A Low-Cost Computer Vision Based Approach for Tracking Surgical Robotic Tools

Rodney Dockter

Timothy M. Kowalewski

Department of Mechanical Engineering,
University of Minnesota

1 Background

The number of Robot-Assisted Minimally Invasive Surgery (RMIS) procedures has grown immensely in recent years. The number of surgeries performed with the da Vinci (Intuitive Surgical, Sunnyvale, CA) worldwide in 2005 was under 50,000. This number grew to more than 350,000 by 2011 [1]. RMIS procedures provide improved patient recovery time and reduced trauma due to smaller incisions relative to traditional open procedures.

Given the rise in RMIS procedures, several organizations and companies have made efforts to develop training and certification criteria for the da Vinci robot. Mimic technologies (Mimic, Seattle, WA) and Intuitive Surgical both produce virtual reality (VR) training simulators based on the da Vinci system. The Fundamentals of Robotic Surgery (FRS) consortium has been created with the goal of developing a standardized, high-stakes certification exam for robotic surgery [1]. While still in the development stage, this exam will consist of seven tasks carried out on a small physical module with an actual RMIS system. Each task is evaluated with a certain set of criteria including completion time, total tool path length and economy of motion, which is a measurement of deviation from an 'ideal' path. All of these metrics can benefit greatly from an accurate, inexpensive and modular tool tracking system that requires no modification to the existing robot.

While the da Vinci uses joint kinematics to calculate the tool tip position and movement internally, this data is not openly available to users. Even if this data was open to researchers, the accuracy of kinematic calculations of end effector position suffers from compliance in the joints and links of the robot as well as finite uncertainties in the sensors. In order to find an accurate, available and low-cost alternative to tool tip localization, we have developed a computer vision based design for surgical tool tracking. Vision systems have the added benefit of being low cost with typical high resolution webcams costing around $50. The stereo setup for this design cost around $120.

Chmarra et al. reviewed the available non-robotic, laparoscopic tracking devices in 2007 and discussed 4 main technologies for tracking; mechanical, visual, ultrasonic, and electromagnetic [2]. From this work, it became apparent that in order to track robotic tools, only the visual or ultrasound-based methods would be feasible. The goal of our research was to develop a tracking system which could accompany the FRS module or be used separately during real procedures to gauge performance post-procedure using the da Vinci camera feed. The system was designed to use only a camera setup and a computer loaded with the computer vision software. Given the bandwidth of surgical tool motions has been experimentally determined as falling below 8 Hz [3], we adopt 16 Hz, or frames per second (FPS), as a minimal frame rate required to accurately capture surgical tool motion data.

Manuscript received March 15, 2013; final manuscript received April 26, 2013; published online July 3, 2013. Assoc. Editor: Arthur G. Erdman.

2 Methods

Our system design consists of both a hardware element and a software element. The hardware element primarily incorporates the video source which will then be analyzed via software. The video source for this algorithm is agnostic to its stereoscopic source: it accepts the DVI output of the cameras on the da Vinci system or alternative sources. For development purposes, two Microsoft Lifecam Studio, high-definition USB cameras from Microsoft (Microsoft Corp, Redmond, WA) were used. These cameras were mounted parallel with an interaxial separation of 38.86 mm. The video feed from these cameras were used to construct a stereoscopic depth channel for object recognition and tracking in three dimensions. The computer used for prototype testing was equipped with an Intel © Core i7 processor, 1.73 Ghz and 8gb ram.

The software element was written in C++ in conjunction with two open source libraries. For the computer vision aspect, we used the OpenCV library from WillowGarage [4]. For the user interface and frame buffering capabilities, we used the Qt graphical user interface (GUI) library from Digia Plc [5]. The software consists of 4 main component classes; here each component inhabits its own processing thread (Fig. 1). These threads are setup using the Win API threading and event functions.

The first component is the video capture and buffer functions. In the first thread, the OpenCV VideoCapture function captures the frames from each camera and gets the current millisecond time stamp. This information is then sent as a structure to two separate buffers. The buffer operates in a second thread and creates a list of these frame structures. This buffer uses the first in-first out Qt Queue function. This Queue adds frames to the end of the list and when queried, returns the top of the list and removes that frame.

The second component is the stereoscopic matching and correspondence. Before tracking, the cameras are calibrated using a chessboard diagram and several OpenCV functions in order to

