Viewing Enhancement in Video-Endoscopy[1]

***Abstract**. Video-endoscopy (Figure 1), a mode of minimally invasive surgery, has proven to be significantly less invasive to the patient. However, it creates a much more complex operation environment that requires the surgeon to operate through a video interface. Visual feedback control and image interpretation can be difficult. Poor visual feedback in video-endoscopy prolongs the operation time, increases the risk to the patient, and drives up the cost of health care. It is a major roadblock in replacing the traditional, highly traumatic open surgical procedures with the much less invasive, more patient friendly video-endoscopy, and in training the surgeons to master this new mode of operation. Our research objective is thus to design, code, and validate on real images novel image analysis and rectification algorithms to enhance the visual feedback to the surgeon in video-endoscopy.*
***Index terms**:* computer-assisted medicine, eight-point algorithm, endoscopy, feature tracking

I. Introduction

Our research objective is to develop image analysis and rectification algorithms to enhance the visual feedback to the surgeon in the emerging *minimally invasive surgery* **[2,3]**. There has been a revolution in medical surgery in recent years toward minimally invasive surgery. Minimally invasive surgery reduces the trauma inflicted on the patient during surgery, significantly shortens the time for the patient to recuperate, and lowers the cost of the treatment.

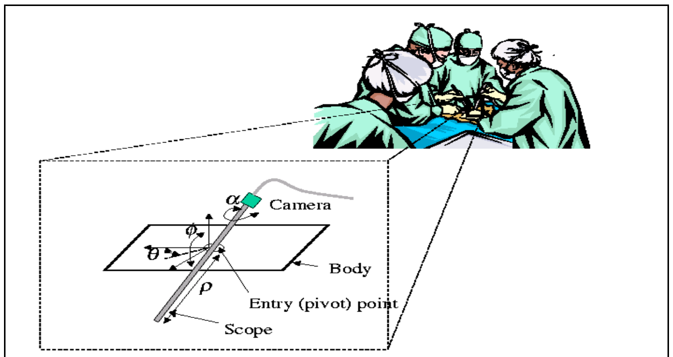

Figure 1: Video-endoscopy and constraints in maneuvering the scope imposed by the pivot point

A key technological advance that has fueled the minimally invasive revolution is video-endoscopy. Endoscopic procedures (Figure 1) are minimally invasive surgical procedures where several small incisions are made on the patient to accommodate surgical instruments such as scalpels, scissors, staple guns, and a video endoscope (also referred to as a scope or a telescope). The scope acquires video images of the bodily cavity that are displayed in real time on a monitor to provide the visual feedback to the surgeon. This setup enables the surgeon to operate instruments through the small incisions, as opposed to a large incision for direct viewing.

From our discussions with practicing surgeons and equipment suppliers, we have identified a critical need for image processing assistance to enhance the surgeon's visual feedback in video-endoscopy. The problem is briefly described as follows: When the scope is inserted into a highly constrictive body cavity and is subject to the entry point constraint (Figure 1), blind spots in the view arise. Often times large panning and rotation of the scope is used to eliminate such blind spots. In fact, the scopes are purposely designed with the viewing direction deviating from the length of the instrument to provide an "off-axis" view so that an axial rotation of the scope has a panning effect to enlarge the view volume (Figure 2). However, the view thus acquired is highly non-intuitive, since the physical "up" direction will in general not so appear on the monitor. This effect is called ``*dis-orientation*."

