Using Kinect™ and a Haptic Interface for Implementation of Real-Time Virtual Fixtures

Fredrik Rydén, Howard Jay Chizeck, Sina Nia Kosari, Hawkeye King and Blake Hannaford

***Abstract*—The use of haptic virtual fixtures is a potential tool to improve the safety of robotic and telerobotic surgery. They can "push back" on the surgeon to prevent unintended surgical tool movements into protected zones. Previous work has suggested generating virtual fixtures from preoperative images like CT scans. However these are difficult to establish and register in dynamic environments. This paper demonstrates automatic generation of real-time haptic virtual fixtures using a low cost Xbox Kinect™ depth camera connected to a virtual environment. This allows generation of virtual fixtures and calculation of haptic forces, which are then passed on to a haptic device. This paper demonstrates that haptic forces can be successfully rendered from real-time environments containing both non-moving and moving objects. This approach has the potential to generate virtual fixtures from the patient in real-time during robotic surgery.**

I. Introduction

This paper presents a potential method to improve the safety and efficacy of robotic and telerobotic surgery. Provision of haptic feedback to the surgeon, so that a "sense of touch" is available to assist the surgeon, is an ongoing topic of research at several labs. Of particular interest are methods to constrain robot tool movements and to provide the surgeon with haptic indication of "don't cut" zones. Such constraints are often called virtual fixtures [10]. They have been discussed in the context of telerobotics and surgical robotics for the past two decades.

Abbott et al. [1] suggests that Virtual Fixtures can be divided into two categories: *Guidance Virtual Fixtures*, which guide the operator of the haptic device along a specified path; and *Forbidden-Region Virtual Fixtures* that "push back" on unintended movements into certain protected zones.

In this paper, we propose a novel method for the automatic generation of Forbidden-Region Virtual Fixtures, based upon images obtained from a depth camera.

Virtual Fixtures for telerobotics are most commonly specified as simple geometric shapes. However, for use in surgical robotics, these fixtures may need to represent (or be derived from) portions of the patient's anatomy. Thus a challenge arises: How can virtual fixtures be appropriately specified, relative to the patient—and can this be done in real time, so as to compensate for movements and deformations during a surgical procedure?

Li et al. [7] suggest that shapes generated from pre-operative CT scans (via 3D Slicer [3]) can be used as virtual fixtures. This approach can work for surgery when there is little or no movement, if the CT scan can be adequately registered to the patient. However, when robotically-assisted minimally invasive surgery results in relative motion of different organs and tissues, then this use of pre-operative imagery is problematical. Repeating CT scans during the surgical procedure presents logistical complications, as well as increased radiation exposure to the patient.

A solution to this problem is to generate haptic virtual fixtures from depth camera imagery in real-time. Haptic rendering from RGB-D data was first investigated by Cha et al. [4]. Their method requires pre-processing of recorded camera data and therefore is not suitable for real-time rendering of dynamic environments.

This paper demonstrates generation of real-time haptic virtual fixtures using Xbox Kinect, a low cost device developed for the gaming purposes.

In Section II depth cameras and RGB-D cameras are discussed. In III the implementation of the proposed approach is presented. Experimental evaluations are contained in Section IV. Finally, Section V provides suggestions for future work.

II. Depth Cameras

RGB-D cameras specify color information for each pixel and also specify an estimated distance from the camera to the pixel. This depth estimation is most commonly calculated using either time-of-flight, active stereo or projected patterns.

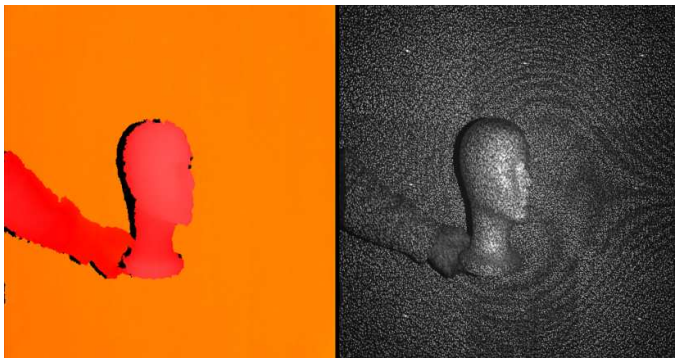

Fig. 1. IR dot pattern of a scene with a Styrofoam head held up against a plain surface (Right). Color representation of depth estimation (Left). Note the black parts of the image where there are no projected dots and hence no depth estimation.

