Int J Adv Manuf Technol
DOI 10.1007/s00170-013-5174-8

ORIGINAL ARTICLE

A system for automated disassembly of snap-fit covers

Paul Schumacher · Musa Jouaneh

Received: 6 November 2012 / Accepted: 3 July 2013

Abstract This paper reports on a prototype automated system for the disassembly of batteries from a family of electronic devices whose plastic, snap-fit covers house AA or AAA batteries, such as remote controls and calculators. Included in the development of the prototype system was the design of a disassembly tool that uses three force-sensing resistors to provide force feedback information. A pneumatically actuated vacuum gripper and electromagnet system was also developed for recovering the snap-fit cover and batteries once they were released by the disassembly tool. The disassembly module was mounted on the tool head of a three-axis translational motion robot, and a Visual Basic application was developed to interface and control the robot with a Galil digital motion controller. A model-based computer vision application was also developed in Visual C++ using a Kinect sensor and the Open Source Computer Vision library to identify and localize the electronic device placed on the disassembly robot. Using the information gathered by the model-based computer vision application, the robot was able to use the disassembly tool module to perform the necessary disassembly operations to remove the device's snap-fit cover and batteries. The design of such a disassembly system could aid the future development of fully autonomous disassembly systems that can handle a broader range of electronic products.

Keywords Automated disassembly · Force-sensing resistors · Snap-fits · Kinect for Windows

P. Schumacher · M. Jouaneh (✉)
Department of Mechanical, Industrial, and Systems Engineering, University of Rhode Island, Kingston, RI 02881, USA
e-mail: jouaneh@egr.uri.edu

Present Address:
P. Schumacher
Sensata Technologies, Attleboro, MA, USA

1 Introduction

The problem of electronic waste (e-waste) is a growing worldwide issue. E-waste is the fastest growing and most toxic component of municipal garbage [1]. Consumer electronics, especially computers and televisions, contain more than 700 different types of materials, a number of which are hazardous [2]. Some of the hazardous materials found in e-waste are the following: lead, cadmium, mercury, hexavalent chromium, and brominated flame retardants. These materials have damaging effects on humans and the environment.

One proposed way to protect human health from hazardous materials and increase efficiency and economic benefits is to develop automated systems for the disassembly of e-waste. Manual disassembly is costly and time-consuming and exposes workers to hazardous materials. As a result, many electronic devices are simply fed into large shredders [3]. The shredded e-waste is then processed to extract different materials. While some plastics and metals can easily be separated during this process, additional separation by different types of metals and different types of plastics is a difficult task. Consistent collection and processing of recycled glass and plastics at acceptable quality and cost levels so as to be used in new products is an even more difficult recovery job [4]. Therefore, creating an intelligent, automated system of robots for the disassembly of end-of-life electronics could increase both the quantity of precious and toxic materials recovered and the quality of the material recovered. In order to efficiently and economically automate the disassembly of electronic devices, the development of flexible disassembly tools is required.

According to Metech [3], a company active in electronics recycling and asset recovery, one of the most common hazardous materials that must be recovered from electronics is batteries. For many electronic devices, such as calculators, remote controls, alarm clocks, music players, etc., the batteries are contained behind a plastic panel fastened with a

Published online: 24 July 2013

single, cantilever snap-fit. Snap-fit fasteners are very common in the design of electronic devices, especially in the outer plastic housings. When removed, the absence of these plastic housings allows further access to the components contained within the device. Therefore, creating a device to remove snap-fits is essential to the development of an automated disassembly system. While snap-fits work excellent for rapid assembly by decreasing the number of components and replacing screws and nuts, they present more challenging issues for disassembly [5]. During assembly, snap-fit components are merely oriented and pushed into place, which is a simple process requiring basic manipulation. In comparison, during disassembly, the cantilever of the snap-fit must be disengaged while simultaneously pulling or sliding the component out of the assembly. Disengaging snap-fits requires a special tool and more sophisticated coordination. Due to the necessity of recovering batteries for proper electronic waste disposal, this paper reports on the design of a disassembly system capable of releasing snap-fits with the purpose of recovering the hazardous batteries enclosed within an electronic device. A robotic mechanism for recovering batteries is a good starting platform for the eventual development of an intelligent, automated electronics recycling system.

Several researchers have investigated the design of disassembly tooling and grippers. Rebafka et al. [6] proposes a flexible unscrewing tool that creates its own acting surfaces in order to improve loosening. Park and Kim [7] discuss the development of a six-axis force/moment sensor for an intelligent robot's gripper. Feldmann et al. [8] detail the design of a so-called drilldriver, which can be used for three different methods of disassembling joining members: use an existing working point to remove a fastener, drill to create a working point to remove a fastener, or drill to destroy the joint.

A number of researchers have also reported on the development of semi-automated disassembly cells. One such study presented the components of a flexible disassembly cell for obsolete TVs and monitors [9]. The disassembly cell was able to disassemble various types of TVs and monitors and react on small changes of the state and the properties of the product. The conclusion of the study indicated that special disassembly tools and fixtures are needed to be further developed in order to avoid complicated disassembly operations and robot failures. Another case study presented a prototypical disassembly factory for large electronic household appliances, such as washing machines [10]. The disassembly system utilized a number of specially designed disassembly tools integrated as modules. At the Darmstadt University of Technology, Germany, two systems [11] were developed—one for disassembly of video cameras and the other for personal computers. At the University of Alicante in Spain [12], another system was developed for the disassembly of personal computers. Furthermore, Basdere and Seliger [13] describe a hybrid disassembly factory for cellular phones developed in reaction to the dramatic growth of cellular phone users worldwide. A similar study on electronic waste concluded that disassembly automation is absolutely necessary worldwide in the near future because of the dramatically increasing amount of electronic scrap [14].

The remainder of this paper is organized as follows. The next section discusses the overall design of the automated disassembly system. This is followed by a discussion of the three main components of the system: disassembly module, robotic platform and controller, and Kinect sensor. An illustration of the disassembly operations is discussed in Section 4. The force sensor readings are discussed in Section 5. Section 6 presents analysis of the Kinect vision system, while Section 7 discuses another tool tip for internal snap-fits. The concluding remarks are given in the last section.

2 Overall system design

The complete automated disassembly system is shown in Fig. 1 and has three main components: a sensor-based disassembly module, a robotic platform and controller, and a vision system. The disassembly module consists of a force-based disassembly tool, a vacuum gripper, and an electromagnet recovery system. The disassembly module was mounted on the tool head of the three-axis robot. The vision system uses a Kinect for Windows sensor located above the robot base. In the following section, each component of the disassembly system will be discussed in further detail.

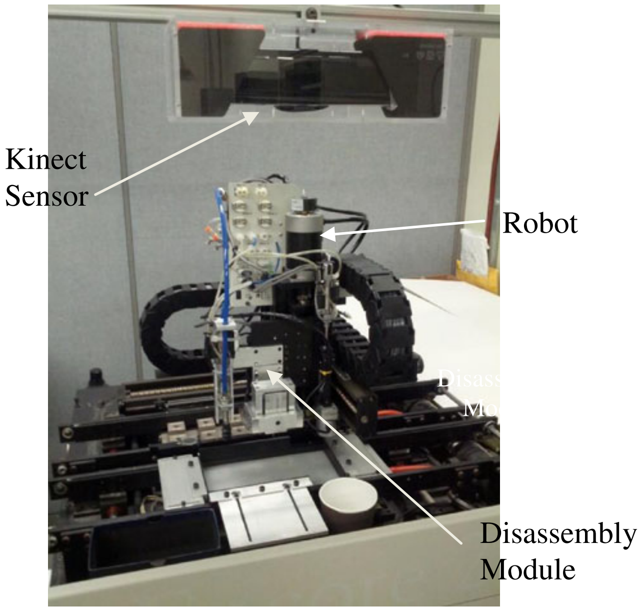


Fig. 1 Overall disassembly system

It should be noted that this paper only concerns the design of a disassembly tool that uses careful manipulation to release the snap-fits and batteries. It is possible that destructive methods could be a quicker alternative to removing the snap-fit cover. Other methods could also be explored for the removal of the batteries, such as a vibratory system that shakes the batteries out of the electronic device. However, the system discussed here was designed using nondestructive strategies.

3 System components

3.1 Disassembly module

A sketch of the disassembly tool is shown in Fig. 2.