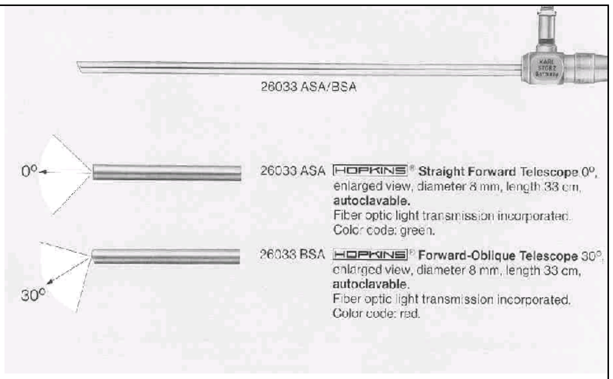
26033 ASA/BSA
0°
26033 ASA HOPKINS® Straight Forward Telescope 0°, enlarged view, diameter 8 mm, length 33 cm, autoclavable. Fiber optic light transmission incorporated. Color code: green.
30°
26033 BSA HOPKINS® Forward-Oblique Telescope 30°, enlarged view, diameter 8 mm, length 33 cm, autoclavable. Fiber optic light transmission incorporated. Color code: red.

Figure 2: Video scopes of different viewing angles

The consequences of dis-orientation are that the surgeon can easily lose the bearing after repeated large movement and rotation of the scope view. This is because in video-endoscopy the body anatomy is not exposed openly. There is no external environment fixture visible in the images to help register the anatomy in the operating room environment. The surgeon has to deduce the scope's bearing based on his/her understanding of the body anatomy.[2] This is difficult if images are not displayed with a strong resemblance to what the surgeon sees in open surgery**.** Hence, solving this problem is part of the general research area of making endoscopic procedures more ``*open-surgery-like*'' and amenable to visual interpretation.

We propose an image rectification algorithm whose objective is to maintain the surgeon's sense of up and down. In the language of computer graphics [1], the ``head-up" vector is to be displayed as upward on the screen at all times. The method we have developed and tested extensively on real images can be summarized as follows. We analyze a video stream from an endoscopy procedure to

[1] Supported in part by Karl-Storz Imaging, Inc and the University of California Micro Program

[2] An analogy is having a user wear a head-mounted helmet that completely covers the user's field of view. The user sees *only* the computer-generated virtual world. After lengthy immersion, the user loses track of the correlation of the virtual and real worlds.

deduce the orientation of the camera and the projection onto the image plane of the physical environment's "up" direction. Panning and rotation of the camera make this projected direction deviate from the screen's upward direction. We then rotate individual video frames by this computed amount, rendering rectified video that properly maintains the head-up direction.

We have broken down the task of computing the rectification angle into three steps. Firstly, 2D features are identified and tracked from frame to frame. Secondly, sets of tracked features are used to deduce the 3D motion undergone by the camera. Finally, the "up" direction, as projected on the image plane, is obtained and the video frames are rotated by this amount.

While the rectification algorithm might appear to be a simple combination of existing techniques, our contribution is in significantly improving the efficiency and robustness of traditional techniques to suit this new application domain. In particular, we reformulated the 2D tracking problem using Fourier analysis and were able to achieve close to 30-fold speed increase over conventional implementation. For 3D tracking, we employ redundancy and robust error norm to significantly improve accuracy and minimize the possibility of loss of track.

II. Mathematical Formulation

The algorithm that processes the video stream comprises essentially three stages. In the first stage, a number of image features are selected and tracked. The features are areas of the image that have a high number of edges or corners with a high intensity contrast. The correspondences of image features are established using an affine model.

The second stage takes the 2D coordinates of the corresponding features in successive frames and estimates the camera motion parameters, using the 8-point algorithm [4]. Here the coordinates of the centers of the tracked 2D features are taken as inputs to the 8-point algorithm. Thirdly, after the camera transformation is obtained, we infer the direction of the environment's "head-up" vector in the camera's current reference. Knowing the vector's 3D coordinates in the camera's current frame allows us to project it onto the image plane. The deviation of this projection from the y-axis tells us by how much the image should be rotated to make the arrow appear again as "up" on the screen. Execution of this computed rotation then rectifies the camera frame, as well as the entire image, to confirm the surgeon's frame of reference. We describe these steps in more details below.

