Abstract

Several hospitals face nurse staffing shortages for surgeries. This research focuses on building a system with a Baxter robot capable of identifying surgical tools using computer vision and delivering them to the surgeon on demand. This would deal with the issue of nurse unavailability during simple surgical procedures. The key aspects of the project were testing the accuracies of various artificial neural networks (ANNs) in classifying surgical instruments and programming Baxter to implement a surgical tool delivery system using magnets at the tip of its 7 degrees of freedom (DOF) robotic arms. The methodology consisted of first implementing algorithms to enable Baxter to do pick and deliver tasks for surgical tools, and second, gathering HuMoments of various tools using the cameras on Baxter’s arm, which were then used to train the ANNs. Tool detection accuracies of ANNs with hidden layer neuron number varying from 5–50 and learning rates varying from 0.005–0.1 were collected. Then, the tool identification and tool delivery system were merged together to create a turn-by-turn dynamic tool tracking and delivery system, which retrieved tools, based on the surgeons input, through a Leap Motion Controller. In addition to delivery, the system was modified to retrieve used tools from the surgeon, using a computer vision-based approach. The optimal ANN configuration consisted of an ensemble of various ANNs working together and achieved a detection accuracy of 93%. The average time taken for a mock abdominal incision surgery with the system is expected to be around 10 minutes and 30 seconds.


Keywords

computer vision, neural networks, Baxter, intelligent systems, surgical tool detection, machine learning, Robot Operating System (ROS), tool classification, multi-ANN systems, gesture-based input, surgical tool delivery
INTRODUCTION

In recent years, hospitals have started facing a dearth of nursing staff (Janes, 2014). In numerous medical workplaces, an adequate number of nurses are unavailable to cater to the number of surgeries performed, which leads to critical delays in making proper treatment available to patients. Overwork and understaffing leads to nursing errors, which can be avoided by using alternative measures.

Robots, since the information revolution of the 1980s, have made a profound impact on industry and revolutionized the ways products are manufactured and assembled, leading to a culture of mass production. Lately, robots have been found to be a viable alternative to manpower in diverse areas of medical interest. Currently, most of the applications focus on remote control of robots by expert surgeons—a field of ever-increasing applicability—called remote surgery (Ayache, Delingette, Golland, & Mori, 2012). Technological breakthroughs like the Da Vinci robot have made remote surgery both possible and viable, and are being used by a number of medical facilities around the world.

In our effort to address the nurse shortage problem, it was natural to consider the possibility of using robots. Consequently, this research project took a different approach to the applicability of robots during surgery. A big part of a nurse’s job description involves making the proper tools available to the surgeon at the right time, enabling the surgeon to devote complete attention to the actual task at hand. This research focused on developing a system that was capable of proper tool delivery to surgeons, on demand, and retrieval of the tools after use.

In the course of the research, a surgical tool tracking and delivery system capable of identifying surgical tools placed randomly on a table using ANNs (Marsland, 2009) was implemented with Baxter, a research robot manufactured by Rethink Robotics. The system was capable of delivering tools to the surgeon on demand and retrieving tools back after use. The tools were tracked at regular intervals over the entire period of the surgery, enabling the robot to work in a more realistic surgery scenario, where the conditions are chaotic and the position of objects cannot be assumed to be constant.

The following sections take a thorough look at the composition of the subsystems comprising the final system, the role they play in the final system, and the calibration processes necessary to ensure seamless operation of the system. Thereafter, the results from the initial system are presented, along with an analysis of the preliminary results. Based on the initial analysis, modifications and further improvements are discussed (see Figure 1).

Figure 1. Final system layout including system components.
**METHODS**

Baxter works on the Robot Operating System (ROS) framework designed by Willow Garage. As stated on the official ROS website, ROS is a collection of tools and libraries that make writing software for robotic systems simpler and more efficient. Programs that utilize the ROS framework can be programmed in C++ and Python, and usually need an Ubuntu system to function. For this project, Python, along with an Ubuntu workstation running Precise Pangolin (Ubuntu 12.04), was chosen to design the system. Baxter communicated with the workstation through the ROS framework on the Ubuntu machine, which in turn was controlled by the Python (Oliphant, 2007) programs.