The basic concept of this design was to create a force sensing tip that could unfasten the snap-fit utilizing the *x*–*y*–*z* translational motion of the robot. Rather than using expensive load cells, the tool is equipped with three, low-cost force-sensing resistors (FSRs) to detect horizontal forces applied to the tool tip along the *x*-axis and to detect vertical forces applied to the tool tip along the *z*-axis. When a horizontal force is applied to the tip, the main block swivels and touches an FSR located inside the main housing. Likewise, when a vertical force is applied to the tool tip, it slides up in the bushing and touches the FSR in the main block. The tool functions by moving to the snap-fit location and then moving the tip down until it contacts the surface of the electronic device. When the tip touches down, the FSR in the main block will register the vertical force and signal the tool to stop moving in the *z*-axis. Then the tool will move in the *x*-axis and push the snap lever with the tool tip until it is fully deflected. The tool knows when the snap-fit is fully deflected by monitoring the force readings of the appropriate FSR in the main housing. The tool then moves up in the *z*-axis, releasing the snap-fit. Again, the tool recognizes when the snap-fit is unfastened because the force read by the FSR will decrease to zero as the snap-fit is released. Another important aspect of this design concept is that the same disassembly routine used to unfasten the snap-fit can also be applied to releasing the batteries. In the case of the batteries, the tool tip is inserted on the edge of the battery opposite the spring. The tool then pushes the battery, compressing the spring. Once the spring is compressed, the tool moves up in the *z*-axis releasing the battery. The details of the disassembly tool design and the characterization of the FSRs used in this tool are discussed in [15].

The prototype disassembly tool provides the means to release the snap-fit and batteries; however, these components are also needed to be lifted and removed from the electronic device. Since plastic snap-fit covers are lightweight and in most cases have a relatively flat contour, a vacuum gripper system was developed to retrieve the plastic covers from the electronic devices. The vacuum gripper system is labeled 1 in Fig. 3 which shows a picture of the complete assembly module. The gripper system consists of a suction cup, spring compensator, and pneumatic actuator. The pneumatic actuator is used to raise and lower the suction cup. The spring compensator is used to adjust to variations in object height and ensures that cup pressure is consistent on each pickup which increases the life of the cup. A double bellows suction cup is utilized for additional height compensation and for

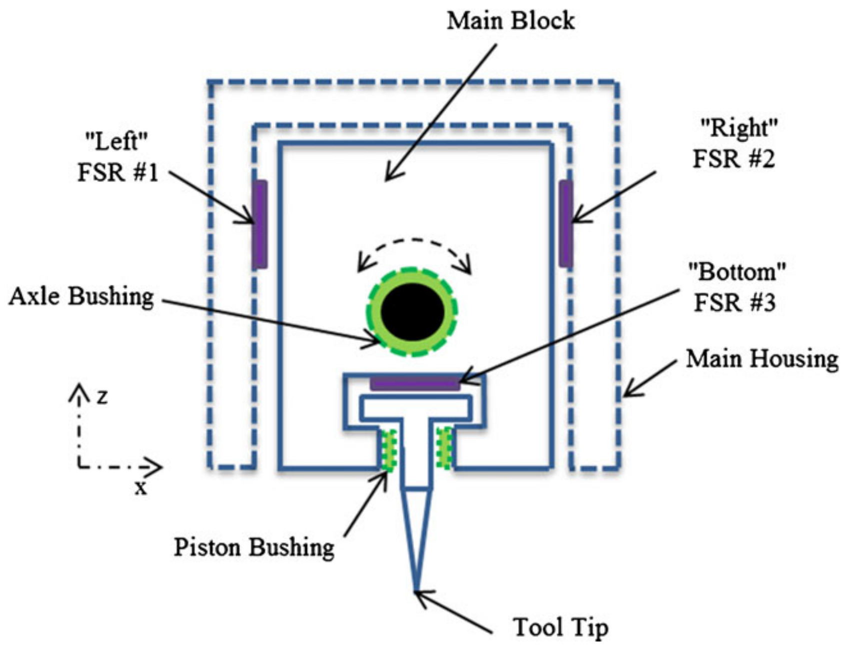


Fig. 2 Sketch of the disassembly tool

greater flexibility in the removal of the plastic cover. The surface of the plastic covers is relatively flat, but once they are unfastened for retrieval, they are propped up at an angle. The double bellows design compensates for this slight angle and does not require additional complex positioning routines.

Due to a battery's convex shape and limited vacuum pressure, a small vacuum gripper is not suitable for lifting batteries. In order to simplify the coordinated motion necessary to recover the batteries and take advantage of their magnetic properties, an electromagnet retrieval system was developed. The electromagnet used is a Kanetec magnetic holder model KE-3RA. This electromagnet has a maximum holding power of 30 N (6.74 lbs) under ideal conditions, which is strong enough to pick up both AA and AAA batteries. An AA battery weighs approximately 23 g (0.051 lbs), and an AAA battery weighs approximately 11 g (0.024 lbs). The battery retrieval system is labeled 3 and includes the electromagnet, a 0.2-in. Interlink FSR, and a pneumatic actuator. Again, the pneumatic actuator is used to raise and lower the electromagnet so that it may reach below the level of the disassembly tool. The FSR is housed inside an aluminum piston block and is used as a touch sensor to indicate when the electromagnet contacts the surface of the battery. This allows the electromagnet retrieval system to adapt to differently sized electronic devices and batteries. The electromagnet is attached to a piston that moves up inside the piston block and presses the FSR when a vertical force is applied to the bottom of the electromagnet. In order to hold the devices in place during disassembly, an adjustable vice was built onto the robot's base. The vice consists of one stationary bracket (labeled 4) and three adjustable brackets (labeled 5) as shown in Fig. 4. The adjustable brackets allow the vice to adapt to differently sized devices and to vary the position of the devices. Each bracket is lined with a rubber pad to increase the vice's gripping ability.

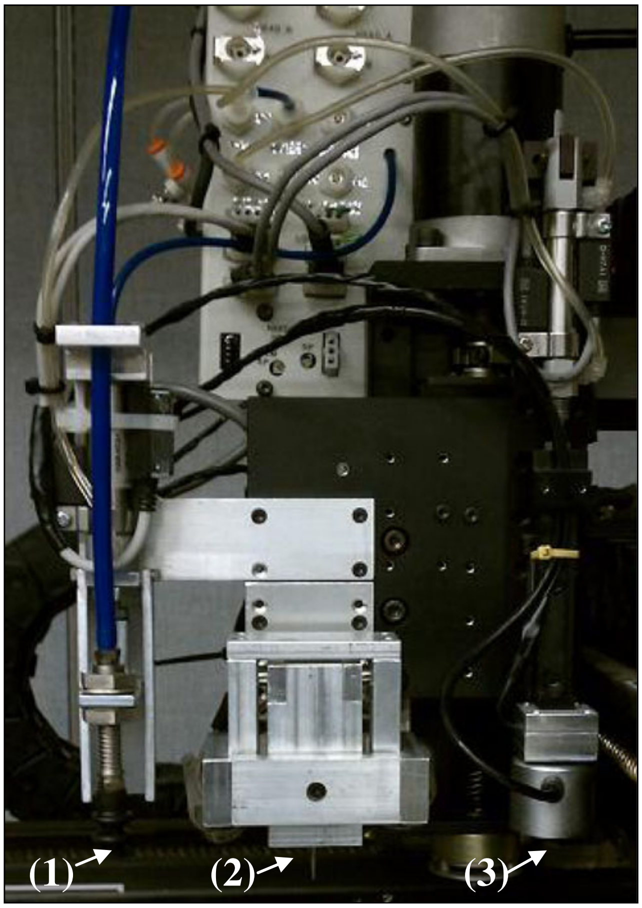


Fig. 3 Disassembly module

3.2 Robotic platform and controller

The robotic base was originally used for small electronic component assembly (ENCore 2S System from MRSI, North Billerica, MA) and was modified to use for this application. The modified system is controlled with a Galil DMC-2143 four-axis digital motion controller. Microsoft Visual Basic 2010 Express was used to develop a graphical user interface (GUI) to control the system operation. An image of the developed GUI is shown in Fig. 5. The GUI was designed for both manual and automated control.

