.* **3**(4) (2018) 3300–3307.

54. F. Hogan, M. Bauza Villalonga, O. Canal Anton, E. Donlon and A. Rodriguez, Tactile regrasp: Grasp adjustments via simulated tactile transformations, in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018.
55. H. Dang and P. K. Allen, Stable grasping under pose uncertainty using tactile feedback, *Autonomous Robots* **36**(4) (2014) 309–330.
56. M. Li, Y. Bekiroglu, D. Kragic and A. Billard, Learning of grasp adaptation through experience and tactile sensing, in *2014 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014, pp. 3339–3346.
57. E. Hyttinen, D. Kragic and R. Detry, Estimating tactile data for adaptive grasping of novel objects, in *2017 IEEE-RAS 17th Int. Conf. Humanoid Robotics (Humanoids)*, 2017, pp. 643–648.
58. C. Chuang, M. Wang, Y. Yu, C. Mu, K. Lu and C. Lin, Flexible tactile sensor for the grasping control of robot fingers, in *2013 Int. Conf. Advanced Robotics and Intelligent Systems*, 2013, pp. 141–146.
59. F. Veiga, H. van Hoof, J. Peters and T. Hermans, Stabilizing novel objects by learning to predict tactile slip, in *2015 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2015, pp. 5065–5072.
60. M. Kaboli, K. Yao and G. Cheng, Tactile-based manipulation of deformable objects with dynamic center of mass, in *2016 IEEE-RAS 16th Int. Conf. Humanoid Robots (Humanoids)*, 2016, pp. 752–757.
61. J. Venter and A. M. Mazid, Tactile sensor based intelligent grasping system, in *2017 IEEE Int. Conf. Mechatronics (ICM)*, 2017, pp. 303–308.
62. L. Cramphorn, B. Ward-Cherrier and N. F. Lepora, Tactile manipulation with biomimetic active touch, in *2016 IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 123–129.
63. H. van Hoof, T. Hermans, G. Neumann and J. Peters, Learning robot in-hand manipulation with tactile features, in *2015 IEEE-RAS 15th Int. Conf. Humanoid Robots (Humanoids)*, 2015, pp. 121–127.
64. R. B. Hellman, C. Tekin, M. van der Schaar and V. J. Santos, Functional contour-following via haptic perception and reinforcement learning, *IEEE Trans. Haptics* **11**(1) (2018) 61–72.
65. A. Yamaguchi and C. G. Atkeson, Neural networks and differential dynamic programming for reinforcement learning problems, in *IEEE Int. Conf. Robotics and Automation (ICRA'16)*, 2016.



Akihiko Yamaguchi received the BE degree from the Kyoto University, Kyoto, Japan, in 2006, and the ME and the PhD degrees from Nara Institute of Science and Technology (NAIST), Nara, Japan, in 2008 and 2011, respectively. From April 2010 to July in 2011, he was with NAIST as a JSPS, Japan Society for the Promotion of Science, Research Fellow. From August 2011 to March 2015, he was with NAIST as an Assistant Professor of the Robotics Laboratory in the Graduate School of Information Science. From April 2014 to March 2015, he was a visiting scholar of Robotics Institute in Carnegie Mellon University, and from April 2015 to 2017, he was a postdoctoral fellow of the same institute. From 2017 to present, he is with Tohoku University as an Assistant Professor of the Graduate School of Information Sciences.

His research interests include motion learning of robots, reinforcement learning applications to robots, machine learning, artificial intelligence, robotic manipulation, and tactile sensing.

Christopher G. Atkeson is a Professor in the Robotics Institute and Human–Computer Interaction Institute at Carnegie Mellon University. He received the M.S. degree in Applied Mathematics (Computer Science) from Harvard University and the Ph.D. degree in Brain and Cognitive Science from M.I.T. He joined the M.I.T. as faculty in 1986, moved to the Georgia Institute of Technology College of Computing in 1994, and moved to CMU in 2000.