The final system was capable of receiving user input for the selection of a tool. The robot identified the tool selected using computer vision, figured out a path to pick the tool up, and delivered it to the surgeon. After use, the robot retrieved the tool and stored it away from the surgery. Accordingly, the system consisted of three subsystems working in unison: tool request, tool identification, and tool delivery/retrieval. Calibration programs were used to smoothly transition through the two coordinate frames used in the system: Baxter 3-D world space and 2-D image coordinates. Figure 2 depicts the final system and its subsystems.

**Tool Request**

Traditionally, human-computer interaction has been dominated by serial lines of communication using a mouse or a keyboard. During a surgery, either method is ineffective. A more intuitive and natural method of input was sought so that surgeons can concentrate on the surgery, rather than deviate their attention on actual tool request. Motivated by the goal of keeping surgeon-system interaction as natural as possible, and guided by the observation that surgeons communicate with nurses using hand gestures, a hand gesture-based tool request system was agreed upon. A Leap Motion Controller was used to track the surgeon’s hands during surgery. The sensor communicated with the Ubuntu workstation through a USB port and provided real-time 3-D data about the surgeon’s hand position, orientation, finger count, finger joint velocities, and so forth.

![Figure 2. Final system process flow along with subsystems.](image-url)
The finger count data was used to differentiate between individual tools. As the finger count can range from 0 through 5, for testing purposes a predetermined set of 6 tools was selected. Each tool corresponded to a specific finger count. The set of tools along with their corresponding finger count are listed in Table 1. The finger count of the number also known as the Unique Tool ID, was used to represent the tool internally in the Python scripts.

<table>
<thead>
<tr>
<th>Finger Count</th>
<th>Tool</th>
<th>Unique Tool ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scalpel</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Retractor</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Hemostat</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Scissors</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Hook</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Needle</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 1. Tool set and unique tool identification of each tool.

The data obtained from the Leap Motion Controller was further utilized to keep track of the occurrence of a simple key-tap gesture needed for temporal segmentation. The finger count was recorded continually, regardless of the surgeon’s intent to request a tool. Therefore, the key-tap gesture was adopted as a temporal segmentation technique to initiate the tool request sequence. Whenever the surgeon wanted a tool, he/she, would perform a key-tap gesture and then extend the appropriate number of fingers corresponding to the specific tool choice. The system would then pass the Unique Tool ID to the system for tool identification and subsequent delivery to the surgeon.

**Tool Delivery**

Baxter’s left arm rested in a neutral position directly above the tool retrieval location when not delivering a tool. The delivery sequence was initiated by the tool identification system, which provided the tool delivery system with the 2-D pixel coordinates of the centroid of the requested tool in the image captured by Baxter’s right arm. The 2-D image coordinates were converted to 3-D Baxter space coordinates using the rotation and translation matrix obtained from calibration process (described later). The 3-D Baxter world space coordinates were provided to the inbuilt IKFast in Baxter. IKFast also received the quaternion representation of the orientation of the tip of the delivery arm from the orientation control system. The algorithm finally output the joint space values for the delivery arm of Baxter, which were used by ROS to actually move the arm. See Figure 2 for reference.

Upon being triggered, the left delivery arm was moved to the center of the surgical table in a quick motion and was slowly lowered onto the tool. The tool was picked up by the magnet on contact and then the arm was moved to the delivery location, which was prespecified as a particular set of joint space values. After delivery, the arm moved back to the neutral position and monitoring of the retrieval zone resumed.

**Tool Retrieval**

The camera at the bottom plane of Baxter’s delivery arm was used to track the retrieval zone. In a process similar to delivery, on being triggered the delivery arm moved to the centroid of the tool. The tool was picked up using the magnet at the tip and then the arm moved to the retrieval box, placed away from the surgical area. The tool was lowered into the box though a linear slit on the top surface of the box. Once inside, the orientation of the arm changed so that as the arm moved away, the tool was held back in the box by the flaps at the top surface. After retrieval, the arm moved back to the neutral position.