The Robot Visual Basic application provides three different methods for control: jog control, position control, and automated control. Only one type of control can be enabled at any one time. The jog control method essentially provides a joy stick-like control of the *x*–*y*–*z*-axes. The speed of each axis can be set using the *axis speed selection* box. The position control method allows the user to specify a desired set of *x*–*y*–*z*-coordinates for the tool head module to move to. The automated control method implements the disassembly routine for removing snap-fits and batteries. The automated disassembly system utilizes both the Robot Visual Basic application and a separate Kinect Visual C++ application. The disassembly method uses a model-based vision system for recognizing the electronic device loaded on the machine and localizes it in the image frame. The Visual Basic and Visual C++ applications communicate with each other

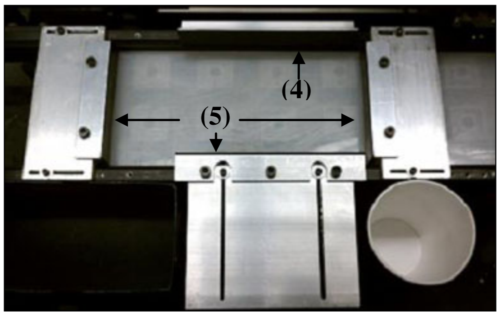
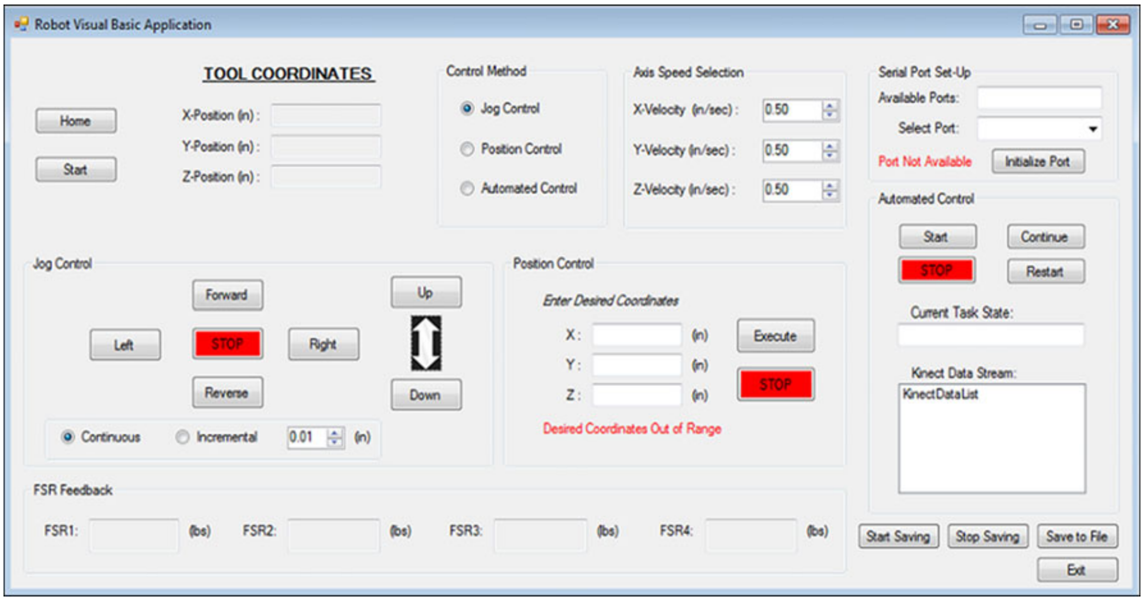


Fig. 4 Adjustable vice



Control Method: [✓ Jog Control] Position Control Automated Control

X-Velocity (in/sec): 0.50
Y-Velocity (in/sec): 0.50
Z-Velocity (in/sec): 0.50

Port Not Available

[✓ Continuous] Incremental 0.01 (in)

Desired Coordinates Out of Range

Fig. 5 Robot Visual Basic GUI

through a serial port. A block diagram of all the main system components is shown in Fig. 6.

A state-transition diagram [16] for the operation of the automated disassembly system is illustrated in Fig. 7. In order to simplify the diagram, not all transition conditions are included. However, the diagram does serve to summarize the flow of the disassembly routine. The main state-transition routine runs in the Robot Visual Basic application. By pressing the *Start* button, the application enters the first state of the disassembly sequence, "Initialize Automated System." The disassembly routine has a total of 15 states that each performs a specific function in the disassembly process. After the system is initialized, the robot cycles through the automated disassembly states until the snap-fit cover and all batteries are removed from the electronic device. At this point, the disassembly routine enters the "Task Completed" state, and the application waits for the user to load a new electronic device on the robot and press the *Restart* button to reinitialize the disassembly routine back to state 1. The automated control can be stopped at any point during the disassembly routine by pressing the *Stop* button.

3.3 Kinect sensor

A computer vision application was integrated into the robot in order to add flexibility to the automated disassembly system. The computer vision application enhances the robot by performing the following operations:

(a) recognizes the current electronic device and its orientation by utilizing a data bank of device models;
(b) localizes the *x*–*y* pixel coordinates of the device origin in the image frame; and
(c) measures the relative depth of the device.

These features allow the robot to adapt to changes in the type of electronic device and the location of that device. The system is, of course, limited to the device models saved in the data bank. Nevertheless, the ability to recognize and disassemble multiple devices is essential to dealing with the large variety of electronic waste. Furthermore, in a completely autonomous disassembly system, the electronic devices would need to be picked and placed on the disassembly platform with an additional robot. An adaptive vice would also have to be added to tighten automatically on the differently sized devices. Small variations in the position of the devices would occur due to the automated adaptive vice. Thus, the ability to localize the device and adapt to small position changes is an important aspect of the automated disassembly system.

The implementation of the computer vision system was realized with a Kinect for Windows sensor and the Open Source Computer Vision (OpenCV) library [17]. The Kinect sensor features a RGB camera, infrared sensor, and multi-array microphone. Traditionally, in order to achieve a 3D