The Xbox Kinect is a low cost RGB-D camera (with an approximate retail price of US$150) that has been developed for use with video games. The Kinect consists of one infrared (IR) projector, one IR camera and one regular RGB camera. The IR projector emits a dot pattern, which is known for a certain depth, into the room (see Figure 1). The IR camera

then captures reflections of this pattern from objects in the room. This is done at 30 frames per second. A triangulation is then performed on the dots, which results in estimates of the depth value for each pixel in a 640x480 matrix (with depth estimates taking values between 0 and 2047 units). [6] [5]

At $2m$ distance the depth resolution is $1cm$ and the horizontal/vertical resolution is $3mm$. The horizontal field of view is $H_{FOV} = 58°$ and the vertical field of view is $V_{FOV} = 45°$. The operating range is $0.8 - 3.5m$. [8]

III. Implementation

A. System Configuration

A diagram of the system is shown in Figure 2. It consists of three subsystems:

1) The depth camera: The Kinect RGB-D camera collects depth information from the physical world and sends it to a PC.

2) PC running a virtual environment: The Kinect camera is connected to a PC running a virtual environment developed by the authors that uses the open source OpenKinect driver (source downloaded 12/08/2010, compiled on Ubuntu 10.10 64 bit). This virtual environment visualizes the depth map as a point cloud and the Haptic Interaction Point (HIP) as a sphere. The haptic forces are rendered from this information. The forces are re-calculated 1000 times per second.

This PC is connected via a UDP network socket (represented by the vertical dotted line in Figure 2), to a second PC which is connected to a haptic device. Haptic force information (derived from the Kinect) is sent outwards over the network, and haptic tool tip movement information is received from the network.

3) PC connected to haptic device: This PC is an interface for the Phantom Omni® (SensAble Technologies Inc., MA, USA) haptic device to receive forces and send movements for the HIP. The OpenHaptics™API is used for communication between the haptic device and the PC.

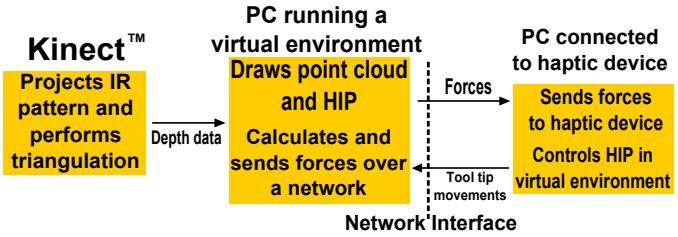

Fig. 2. System Configuration for generation of real-time haptic virtual fixtures

B. Interpretation of depth data

The Kinect camera returns depth data in a 640x480 matrix $M(u, v)$ where each element corresponds to a pixel with a depth value ranging between 0 and 2047. The depth data is then transformed (see Equation 1), using a homogeneous transform, to a Cartesian coordinate system where 1 unit corresponds to 1 meter.

$$u \in [0, 639], v \in [0, 479], d \in [0, 2047]$$

$$\begin{pmatrix} u & v & d & 1 \end{pmatrix} \begin{pmatrix} 1/f_x & 0 & 0 & 0 \\ 0 & -1/f_y & 0 & 0 \\ 0 & 0 & 0 & a \\ -c_x/f_x & c_y/f_y & -1 & b \end{pmatrix}$$

$$= \begin{pmatrix} x' & y' & z' & w \end{pmatrix} \quad (1)$$

$$= w \begin{pmatrix} x & y & z & 1 \end{pmatrix} \quad (2)$$

The parameters in Equation 1 are estimated values for the camera intrinsics (f_x, f_y, a are scaling factors and c_x, c_y, b are offset constants) obtained from calibration by Burrus [2] (see Table I below).