**Orientation Control**

During operation, the orientation of Baxter’s arm was controlled using the orientation control system. Although Baxter ships with an in-built IKFast Solver, quaternions are used to specify orientation, which can be counterintuitive...
to the human mind. Hence, a program was devised that takes as input a unit vector in 3-D space, which specifies the orientation. The program output the necessary quaternion orientation of the unit vector, which was directly used in the IKFast Solver afterward. As a result, Baxter’s arm could move to a specific 3-D position and point in a specific direction while at that position. The 3-D unit vector was converted to Euler angles using the following equations:

$$\Theta = \cos^{-1} \frac{V_z}{|V|}$$

$$\Phi = \sin^{-1} \frac{V_z}{|V| \times \Theta}$$

As a point of interest, $\Gamma$ was independent of the unit vector used to specify the orientation, as it represented the rotation of the arm around the axis passing through Baxter’s arm. (The gamma value was manipulated during disposal to ensure that the tool was oriented perpendicular to the slit in the box). Rotation matrices $R_x$, $R_y$, and $R_z$, were constructed out of the Euler angles, as the angles $\Theta$, $\Phi$, and $\Gamma$ specified rotation around the X, Y, and Z axes, respectively.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \Theta & -\sin \Theta \\ 0 & \sin \Theta & \cos \Theta \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos \Phi & 0 & \sin \Phi \\ 0 & 1 & 0 \\ -\sin \Phi & 0 & \cos \Phi \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos \Gamma & 0 & \sin \Gamma \\ 0 & \cos \Gamma & -\sin \Gamma \\ 0 & \sin \Gamma & \cos \Gamma \end{bmatrix}$$

These matrices were concatenated to form the final rotation matrix $R_{\text{final}}$.

$$R_{\text{final}} = R_x \times R_y \times R_z$$

Finally, the quaternion representation was calculated from $R^*_{\text{final}}$.

**Tool Identification**

The tool identification process is the heart of the system. The right arm of Baxter, which is not used for delivery, has a camera at the bottom plane and is used for capturing images. After mapping the tools present on the surgical area, pixel coordinates of the centroids of these tools are obtained. These pixel coordinates are converted into 3-D Baxter space coordinates, using the transformation matrix obtained during the calibration process described later. For tool identification, we chose to use ANNs as the base classifier and tested their efficiency in a variety of scenarios. All image processing algorithms were implemented using OpenCV (Bradski, 2000), because it is an open source, free, and a potent and robust library for image processing. It also comes with a subset of tools required for the system, which is inbuilt into the library. A breakdown of the steps involved follows.

**Obtain Image**

Baxter’s arm was positioned at a prespecified height above the table and an image of the surgical table was obtained using the camera at the bottom plane of the arm. The height above the table was experimentally optimized to ensure that the table completely filled the image. The images obtained were of resolution 480 x 640 pixels (see Figure 3).

![Figure 3. Original image obtained through left arm camera of Baxter: scissors (left), hook (middle), and hemostat (right).](image)

**Preprocess Image**

The tool table had four dots at the corners of the table. After obtaining the image, it was clipped at these four points to ensure that the image was only comprised of the table surface and tools. This removed undesirable borders from the image. As the tools were metallic, they exhibited luster on their surface, which lead to poor detection. To reduce the shininess of the metals, the clipped image was passed through a low-pass Gaussian blur filter, with a disc kernel of size 5. This effectively removed luster heat spots on the tools.

**Foreground (Tool) Extraction**

In this step, the preprocessed image was converted into a binary image of a white tool foreground against a black background. A variety of background extraction techniques were tested to obtain the appropriate binary image, including Gaussian Mixture Models, HSV back-projection (Swain & Ballard, 1990), Otsu’s binarization (1979), and HSV thresholding. HSV thresholding was chosen as the final extraction technique as the saturation minimum (smin) value of the threshold limit could be varied to accommodate illumination changes during system operation. After obtaining the image, various smin values were tested and a final smin was chosen based on a criteria set including size, shape, number, and area of tools detected. HSV thresholding actually subtracted the tools from the image. Hence, the image was inverted to obtain the desired binary image.