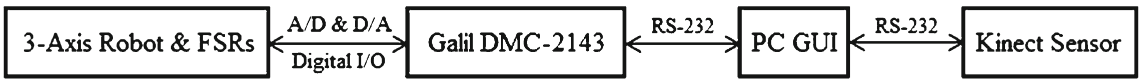


Fig. 6 System component block diagram

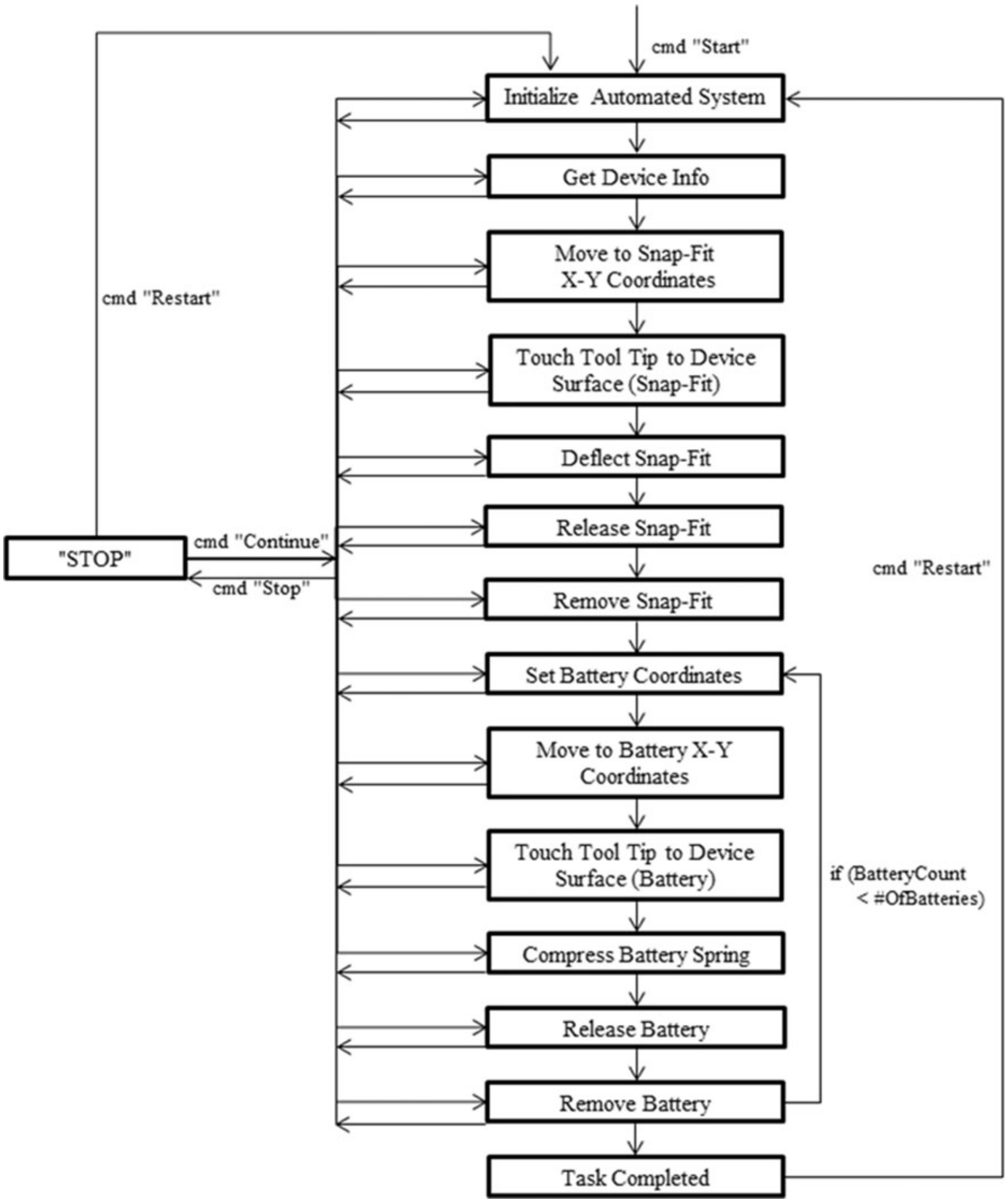


Fig. 7 Automated disassembly state-transition diagram

computer vision system, the technology of stereo cameras is used in which 3D information is extracted from a scene by comparing information from two cameras at different vantage points. The Kinect, however, uses a built-in infrared sensor to extract 3D depth information from a scene. A cheap stereo vision system can be bought for about $600, but most commercially available systems are on the order of at least several thousand dollars. In comparison, the Kinect can be purchased for as little as $150. The Kinect sensor was chosen for this application due to its low cost and ability to stream both raw depth and color data frames. The Kinect sensor was also chosen because it comes with the Kinect for Windows Software Development Kit, which provides the capability to build applications with C++, C#, or Visual Basic by using Microsoft Visual Studio 2010. OpenCV is a library of programming functions for real-time computer vision. OpenCV can interface with a number of programming languages; however, Visual Basic is not supported. As a result, the computer vision application was written in Microsoft Visual C++ 2010 Express in order to interface both the Kinect and OpenCV library.

A plastic box was designed to house the Kinect sensor in a centralized position over the adjustable vice. An image of the Kinect hardware inside its plastic housing can be seen in Fig. 8.

The Kinect Visual C++ console application starts by opening the Kinect data streams and initializing the serial port. After the Kinect data streams are opened, a thread is created to process the incoming color and depth frames. The program then enters an infinite do loop where it constantly reads the serial port to receive commands from the Robot Visual Basic application. When a command is received, the program selects the appropriate section of code based on the command value. The application utilizes four different states which are assigned as follows:

Case 1: "Perform template matching"
The function *TemplateMatching*( ) is called to determine the current device type and localize it in the image frame. The device type is then written to the console and sent as a coded number to the Visual Basic application through the serial port.

Case 2: "Send *x* origin pixel"
The *x*-coordinate of the origin pixel for the device that was calculated by the template matching function is written to the console and sent to the Visual Basic application.

Case 3: "Send *y* origin pixel"
The *y*-coordinate of the origin pixel for the device that was calculated by the template matching function is written to the console and sent to the Visual Basic application.

Case 4: "Determine device depth"
The function *GetDeviceDepth*( ) is called to measure the relative depth of the device to the Kinect sensor. The depth value is then written to the console and sent to the Visual Basic application.

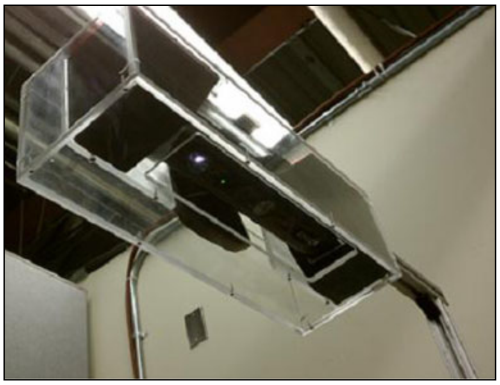


Figure 8 Kinect housing image

The template matching function is a method for finding areas of a source image that match to a template. The source image is a color frame from the Kinect in which it is expected to find a match to the device template image. In order to identify the highest matching area, the OpenCV function *matchTemplate*() is called which compares the template image against the source image by sliding it across the source image. The template image is moved one pixel at a time (left to right, up to down), and at each location, a metric is calculated to represent how well the template matches the source at that location. For this application, the metric is a normalized value from 0 to 1. The match values are stored in a result matrix. The OpenCV function *minMaxLoc*( ) is then called to locate the pixel coordinates of the highest match value in the result matrix, which corresponds to the location of the top left corner (origin) of the template image. After the template matching is complete, the source image is displayed with a rectangle drawn around the matched template as shown in Fig. 9.

In the Kinect Visual C++ application, the template matching function cycles through an array of the device template images and determines the device with the highest match value. Furthermore, in order to make the system more flexible, two templates are created for each device in the data bank, one for the "left" orientation and one for the "right' orientation. Thus, the vision application is able to inform the robot application of the orientation of the electronic device. If the highest match value of all the devices is less than 0.8, then it is determined that no match was found. This threshold value was determined through testing. Due to the template matching function's sensitivity to lighting variations, the range 0.8 to 1 allows for a good match to still be detected with slight lighting changes. However, once the match value is below the 0.8 threshold, a device cannot be reliably recognized or localized.

The function *GetDeviceDepth*() determines the approximate depth of the device loaded in the vice. The Kinect depth stream returns image frames where each pixel holds a depth value in millimeters. The approximate depth of the device is evaluated by computing the average pixel depth across the length of the center of the device. While electronic devices vary in both size and shape, the center line across the length of the device is, in most cases, the high point for devices of the device family explored in this application. The depth value sent to the robot application is used to calculate the height of the device in world coordinates. This allows the tool tip to rapidly move down to a height just above the approximate depth and then slowly move down from that point until the FSR indicates that the device surface has been reached.

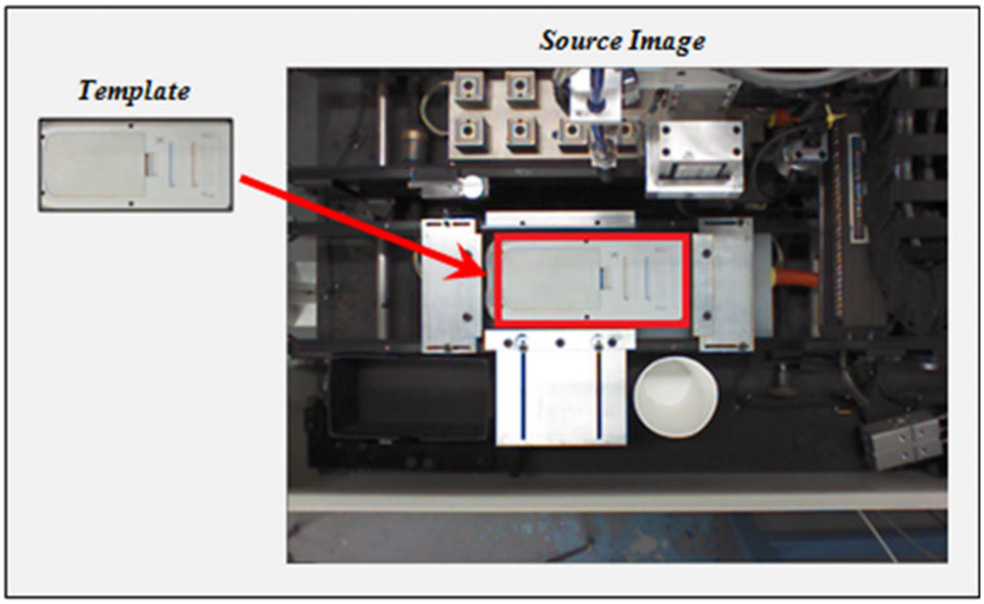


Fig. 9 Template matching example

4 Illustration of disassembly routine

The automated disassembly routine starts by moving the tool head to a position that allows the Kinect a clear view of the electronic device, as seen in Fig. 10. In this case, the electronic device is a TV remote control.

Once the tool head is in position, a command is sent to the Kinect Visual C++ application to begin its template matching function. The Kinect application determines which device is loaded in the vice, localizes the device in the image frame, and measures its depth. The image frame is also displayed which shows the localized device with a box around it. The Kinect Visual C++ application then displays this information in the console and sends it to the Robot Visual Basic application through the serial port. The Visual Basic application then calls a function to look up the parameters of the recognized device and transforms the pixel coordinates of the image frame into world coordinates. At this point, the robot is able to begin performing disassembly operations. An image displaying the matched device is shown in Fig. 11. The C++ console output and the data transferred to the Visual Basic application are pictured in Figs. 12 and 13, respectively.