First stage: 2-D feature tracking

The algorithm starts out by tracking a set of 2-D features as they move from one frame to the next. In order to match a set of pixels in one frame to the corresponding pixels in the next frame, an affine model is used. A general affine transform maps a vector **w** to **w'** according to the following form: $\mathbf{w}' = \mathbf{A}\mathbf{w} + \mathbf{b}$ where $\mathbf{A}$ is a constant matrix and $\mathbf{b}$ is a constant vector. It can be shown that this transformation is equivalent to the following sequence of operations: a) shearing, b) rotation and isotropic rescaling, and c) translation. This is most easily understood by expressing the matrix **A** as follows:

$$\mathbf{A} = \mathbf{U\Sigma V^T} = (\mathbf{UV^T})(\mathbf{V\Sigma V^T}) = \mathbf{W}(\mathbf{V\Sigma V^T})$$

$$= \mathbf{WV}\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}\mathbf{V^T} = (\sigma\mathbf{W})(\mathbf{V}\begin{bmatrix} \varepsilon & 0 \\ 0 & 1/\varepsilon \end{bmatrix}\mathbf{V^T})$$

where $\sigma = \sqrt{\sigma_1\sigma_2}$ and $\varepsilon = \sqrt{\sigma_1/\sigma_2}$ and **W** and **V** are orthogonal matrices.

Step a) is represented by the second factor and step b) is represented by the first factor. The reason for dividing the affine transform into these steps is two-fold. Firstly, since each of the three steps involves only two parameters (out of the six affine parameters), searching for optimal parameters in this limited space is much more time-efficient. In other words, the 6-dimensional search is conducted within three 2-dimensional subspaces in a sequential manner, with previous subspace solutions being used as the starting points in the next subspace search. The second reason that we have divided the affine transform into steps a-c is that for step b) and step c), we have developed efficient search methods. These are now described.

In order to track a feature, we define an objective function that captures the overlap of an affine-transformed feature in one frame with the feature in the next frame. If the affine parameters are correctly chosen, then the overlap will be significant and this function should have a small value. Thus, we are defining a function whose minimum we seek. The function we use is the following:

$$f(T) = \sum_{\mathbf{x} \in boundingbox} [I(\mathbf{x}) - J(T\mathbf{x})]^2$$

where I and J are two adjacent frames, T represents the transformation we seek and the bounding box is a square of length *2s* surrounding the feature. To simplify the following discussion, let us assume that T represents only a translation. In that case, we have:

$$f(\mathbf{b}) = \sum_{\mathbf{x} \in boundingbox} [I(\mathbf{x}) - J(\mathbf{x} + \mathbf{b})]^2$$

where **b** is allowed to vary within some "possibility window" (centered around the 0-vector). Now, if we let the possibility window have the same size as the bounding box, then this is equivalent to allowing the feature to migrate by *+/- s* in each direction. In such a case, the area "swept" out in J by all possible locations of the translated bounding box covers a square of size *4s*. Since this represents the universe of pixels being examined in J for this particular computation, it becomes convenient to express f as a sum over this larger box ("swept box"). To keep the value of the sum unchanged, we need to introduce a multiplicative "mask", whose value is unity inside the bounding box and zero outside. Putting this together, we get:

$$f(\mathbf{b}) = \sum_{\mathbf{x} \in sweptbox} [I(\mathbf{x}) - J(\mathbf{x} + \mathbf{b})]^2 \theta(\mathbf{x})$$

where theta is the mask. Now, it is possible to extend the allowable values of **b** by a factor of 2, as long as we define the argument of *J* as "wrapping around" when the limits of "swept box" are exceeded. There is no harm in permitting these additional values of **b**, since the wrapping around of the "tentative" feature in *J* will produce a very bad match with the feature in *I*. Thus, these additional values of **b** will not be selected as the solution minimizing *f*. The motivation for augmenting **b**'s domain is that by doing so we obtain a standard circular convolution expression:

$$f(\mathbf{b}) = \sum_{\mathbf{x}\in sweptbox}[I^2(\mathbf{x})\theta(\mathbf{x}) - 2I(\mathbf{x})\theta(\mathbf{x})J(\mathbf{x}+\mathbf{b}) + J^2(\mathbf{x}+\mathbf{b})\theta(\mathbf{x})]$$

Now, the first term is a constant and the second and third terms are convolutions that can be evaluated efficiently for all values of **b**. The use of the Fast Fourier Transform allows this computation to be done in $\mathrm{O}(n\log n)$ time, instead of $\mathrm{O}(n^2)$ time (where n = number of pixels in "swept box"). Once *f* has been computed for all values of **b**, the minimum is selected as the solution.