TABLE I
Parameters used in transformation

f_x	594.214342
f_y	591.040537
c_x	339.307810
c_y	242.739138
a	-0.003071102
b	3.330949516

C. Properties of the Point Cloud

The x, y and z values in Equation 2 can be visualized in one image as a point cloud where each point is represented by a small dot.

The maximum number of points in one image are $N_{tot} = 640 \times 480 = 307200$. The point cloud will be denser for short distances and sparser for longer distances. For a given distance z (in meters), the maximum number of points per square meter is given by Equation 5.

$$h = 2z \times \tan(\frac{V_{FOV}}{2}) \quad (3)$$

$$w = 2z \times \tan(\frac{H_{FOV}}{2}) \quad (4)$$

$$\rho_{points} = \frac{N_{tot}}{h \times w} \quad (5)$$

D. Haptic Rendering

Since the haptic forces in this paper are generated from a point cloud and not from a solid mesh, conventional haptic rendering methods based on collision detection cannot be used. McNeely et al. [9] suggested a solution to this problem using the Voxmap Pointshell method, but this solution uses point clouds with pre-computed surface normals.

Instead, for calculation of the forces on the HIP the following algorithm is used, where N_{max}, N_{min}, D_{thresh} and k_s are pre-defined parameters.

1) First we determine N, the number of points involved in the interaction with the HIP. Let N_{thresh} be the number of points within the radius D_{thresh} of $\mathbf{P_{HIP}}$ (see Figure 3). Then, use the following rule:

If

$$N_{thresh} > N_{max} \tag{6}$$

choose the $N = N_{max}$ points closest to $\mathbf{P_{HIP}}$.
If

$$N_{min} \leq N_{thresh} \leq N_{max} \tag{7}$$

choose all points within the radius D_{thresh} of $\mathbf{P_{HIP}}$.
If

$$N_{thresh} < N_{min} \tag{8}$$

do not choose any points. That is, $N = 0$.

2) Designate the chosen points $\mathbf{P_k}, 1 \leq k \leq N$, where N is the number of chosen points.
3) Calculate the vectors from each point $\mathbf{P_k}$ to $\mathbf{P_{HIP}}$.

$$\mathbf{V_k} = \mathbf{P_{HIP}} - \mathbf{P_k} \tag{9}$$

4) Attach a virtual spring force to each point.

$$\mathbf{F_k} = k_s(D_{thresh} - ||\mathbf{V_k}||_2) \times \frac{\mathbf{V_k}}{||\mathbf{V_k}||_2} \tag{10}$$

5) Calculate the total force on the HIP, normalized by the number of chosen points.

$$\mathbf{F_{HIP}} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{F_k} \tag{11}$$

The parameters in the equations above are obtained by experimental evaluation (ie, trial and error).

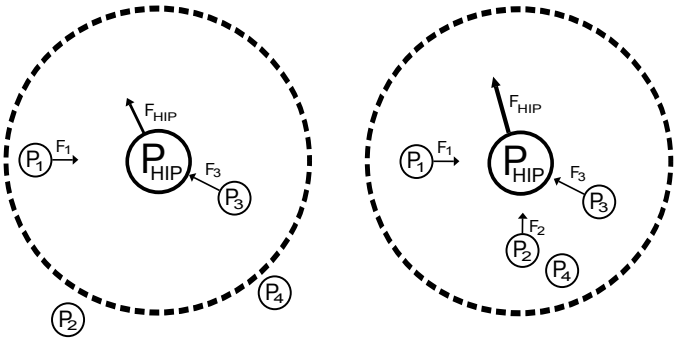

Fig. 3. Illustration of the point cloud around the HIP (Haptic Interaction Point) for two different cases using parameters $N_{min} = 1, N_{max} = 3$. In the first case, $\mathbf{P_2}$ and $\mathbf{P_4}$ are outside the threshold distance and do not affect the HIP (Left). In the second case, all points are within the threshold distance but only the three closest points, $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_3}$, affects the HIP. (Right).

IV. Results

A. Virtual Environment

Figure 4 shows the real image side-by-side with the virtual environment. The point cloud visualizes the depth information captured by the Kinect camera. The point cloud can be rotated and viewed from any angle.