**Postprocess Image**

The binary image was passed through a median filter to remove salt and pepper noise. The image was further subjected through morphological closing to fill in the remaining small black holes in the tool body. The image was then eroded to remove the last of the white spots in the image (see Figure 4).
Find Contours

The postprocessed image was passed through OpenCV’s findContours function, based on Suzuki and Be’s border-following algorithm (1985). A list of contours of the white blobs that represented the different tools in the image was obtained. The contours were lists of 2-D pixel coordinates. Properties of these contours including arc length, hierarchy index, and contour area were calculated using OpenCV (see Figure 5).

Filter Contours

The initial list of contours contained contours other than that of the tools. To screen these unwanted contours, two criteria were used. First, only base contours with no parents (enclosing contours) were selected. This was based on the hierarchy list of contours provided by OpenCV, and hence, only contours with parent ID of -1 were selected. Further, only contours with a boundary perimeter greater than 400 were chosen for subsequent analysis. This ensured that smaller contours of holes in the background were not identified as tools.

Calculate Modified HuMoments

To increase the speed of the identification process and reduce training time for the neural networks used for identification, instead of working directly with the contours, 2-D shape descriptors were used to represent the contours. HuMoments were chosen to represent the homomorphism between the 2-D contour points and the shape descriptors. HuMoments are a set of seven numbers invariant to translation scaling and rotation in a plane. They are used to represent a 2-D contour or image in a succinct manner.

Due to the invariability of the HuMoments to rotation, translation, and scaling, the tool identification process was unaffected by the location, size, and orientation of the tools (Huang & Leng, 2010). However, as proposed by Flusser and Suk (2006), the set of seven HuMoments is incomplete, and following their work an eighth moment was included in the set. Thereafter, each tool was represented by 8 HuMoment Invariants. Figure 6 depicts the calculation of HuMoments from the scale of invariant moments represented by η.

\[
\begin{align*}
I_1 &= \eta_{20} + \eta_{02} \\
I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
I_3 &= [\eta_{20} - 3\eta_{02}]^2 + (3\eta_{11} - \eta_{00})^2 \\
I_4 &= (\eta_{20} + \eta_{02})^2 + (\eta_{11} + \eta_{00})^2 \\
I_5 &= (\eta_{20} - 3\eta_{02})(\eta_{02} + \eta_{02})[(\eta_{02} + \eta_{02})^2 - 3(\eta_{11} + \eta_{00})^2] \\
&+ (3\eta_{11} - \eta_{00})(\eta_{11} + \eta_{00})[3(\eta_{02} + \eta_{02})^2 - (\eta_{11} + \eta_{00})^2] \\
I_6 &= [\eta_{20} - \eta_{02}][(\eta_{02} + \eta_{02})^2 - (\eta_{11} + \eta_{00})^2] + 4\eta_{11}(\eta_{02} + \eta_{02})(\eta_{02} + \eta_{02}) \\
I_7 &= (\eta_{20} + \eta_{02})(\eta_{02} + \eta_{02})[(\eta_{02} + \eta_{02})^2 - 3(\eta_{11} + \eta_{00})^2] \\
&- (\eta_{11} - 3\eta_{00})(\eta_{11} + \eta_{00})[3(\eta_{02} + \eta_{02})^2 - (\eta_{11} + \eta_{00})^2] \\
I_8 &= \eta_{11}[(\eta_{02} + \eta_{02})^2 - (\eta_{00} + \eta_{00})^2] - (\eta_{20} - \eta_{02})(\eta_{02} + \eta_{02})(\eta_{02} + \eta_{02})
\end{align*}
\]

Pass to ANN and Identify Tool

Neural networks were used to classify the HuMoments obtained and identify different tools. The set of 8 HuMoments was passed to the ANN as input, and finally the network classified the tool into one of the six tool categories mentioned earlier. The details of the ANN configuration are described in the next section (see Figure 7).