The disassembly tool moves in the *x*–*y*-plane to the snap-fit location. The tool is then moved down in the *z*-axis until it contacts the device's surface, at which point the tool tip is moved in the appropriate direction to deflect the snap lever. Once the tool detects the snap lever is fully compressed from reading the force sensor feedback, the tool moves up in the *z*-axis to release the snap-fit. Once the snap-fit is unfastened, the vacuum gripper moves into position over the snap-fit plastic cover. The pneumatic actuator is then activated, the vacuum valve is switched on, and the *z*-axis is moved down until the suction cup makes a seal on the plastic cover. After the seal is made, the pneumatic actuator is moved up, and the snap-fit cover is brought to its respective receptacle for disposal. The operations to remove the plastic cover can be seen in Fig. 14.

The next task is to remove the batteries. The batteries are released essentially the same way as the snap-fit. The tool tip is inserted against the edge of the battery opposite the spring.

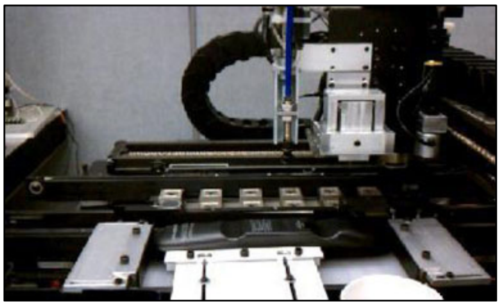


Fig. 10 Dissassembly tool moved to clear Kinect view

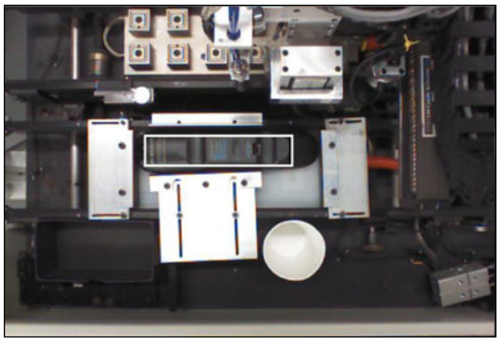
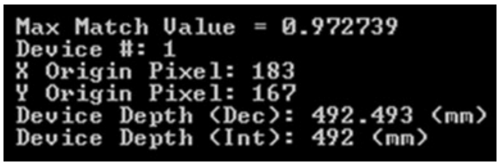


Fig. 11 Matched device image frame



```
Max Match Value = 0.972739
Device #: 1
X Origin Pixel: 183
Y Origin Pixel: 167
Device Depth (Dec): 492.493 (mm)
Device Depth (Int): 492 (mm)
```

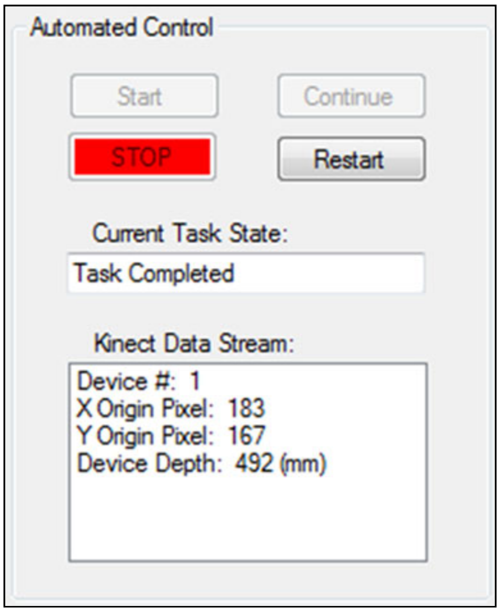
Fig. 12 C++ console output

The battery is then pushed in the *x*-axis, compressing the spring. Once the force sensors indicate that the spring is fully compressed, the tool is moved up in the *z*-axis to release the battery. An electromagnet mounted on a pneumatic actuator is then used to retrieve the battery. The electromagnet is also equipped with a force sensor to indicate when it has made contact with the surface of the battery, at which point the pneumatic actuator is triggered up, and the battery is lifted out of the device. The battery retrieval operations are illustrated in Fig. 15.

The system will continue to remove the batteries until they are all removed. Since the orientation of every other battery is flipped opposite to its neighbors, the tool is needed to work in either direction. In the abovementioned device, the first battery in Fig. 15(8) was orientated so that the tool moved in the positive *x*-direction (right) to compress the spring, whereas in Fig. 15(12), the second battery is released by moving the tool in the negative *x*-direction (left). Likewise, the tool can also adjust to the orientation of the snap-fit when moving to deflect the cantilever.

5 Force sensor readings

The disassembly tools utilize force sensor feedback to control the disassembly sequence of both the cover and the batteries.



Current Task State:
Task Completed

Kinect Data Stream:
Device #: 1
X Origin Pixel: 183
Y Origin Pixel: 167
Device Depth: 492 (mm)

Fig. 13 Serial transmission of data

We display below the force sensor readings recorded during the disassembly operation. The position feedback from the encoders is also shown. In all of the following force plots, FSR1 corresponds to the force sensor that reads horizontal forces when the tool is moved left or along the negative *x*-axis, FSR2 corresponds to the force sensor that reads horizontal forces when the tool is moved right or along the positive *x*-axis, FSR3 corresponds to the force sensor that reads vertical forces applied to the tool tip along the *z*-axis, and FSR4 corresponds to the force sensor that reads vertical forces applied to the electromagnet along the *z*-axis.

The first disassembly operation analyzed is releasing a snap-fit with a right orientation which is shown in Figs. 16 and 17. The *z*-axis moves down quickly until it reaches the depth originally measured by the Kinect sensor and from that point slowly moves down until it comes into contact with the calculator surface. When the tool tip reaches the calculator surface, the FSR3 measures the vertical force applied to the tool, and *z*-axis motion is stopped as seen at approximately 0.35 s. The tool then moves to the right to deflect the snap-fit. As the tool tip presses the snap-fit lever, the force measured by the FSR2 increases until it reaches the programmed threshold force (18 N) that indicates that the snap-fit is fully deflected. When the threshold force is reached (t=0.68 s), motion stops in the *x*-axis, and the *z*-axis is moved up to release the snap-fit. While the *z*-axis moves up, the force measured by the FSR2 decreases as the tool tip swings back to the neutral position. Once the force read by the FSR2 is approximately zero (t=0.76 s), the tool recognizes that the snap-fit has been successfully released and is ready for removal. After the snap-fit is released, the *z*-axis continues to move up to its zero position before the vacuum gripper is used to retrieve the snap-fit cover.

The disassembly method used to release the batteries is similar to that used to unfasten the snap-fit. Thus, the force response plot for the batteries is also comparable to that of the snap-fit. Figures 18 and 19 show the interactions for the release of a battery with a right orientation.

The *z*-axis is moved down to the edge of the battery opposite the spring until the FSR3 reads a force applied to the tool tip in the vertical direction (t=0.4 s). The tool is then moved right, pushing the battery and compressing the spring. Once the FSR2 measures a force greater than the threshold force (18 N), motion on the *x*-axis stops, and the *z*-axis is moved up to release the battery (t=0.6 s). As one can see in Fig. 18, the FSR2 reaches a peak force slightly higher than the threshold force, which is due to a small delay between the force sensor signal and the command sent to stop motion in the *x*-axis. Once the FSR2 indicates that the battery is successfully released, the *x*- and *y*-axes are moved to center the electromagnet over the battery (t=0.76 s). After the pneumatic actuator is triggered, the *z*-axis is moved down until the FSR4 measures a force applied on the electromagnet (t=1.18 s). At