Figure 5 illustrates haptic interaction with a non-moving Styrofoam head on a table. The right side of Figure 5 indicates the force applied to the operator's hand. The red sphere in the point cloud marks the location of the HIP.

Fig. 4. Captured image from the Kinect RGB camera (Left) and point cloud representation of the same scene viewed from a slightly different angle (Right).

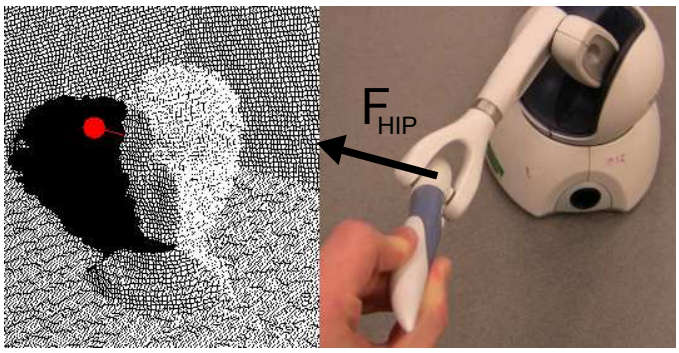

Fig. 5. Closer look at the point cloud representation of the depth map where the red sphere shows the position of the HIP (Left). The haptic device controlling the HIP (Right). F_{HIP} illustrates the force acting on the operator of the haptic device when trying to push the HIP towards the Styrofoam head.

B. Evaluation of Parameters and Haptic Rendering

Choosing the parameters for the haptic rendering is a trade-off problem between haptic resolution, haptic stiffness and haptic stability. A search of N_{max} values (6 values between 50 and 300), k_s values (5 values between 300 and 700) and D_{thresh} (6 values between 0.02 and 0.07) resulted in the selected parameters.

A subjective determination of the "best" parameters was obtained based upon perceived resolution, stiffness and lack of oscillation artifacts. The resulting parameters were $k_s = 500N/m$, $D_{thresh} = 0.04m$, $N_{min} = 10$ and $N_{max} = 100$. This evaluation was performed by one operator. It is operator dependent, since individual sensory capabilities, hand mass and mechanical properties, and preferences will all affect the selection of best parameter values.

The forces F_k in Equation 10 are modelled as spring-forces. Therefore touching a any surface, mapped as a point cloud, will feel like a linear spring.

A sharp 90° edge will feel like a rounded edge with radius D_{thresh}, because the HIP interacts with points within this radius.

A limitation of using this algorithm is that it will be possible to "pop through" a point cloud. This means that the HIP will be able to move through a point cloud if enough force is applied, even if the point cloud represents a rigid object with a very large stiffness.

The theoretical maximum "pop through" force using the parameters above will be $F_{pt-max} = k_s * D_{thresh} = 20N$, for the case with an infinite dense point cloud (no gaps between the points). Naturally, the "pop-through" force F_{pt} will decrease as depth increases and the point cloud gets sparser. F_{pt} will also be lower for complex surfaces with poor depth estimation.

For the point cloud of the table in Figure 4, the "pop through" forces F_{pt} ranges between $12.9N$ at the near-end of the table and $7.5N$ at the far-end.

Some sensor noise on the depth values was observed. This noise had a standard deviation of $0.82mm$ at $0.5m$ depth.

C. Interaction with moving objects

Figure 6 illustrates 3D interaction with a moving virtual fixture, in this case a moving hand. When the hand slowly moves up, the haptic device is forced to move along.