Neural Network Configuration and Training

The ANNs used in the system were simple three-layer networks (Mitchell, 1997) including the input layer, hidden layer, and output layer, that used backpropagation for learning (Nielsen, 2014). Various configurations were tested to gather a sufficiently large base of neural networks to examine identification accuracy. Each data point used in the process was a set of 8 HuMoments representing a specific tool. The gathered data was split into different sets for proper training, validation, and testing. Table 2 presents the number of data points used for each case.
ANN Configuration

The input layer of the neural networks had 8 inputs corresponding to the 8 HuMoments representing a tool. The number of hidden layer neurons was varied. Each layer was interconnected by a fully connected sigmoid layer. The output layer consisted of 6 neurons. Figure 8 shows a sample ANN.

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Data Set Type</th>
<th>Unique Tool ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Training</td>
<td>240</td>
</tr>
<tr>
<td>2</td>
<td>Validation</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>Single-ANN System Testing</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>Multi-ANN System Testing</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2. Number of data points used for each individual process.

Ideally, each neuron was supposed to be a binary number 0 or 1, representing whether the input tool was a tool with ID 0, 1, 2, and so forth. For example, in case of a retractor that had Unique Tool ID 1, the ideal output would look like:

\[ [0,1,0,0,0,0] \]

However, as neural network outputs were real numbers, the maximum of the six output values was given a value 1 and the other five were assigned 0. For example, if the output layer was \([0.2,0.88,0.1,0.05,0.3,0.1]\), it was transformed to \([0,1,0,0,0,0]\). The maximum value occurred at position 1 (if index started at 0). Hence, the Tool ID was 1, so the neuron identified the set of HuMoments as a retractor. The vectors were internally represented using NumPy (Van der Walt, Colbert, & Varoquaux, 2011) arrays.

Choice of Parameters

The number of hidden layer neurons and the learning rate of training were varied to test different ANN configurations. The number of hidden layer neurons was varied from 5–50 units in increments of 5 units, while the learning rate was varied from 0.005 to 0.100 in increments of 0.005. To include three orders of variation in the learning rate possibilities, 0.001 also was included. This resulted in a total of 210 (21 x 10) possible neural network configurations. Figure 9 depicts the visualization of the ANN as a black box with two varying parameters.

Computer Implementation

The entire process was implemented in Python. PyBrain, a machine learning library was used for training and testing purposes (Schaul et al., 2010). Training was terminated either at convergence or a maximum iteration count of 10,000.

Tool Presence Identification for Retrieval

Tool presence identification, as mentioned, was used to trigger the tool retrieval action. It was implemented using the camera on the delivery arm of Baxter. When at the neutral position, the arm pointed directly down at the retrieval table. HSV back-projection was used to obtain a binary image of the retrieval zone. In case of the absence of a tool, the binary image obtained was mostly black. In case of the presence of a tool, the white area in the binary image representing the tool increased. A threshold of 12.5% white area was selected as the trigger for the retrieval action. Once the retrieval action had been triggered twice within 10 seconds, the retrieval sequence was initiated.

Calibration and Change of Coordinates

A calibration process was used to calculate the transformation from 2-D pixel coordinates to 3-D Baxter world space coordinates. In the process, the arm of Baxter was moved to four points located on the corners of the tool table. These points were simultaneously highlighted on the image obtained.
from Baxter using a mouse. Each measurement resulted in a set of 3-D Baxter world space coordinates (represented by $B_x$, $B_y$, and $B_z$) along with a set of 2-D image coordinates ($M_x$ and $M_y$). Motivated by Kabsch’s (1976) approach of using a matrix to relate two sets of vector, a transformation matrix ($T$) was calculated from three out of the four measurements. The matrix was then used to map the 2-D tool center pixel coordinates in the image to actual 3-D locations where the arm could move (Arun, Huang, & Blostein, 1987).