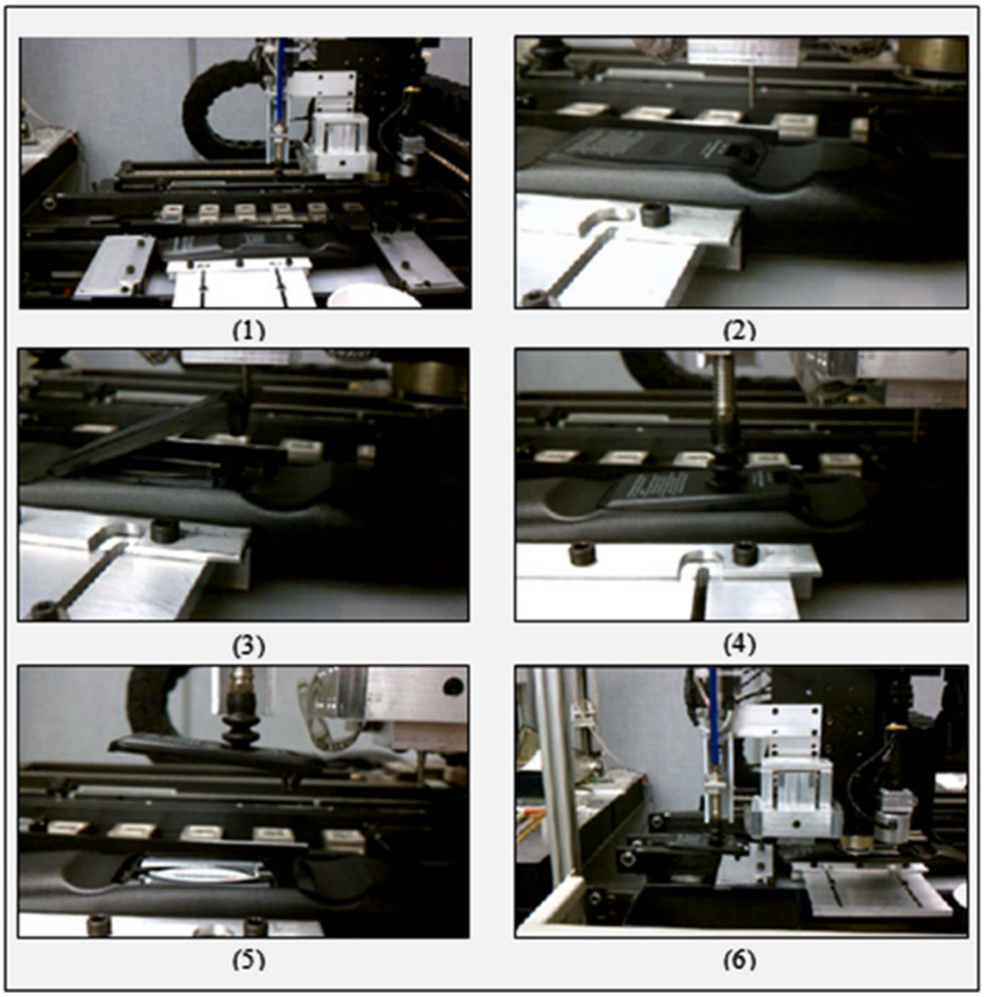


Fig. 14 Snap cover disassembly sequence

this point, the pneumatic actuator fires up, the *z*-axis is moved up, and the electromagnet carries the battery to its respective bin for disposal. The same adaptive force method used for a battery with a right orientation is used to release a battery with a left orientation. In this case, however, the FSR1 is the sensor that measures the force applied to the tool tip as it pushes the battery against the spring.

When the position of the calculator is shifted in the *x*–*y*-plane towards the edge of the image frame, small inaccuracies can occur in the device's localization. Errors on the order of one to two pixels have been observed in the localization, which corresponds to approximately 0.889–1.91 mm in world coordinates. Since the coordinates of the snap-fit and batteries are calculated based on the localized origin pixel coordinates, this error will carry over into these calculations. While the error is relatively small, the space between the device case and the edge of the battery is also on the order of 1.27–1.91 mm. As a result, instances occurred where the tool tip slightly missed the edge of the battery and instead touched down on the top surface of the battery. In reaction to this occurrence, an error routine was developed to adapt to these small inaccuracies.

The error routine essentially checks to make sure that the tool tip catches the appropriate edge of the battery and begins to push the battery against the spring. When the *z*-axis is moved down and detects a vertical force in the tool tip, the tool begins to move along the *x*-axis in the appropriate direction to compress the spring. However, at this point, the tool is unaware if it actually touched down in the correct position. When the tool begins its movement in the *x*-axis, the distance it travels is recorded. If the tool moves more than 5.0 mm without detecting a horizontal force, the tool realizes that it did not touch down in the correct position and moves back to readjust its start position. The tool tip moves up in the *z*-axis and then moves to a position offset 0.75 mm from the original start position along the *x*-axis in the direction opposite the motion of the tool. The tool tip then retries its attempt to release the battery from the new start position. The disassembly tool will make a maximum of three attempts to release any one battery. The error routine can also be used to detect if a battery is missing from an electronic device. The tool will still try to release a battery as if one was there, but after three failed attempts at catching a battery edge, the tool

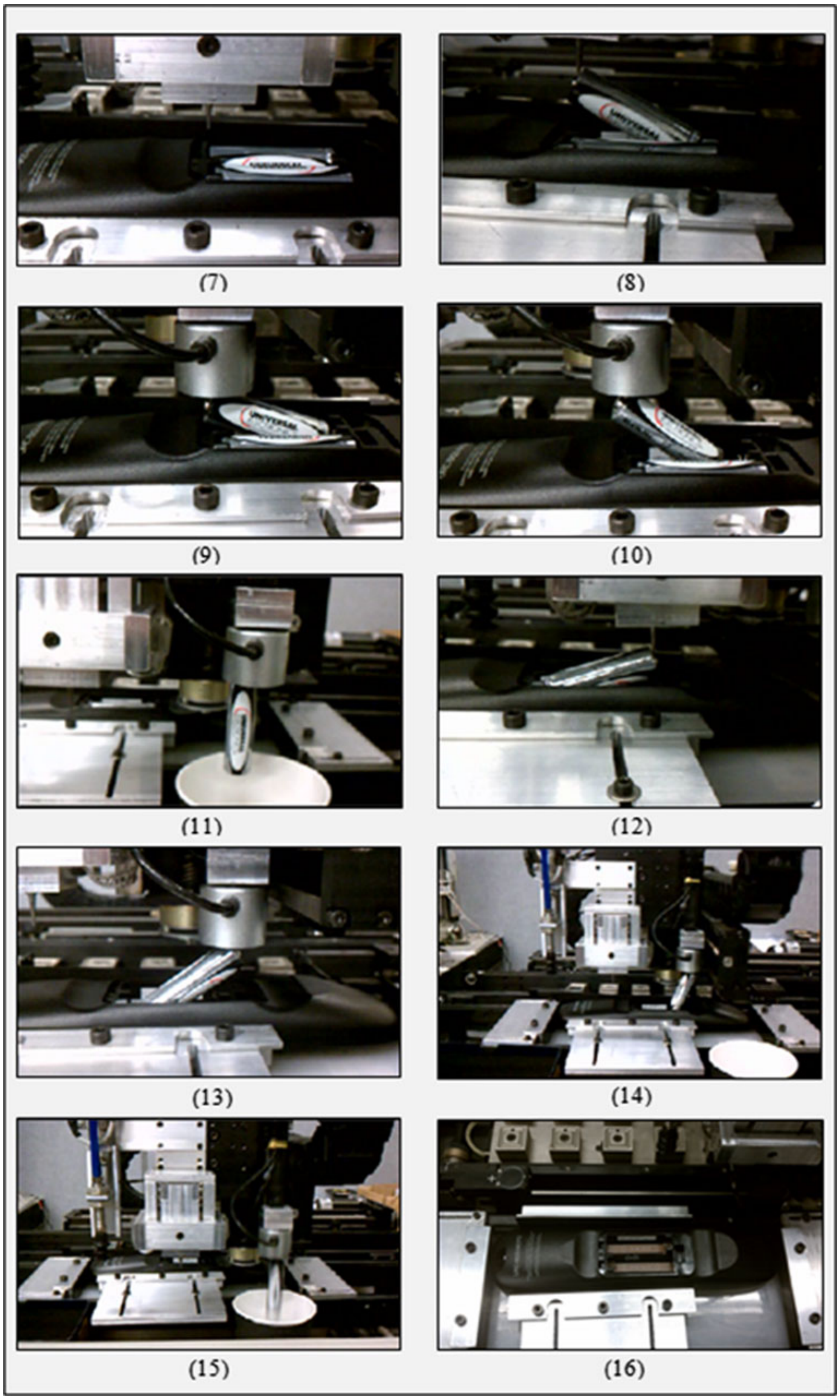


Fig. 15 Battery removal disassembly sequence

will stop and prompt the user that an error has occurred releasing the battery. At this point, the user has the option to command the tool to move to the next battery or restart the disassembly routine.