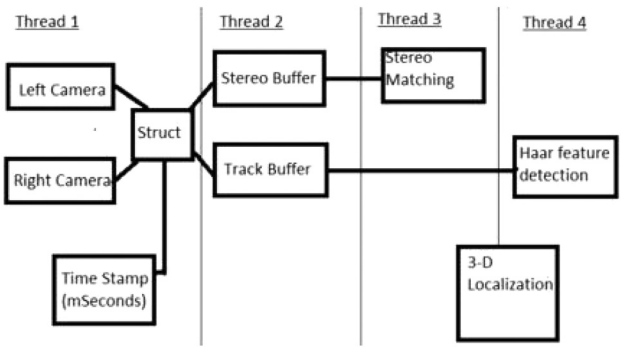

Fig. 1 Software component and thread mapping

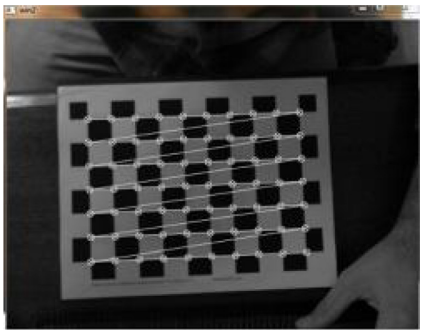

Fig. 2 Calibration

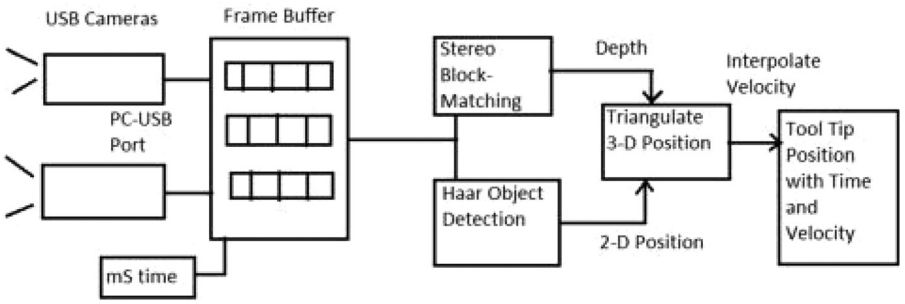

Fig. 3 Design

determine the intrinsic and extrinsic properties of the stereo pair (Fig. 2). After calibrating, the matching is performed by grabbing frames out of the buffer and then providing these frames in the third thread to the OpenCV Block Matching correspondence function [6]. This function determines distinct features in each frame and then compares the separation of these features to extrapolate depth. The output of this component is the estimated depth (distance away from camera) of each pixel.

The third component uses a pre-compiled library of Haar features to locate the robotic tool tip. To accomplish this, a model is built up by taking an array of pictures of the tool tip and training the model to recognize the tool. This model is called the Haar cascade. The model is then queried in each frame in the fourth thread to determine if and where the tool is in the two-dimensional field of view. Localizing pixels in this manner can help infer the two-dimensional global position.

The fourth and final component is the 3-D triangulation and localization. To determine the global position of the tool tip, we compare the local pixel and depth location of the tool tip. Then using intrinsic camera information such as the optical angle of the camera, we build a transformation matrix to determine the tool location in a global coordinate frame.

To test performance, the buffering component, stereo matching, and depth extraction were analyzed using a benchmarking routine to determine the timestamp drop between when the frame structure was captured and stored and when the next thread accessed the Queue and displayed the image (without processing).

3 Results

The capturing component of this design is limited to the 30 FPS that the USB camera can capture video at a resolution of 640×480, higher resolutions are available but at lower frame rates. The benchmarked time differential was around 40 ms resulting in a 5 FPS drop, bringing the total data transfer between the capture and buffering to about 25 FPS or 1 frame every 40 ms.

For the stereo matching and depth extraction performance, the benchmarking routine indicated an additional 20 ms of time delay due to processing. This brings the total FPS down to about 16. The absolute accuracy of the depth extraction has not yet been objectively assessed, but the algorithm can successfully determine the relative change in depth of the tools over another surface, such as the checker board (Fig. 4).

The Haar cascade training model is in place, but the construction of the image library is still underway. No results of the Haar performance are available yet.

4 Interpretation

We present the design and partial implementation of a low-cost system of extracting 3D tool tip positions of robotic surgical tools.

Fig. 4 Stereo matching correctly extracts relative depth of da Vinci tools from background

Our observed frame rate of 16 FPS meets the desired design criteria. A benefit of our design is improved speed due to multi-threading and the ability to retroactively interpolate velocity since each frame structure includes an accurate time stamp. Using this time and position data for each frame should enable accurate computation of several important metrics in gauging surgical skill: tool path, velocity, sub-movement time, and economy of motion.

Future work will include implementing the Haar feature detection component and synthesize this information with the demonstrated depth map. The results of the depth extraction are promising in regards to both speed and accuracy that we expect will improve with better calibration. We also intend to characterize the accuracy of the overall system by comparing our system output with corresponding physical position measurements of the da Vinci tools.

References

[1] Satava, R., and Smith, R., 2012, "Fundamentals of Robotic Surgery: Outcomes Measures and Curriculum Development," Society of Laparoendoscopic Surgeons, pp. 1–2.
[2] Chmarra, M. K., Grimbergen, C. A., and Dankelman, J., 2007, "Systems for Tracking Minimally Invasive Surgical Instruments," Minimally Invasive Therapy, pp. 328–340.
[3] Kowalewski, T. M., 2012, "Real-Time Quantitative Assessment of Surgical Skill," PhD dissertation, University of Washington.
[4] Willow Garage, and Bradski, G., 2012, "Open Source Computer Vision" from http://opencv.willowgarage.com/wiki/
[5] Digia Plc, 2012, "Qt Developer Network" from http://qt- project.org/
[6] Laganiere, R., 2011, *OpenCV2 Computer Vision Application Programming Handbook*, 1st ed., Packt Publishing, Birmingham, UK, Chap. 9.