$$M = \begin{bmatrix} M_{x1} & M_{x2} & M_{x3} \\ M_{y1} & M_{y2} & M_{y3} \end{bmatrix}, B = \begin{bmatrix} B_{x1} & B_{x2} & B_{x3} \\ B_{y1} & B_{y2} & B_{y3} \end{bmatrix}, B = \begin{bmatrix} B_{z1} & B_{z2} & B_{z3} \end{bmatrix}$$

$$T = B \times M^{-1}$$

**RESULTS**

**Initial Results**

The identification accuracy of each ANN configuration was tested on the test data set using receiver operating characteristics (ROC) analysis (Fawcett, 2006). The results were combined and plotted with the learning rate constant on each plot, while the hidden layer neuron number was varied as the independent variable. Figure 10 shows the results for a subset of the ANN configurations.

Initial analysis showed that maximum accuracy was achieved for the neural network possessing 10 neurons in the hidden layer, trained at a learning rate of 0.025 (see Figure 11).

After further analysis using visualization libraries like matplotlib (Hunter, 2007), it was found that the tools could be classified into two groups based on 2-D similarities in the contours of the tools. Group 1 comprised of the scalpel, hook, and needle, while Group 2 included the retractor, hemostat, and scissors. Figure 12 shows the similarities in contours of Group 1 tools. Similarly, the ANNs could be classified into two groups based on the group they were better at classifying. Group 1 ANNs were better at differentiating between Group 1 tools, while Group 2 ANNs similarly had higher accuracy in classifying Group 2 tools. Figure 13 depicts the classification trends.

**Modification Based on Analysis**

To incorporate the results of the analysis mentioned above, a multi-ANN identification system was adopted. Three ANNs with the highest identification accuracy from

---

**Figure 10.** Identification accuracy of various learning rates. Hidden layer neuron number is spotted on the X-axis while the accuracy percentage is on the Y-axis.
each group were combined together to form an ensemble detection system. Each of the six ANNs output what they thought the tool was along with a confidence measure. The output of the ANN with the highest confusion measure was selected to be the final classification. Figure 14 shows the new multi-ANN system, while Figure 15 depicts how the confidence measure was calculated from the output of the ANN. The multi-ANN approach increased identification accuracy to 93%.

**Confusion Matrix**

A confusion matrix was constructed to understand the identification patterns. It was found that in numerous cases, a needle was classified as a scalpel. It was suggested that the confusion could be attributed to the similarity in the 2-D outline of needles and scalpels (see Figure 16). Figure 17 shows the final confusion matrix.
Figure 16. Top image shows original image of needle (left) and scalpel (right). Bottom image shows the similarity in contours of the above-mentioned tools.

Figure 17. Confusion matrix corresponding to ANN classification of tools. Rows indicate actual tool type, while columns indicate classification by ANN. Example: Out of the 17 needle data points, 6 get classified as scalpels, one as a hook, and ten as needles. This indicates high scalpel-needle confusion.

CONCLUDING REMARKS AND FUTURE WORK

Efficiency of System

As of now, the system is only a first prototype. At times during arm movement, the table was dislodged from its position, which hints at the need of an independent path planning system. We hope to build on this work and improve the system to an extent where it is capable of being used in a real-life surgical scenario in an error-free manner.

Possible Areas of Future Work

Future work in this area will attempt to improve needle-scalpel differentiation, adopt a mosaic approach to increase work area, implement surgeon hand tracking for effective delivery, develop a tool orientation control to manage sharp edges of tools, and use an electromagnet to deliver and retrieve tools.

ACKNOWLEDGMENTS

This publication was made possible by the NPRP award (NPRP 6-449-2-181) from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the author.

I would like to express my deep gratitude to my faculty advisor Professor Juan Pablo Wachs in guiding me through this project. I would also like to thank all the members of the ISAT Laboratory, especially my graduate mentor Maria Eugenia Cabrera, for her knowledge and the hours spent in deliberation, trying to find an optimal solution to a problem. Finally, I am grateful to SURF 2014 for the opportunity to work on this challenging and exciting project.

REFERENCES