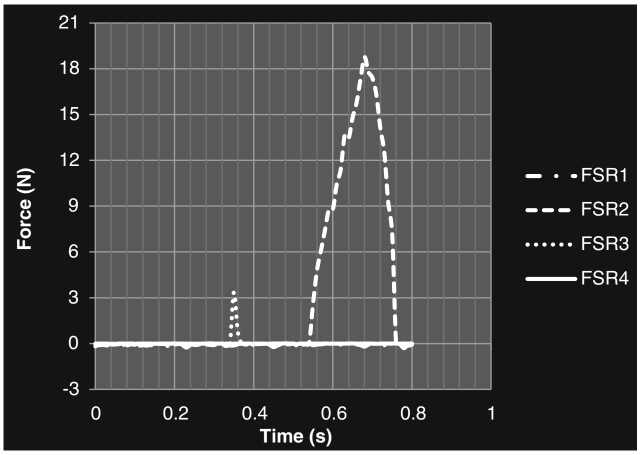


Fig. 16 Force response plot for a snap-fit with right orientation

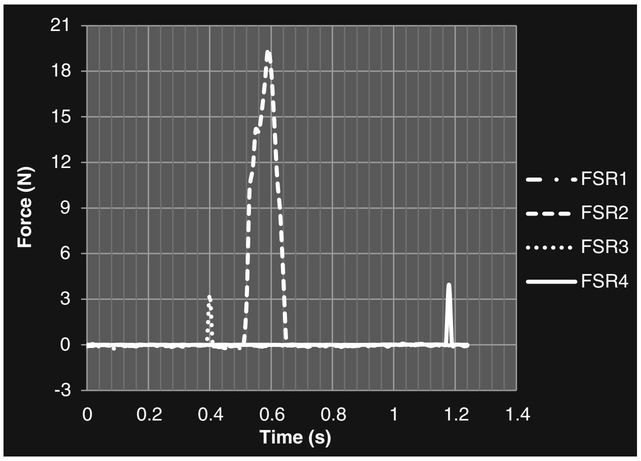


Fig. 18 Force response plot for a battery with right orientation

6 Kinect vision system analysis

The Kinect sensor combined with the OpenCV library enables the automated disassembly system to recognize the present device, localize it in the image frame, and determine the approximate depth of the device. Furthermore, the Kinect system recognizes if the device loaded in the vice is not saved in the data bank. In order to do this, a threshold value is set for the *max match value*, which is a normalized value from 0 to 1 that represents how well the current device matches a device in the data bank. Through experimentation, the max match threshold value was set to 0.8. Thus, if no device in the data bank results in a max match value greater than 0.8, then the Kinect system informs the user that no match was found to the current device loaded on the robot.

Further functionality of the Kinect system is shown in Figs. 20 and 21. The Visual C++ console output is displayed along with an image that shows the device localized in the image frame. Figure 20 shows the results for the calculator that shifted in the *x*–*y*-plane, and Fig. 21 shows the results for the calculator with a small change in height. These results demonstrate how the Kinect system is able to adapt to changes in device orientation, and position.

As noted earlier, localization errors of one to two pixels occur when the device is shifted in the *x*–*y*-plane towards the edge of the image frame. This error is most likely due to the fact that the template matching function is sensitive to lighting conditions, which vary as the device is moved to different positions. Localization accuracy also decreases as the height of the device increases. The Kinect system is robust against small changes in height, but the results weaken as the height change continues to increase. This phenomenon is due to the fact that the template images taken for each device are taken at a finite height. As the height of the device increases, it moves closer to the Kinect camera, causing the device to appear larger in the image frame. As a result, the original template of the device will not match with the distorted image of the device at a different height. In order to quantify this effect, the TI calculator was localized at different height increments until the Kinect no longer accurately detected its position. The results of this test are displayed in Table 1.

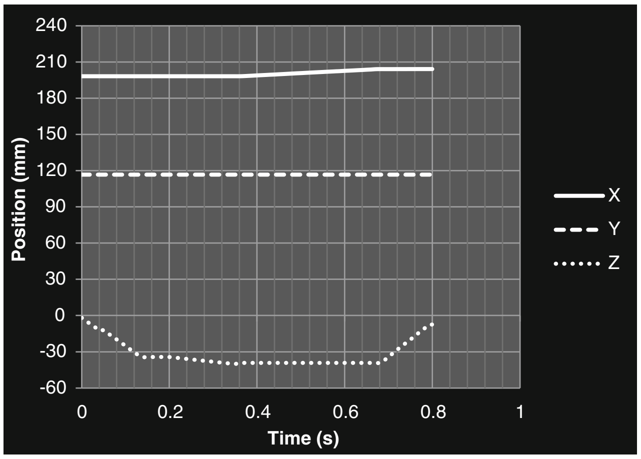


Fig. 17 Position plot for a snap-fit with right orientation

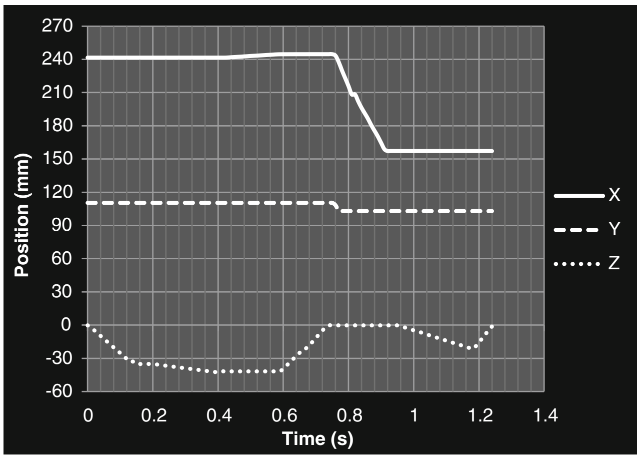
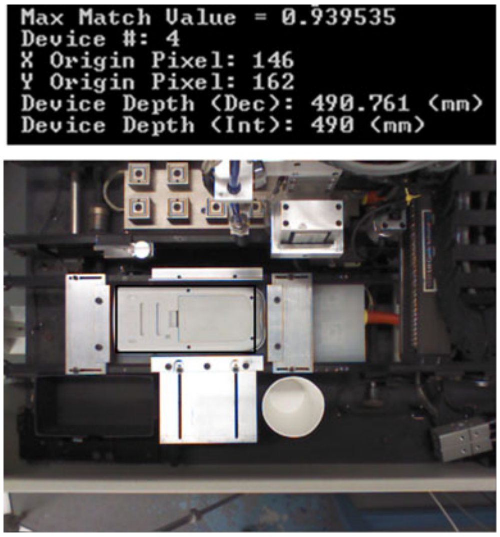


Fig. 19 Position plot for a battery with right orientation



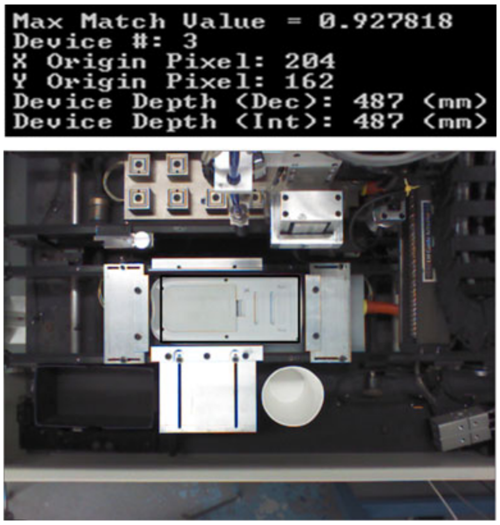
```
Max Match Value = 0.939535
Device #: 4
X Origin Pixel: 146
Y Origin Pixel: 162
Device Depth (Dec): 490.761 (mm)
Device Depth (Int): 490 (mm)
```

Fig. 20 Kinect system output—TI-84 calculator *x*–*y* position shift

In Table 1, as the height of the device increases, the match value decreases. However, for height changes from 0 to 6 mm, the localization accuracy does not change. At a height change of 9 mm, the localization begins to decrease as the *y* origin pixel changes to 161. The localization continues to decrease as the height increases, and at 15 mm, the Kinect is no longer able to recognize the device because its match value decreases below 0.8. Thus, for the most accurate results, the height change of the device should not exceed 6 mm.

In addition to the height adjustment test, the Kinect was tested for its repeatability in recognizing and localizing a device. For this test, the TI-84 calculator was loaded into the vice, and then 25 trials were run for the Kinect system. The results of the repeatability test are displayed in Table 2.

Table 2 indicates that the Kinect system has a highly repeatable output. The Kinect system recognized the correct device and localized it in the same position in each trial.



```
Max Match Value = 0.927818
Device #: 3
X Origin Pixel: 204
Y Origin Pixel: 162
Device Depth (Dec): 487 (mm)
Device Depth (Int): 487 (mm)
```

Fig. 21 Kinect system output—TI-84 calculator height shift

Table 1 Kinect height adjustment results

ΔHeight (mm)	Match value	Device no.	*x* origin	*y* origin	Depth (mm)
0	0.9592	3	204	162	489.151
3	0.9459	3	204	162	486.933
6	0.8978	3	204	162	484.236
9	0.8518	3	204	161	481.191
12	0.8140	3	204	160	479.012
15	0.7882	0	–	–	–

Furthermore, the *match values* and *depth* measurements have very low standard deviation values. In this test, the Kinect system was isolated from the Robot system because the TI-84 calculator was not removed from the vice throughout the entire testing sequence. Therefore, in order to test the combined repeatability of the Kinect and Robot systems, the same test was run except that, in this case, the calculator was removed from the vice between each trial. This repeatability test is essentially determining if the Kinect system is sensitive to small variations that occur when the device is loaded in the vice. In the previous test, the calculator was loaded with a right orientation, whereas in this test, the calculator is loaded with a left orientation, as indicated by the change in device number. The results for the test are displayed in Table 3.