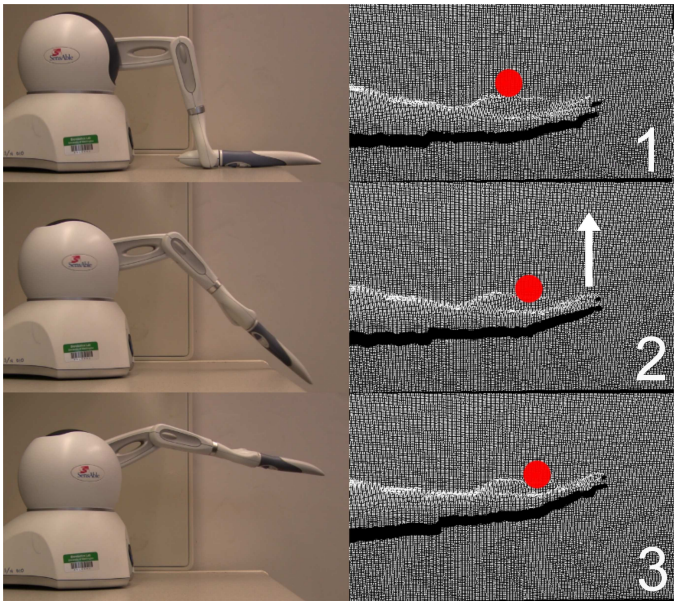

Fig. 6. Starting at rest, the haptic device is forced to move up as the hand moves up. The HIP is enlarged for clarity.

V. Further work

This paper is an initial demonstration of the feasibility of using RGB-D camera images to generate haptic virtual fixtures in real time. Developing this idea into a practical and effective tool for surgical robotics will require a number of improvements and extensions.

One concern is the evident noise in the depth information. There is a need to develop and apply appropriate filtering algorithms to improve the signal-to-noise ratio of depth information.

A second challenge is to enhance the resolution of the depth image. This might be accomplished by modifications of certain physical components of the device. In addition, improvements to the image processing algorithm may address this need.

Incorporation of meshing in the rendering algorithm may significantly enhance the ability to produce rigid objects in the haptic rendering, and will likely improve the transient response properties of the overall system. A related issue is the need for automatic haptic force calibration. Ideally this would be enabled in real time, continuously through the surgery.

A different concern is the need to provide registration of the haptic images with known points in the real environment (such as patient landmarks).

Finally, there is a need to capture the haptic properties of different tissues, and to provide this as haptic feedback to the operator. This might involve synthetic palpation, achieved through small amplitude vibrations of sensor-instrumented surgical tools when in contact with tissue, and processing of the resulting information as a parallel channel in the haptic rendering.

VI. Conclusion

This paper demonstrates the automatic generation of real-time haptic virtual fixtures using a RGB-D camera connected to a virtual environment developed by the authors. It is shown that haptic forces can be successfully rendered from real-time environments containing both moving and non-moving objects. This approach has the potential to generate real-time virtual fixtures from the patient during robotic and telerobotic surgery.

Acknowledgement

This work was funded by the National Science Foundation (NSF grant # 0930930) and by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] J. Abbott, P. Marayong, and A. Okamura. Haptic virtual fixtures for robot-assisted manipulation. *Robotics Research*, pages 49–64, 2007.

[2] N. Burrus. Kinect Calibration, December 2010. http://nicolas.burrus.name/index.php/Research/KinectCalibration.

[3] BWH and 3D Slicer. 3D Slicer. http://www.slicer.org/.

[4] J. Cha, S. Kim, I. Oakley, J. Ryu, and K. Lee. Haptic Interaction with Depth Video Media. *Advances in Mulitmedia Information Processing-PCM 2005*, pages 420–430, 2005.

[5] A. Machline M. Arieli Y. Freedman, B. Shpunt. Depth mapping using projected patterns. US PATENT. PCT No.: PCT/IL08/00458, April 2008.

[6] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments.

[7] M. Li, M. Ishii, and R.H. Taylor. Spatial motion constraints using virtual fixtures generated by anatomy. *Robotics, IEEE Transactions on*, 23(1):4–19, 2007.

[8] Prime Sense LTD. Reference design, January 2011. http://www.primesense.com/?p=514.

[9] W.A. McNeely, K.D. Puterbaugh, and J.J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. *Proceedings of SIGGRAPH, the 26th annual conference on Computer graphics and interactive techniques*, pages 401–408, 1999.

[10] L.B. Rosenberg. Virtual fixtures: Perceptual tools for telerobotic manipulation. In *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, pages 76–82. IEEE, 2002.