Table 3 indicates that the Kinect system is robust against small variations that take place during the loading of the device. The correct device is recognized and localized in the same position for each trial. The standard deviations of the match values and depth measurements increased slightly from the previous test, but are still well within acceptable values.

7 Internal snap-fit tool tip

Rather than the external U-shaped cantilever snap-fit, some electronic device battery covers use an internal cantilever snap-fit to fasten. These covers are removed by pushing down at the edge where the snap-fit is located and sliding the cover off. An example of this type of internal snap-fit fastener is pictured in Figs. 22 and 23.

Table 2 Kinect repeatability results

	Average	Range	Std. Dev.
Match value	0.965519	0.001345	0.000340
Device no.	4	0	0
x origin	196	0	0
y origin	162	0	0
Depth (mm)	490.701	0.381	0.129938

Table 3 Kinect/Robot repeatability results

	Average	Range	Std. Dev.
Match value	0.969219	0.013736	0.003928
Device no.	3	0	0
x origin	205	0	0
y origin	162	0	0
Depth (mm)	490.005	1.758	0.447415

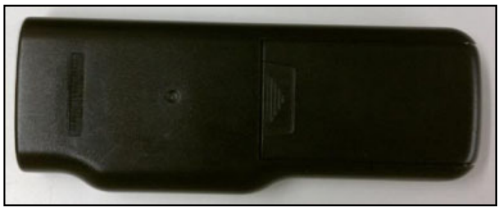


Fig. 22 Internal snap-fit—DVD remote

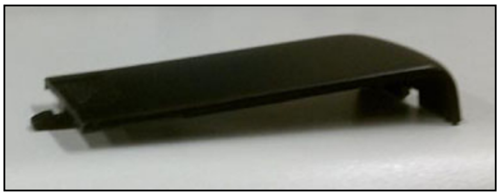


Fig. 23 Internal snap-fit cover

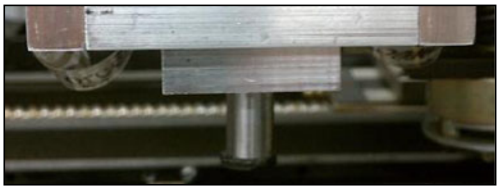


Fig. 24 Internal snap-fit tool tip

In order to increase the universalism of the disassembly system, an additional tool tip was designed to handle these types of snap-fit covers. The original tool tip was designed to screw into the piston assembly, so that additional tool tip designs could be quickly interchanged and tested. An image of the new tool tip mounted on the disassembly tool is shown in Fig. 24. Rather than the tapered design of the original tool tip, the new tool tip consists of a 3/8-in. diameter cylinder with a rubber pad attached to the surface. The increased surface area of the tool tip combined with the rubber pad provides the traction force necessary to slide and release the internal snap-fit cover.

In using this tool, the tool tip is moved to the *x*–*y* position where the snap-fit is located inside the cover. The tool tip is then moved down in the *z*-axis until the vertical force sensor indicates that the tool tip is pressing down on the snap-fit cover. Then the disassembly tool is moved in the appropriate direction along the *x*-axis to slide the cover and release the internal snap-fit. Once the snap-fit is released, the disassembly tool is moved up on the *z*-axis to the zero position. The vacuum gripper is then used to retrieve and dispose of the snap-fit cover using the same strategy as in the original disassembly routine.

8 Conclusions

In this paper, a prototype automated disassembly system was developed to handle the removal of the snap-fit cover and the batteries contained within a family of electronic devices whose plastic, cantilever snap-fit covers house AA or AAA batteries. A disassembly tool equipped with built-in force-sensing resistors was designed and developed for this application. The disassembly tool was mounted on the tool head of a three-axis translational motion robot. A pneumatically actuated vacuum gripper was also mounted on the tool head module for retrieving and disposing of the snap-fit cover. Additionally, a pneumatically actuated electromagnet equipped with an FSR for sensing contact forces was mounted on the tool head module for recovering the batteries. A vision system that uses a Kinect sensor was integrated onto the robot and enhanced the flexibility of the automated disassembly system. The computer vision application recognized the loaded electronic device and its orientation by utilizing a data bank of device models. After gathering information from the computer vision application, the robot was able to use the disassembly tool and recovery modules to perform the necessary disassembly operations to remove the snap-fit and batteries held within the device.

The operation of the disassembly system was successfully tested on a TV remote control unit and a TI-84 scientific calculator which were used as test samples. The TV remote contains two AA batteries, whereas the scientific calculator contains four AAA batteries. Each device was tested for the following orientations and positions: left orientation, right orientation, an *x*–*y* position shift, and a *z*-axis height shift. Based on the success of the disassembly routine and the force sensor results for these test variations, it was concluded that the disassembly tool was able to react to forces applied at the tool tip and accomplish the required disassembly operations. Furthermore, the flexibility of the disassembly tool was also illustrated by the design of the additional tool tip used to successfully release internal snap-fit covers. The interchangeable tool tip designs could be further explored to enhance the disassembly tool's capabilities in separating different types of component mates.

The knowledge gained through this study could be applied to creating force-based disassembly tools capable of removing batteries from other electronic devices such as cell phones and for the removal of exterior plastic panels from larger, more complex devices. The design of such disassembly tools could aid in the future development of fully autonomous disassembly systems.

It should be noted that the current disassembly system is limited by the range of products it can handle and its ability to identify defects in the incoming products. This leads into some of the limitations of the computer vision application that require improvement. The template matching function is not robust against variations in the condition of the incoming devices. For instance, if there is a sticker on the device or the battery cover is missing, the current vision system would not be able to recognize these circumstances. The computer vision application needs more advanced image processing algorithms to detect and adapt to these conditions.

Acknowledgments We would like to acknowledge Dave Ferreira for his guidance in designing and machining the device prototype, as well as James Byrnes for his support in integrating the electronic and pneumatic systems.

References

1. Widmer R, Oswald-Krapf H, Sinha-Khetriwal D, Schnellmann M, Böni H (2005) Global perspectives on e-waste. Environ Impact Assess Rev 25(5):436–458
2. Smith T, Director S (2003) Corporate strategies for electronics recycling: a tale of two systems. Silicon Valley Toxics Coalition, San Jose
3. Ryan C, Gardner J (2011) Metech Recycling, Worcester, MA facility tour and interview
4. Williams JAS (2006) A review of electronics demanufacturing processes. Resour Conserv Recycl 47(3):195–208
5. Masanet E, Auer R, Tsuda D, Barillot T, Baynes A (2002) An assessment and prioritization of design for recycling guidelines for plastic components. In: 2002 I.E. international symposium on electronics and the environment, pp. 5–10
6. Rebafka U, Seliger G, Stenzel A, Zuo B (2001) Process model-based development of disassembly tools. Proceedings of the Institution of Mechanical Engineers, pp. 711–722
7. Park J, Kim G (2005) Development of the 6-axis force/moment sensor for an intelligent robot's gripper. Sens Actuators A Phys 118(1):127–134
8. Feldmann K, Trautner S, Meedt O (1998) Innovative disassembly strategies based on flexible partial destructive tools. In: Proceedings of the IFAC intelligent assembly and disassembly, pp. 1–6
9. Scholz-Reiter B, Scharke H, Hucht A (1999) Flexible robot-based disassembly cell for obsolete TV-sets and monitors. Robot and Comput Integr Manuf 15:247–255
10. Seliger G, Basdere B, Keil T, Rebafka U (2002) Innovative processes and tools for disassembly. CIRP Ann Manuf Technol 51(1):37–40
11. Weigl-Seitz K, Hohm M, Seitz M, Tolle H (2006) On strategies and solutions for automated disassembly of electronic devices. Int J Adv Manuf Technol 30:561–573
12. Torres F, Gil P, Puente S, Pomares J, Aracil R (2004) Automatic PC disassembly for component recovery. Int J Adv Manuf Technol 23:39–46
13. Basdere B, Seliger G (2003) Disassembly factories for electrical and electronic products to recover resources in product and material cycles. Environ Sci Technol 37(23):5354–5362
14. Kopacek B, Kopacek P (1999) Intelligent disassembly of electronic equipment. Annu Rev Control 23:165–170
15. Schumacher P (2012) Design of a tool for automated disassembly of electronic devices. MS Thesis, University of Rhode Island
16. Jouaneh M (2013) Fundamentals of mechatronics. Cengage Learning, Stamford
17. OpenCV Wiki (2012) http://opencv.willowgarage.com/wiki/. (Accessed 21 May 2012).