The procedure for handling step b) (rotation and isotropic rescaling) is much the same, if we perform a certain change of variables. This can best be seen if we express the position **x** as a complex number z. Our expression for *f* for step b) is:

$$f(T) = \sum_{\mathbf{x}\in boundingbox}[I(\mathbf{x}) - J(T\mathbf{x})]^2$$

where T represents rotation and isotropic rescaling. When we employ the complex number z to represent **x**, then T(**x**) becomes $z_0 z$ for some complex constant z_0. Thus:

$$\begin{aligned} J(Tx) &= J(z_0 z) = J(\exp(\ln(z_0 z))) \\ &= J'(\ln(z_0 z)) = J'(\ln(z_0) + \ln(z)) \\ &= J'(w_0 + w) \end{aligned}$$

where *J'* is the composition of the *J* and exp functions and $w = \ln z$ is our change of variables. The key observation to make here is that the argument of *J'* is the *sum* of the bound variable and the free variable and is therefore in the form of a convolution. As in the previous case, this allows efficient evaluation through the use of the FFT. More formally, we have:

$$\begin{aligned} f &= \sum_{z\in boundingbox}[I(z) - J(z_0 z)]^2 \\ &= \sum_{e^w\in b.b.}[I'(w) - J'(w_0 + w)]^2 \\ &= \sum_{w\in b.b.'}[I'(w) - J'(w_0 + w)]^2 e^{2\mathrm{Re}(w)} \end{aligned}$$

The last factor of $e^{2\mathrm{Re}(w)}$ is due to the fact that a *uniformly-spaced* grid in z-space has been replaced by a *uniformly-spaced* grid in w-space. Including this factor in our discrete case is analogous to correctly converting a differential expression (such as d^2x) in the continuous case, upon a change of variables. Finally, if we process this expression in a similar way that was done for step c) (translation), we again end up with a formula efficiently evaluated by means of the FFT.

As for step a) (shearing), when the same complex number approach is taken, the transformation assumes a rather more complicated form than before. Instead of $z_0 z$, we get:

$$z\cosh|z_0| + \bar{z}\left(\frac{\bar{z}_0}{z_0}\right)^{1/2}\sinh|z_0|$$

The problem in this case is the presence of both z and $\bar{z}$, which does not allow for the factoring into a purely z-dependent factor and a purely z_0-dependent factor. Thus, the shearing portion of the affine transform does not appear to be amenable to the same computationally efficient method that may be applied to the translational and rotational/rescaling portions. In our application, since computational efficiency is paramount, we implement the above formulation for steps b) and c).

Second stage: 3-D reconstruction

For the second stage of the algorithm, the point correspondences of at least eight tracked points (obtained in the first stage) are used as inputs. The goal of the second stage is to obtain the 3-D transformation that the camera has undergone between the two frames. Within this stage, there are two separate steps. Firstly, a matrix known as the fundamental matrix is determined from the point correspondences using a technique known as the "epipolar formulation" [**5**]. Secondly, the 3-D transformation is extracted from the fundamental matrix by performing a factorization. In both of these steps an SVD factorization is employed, although in unrelated ways.

In the first step, a mathematical condition known as the epipolar constraint is used to find the fundamental matrix **F**. This condition states that if a point has coordinates in the i'th frame given by $\mathbf{v} = (x, y, f)^T$ (where x and y denote the pixel location and f is the focal length) and coordinates in the (i+1)'th frame given by $\mathbf{v'} = (x', y', f)^T$, then $\mathbf{v'^T Fv} = 0$. In solving for **F**, we treat its elements as being nine independent unknowns (with an overall undecidable scale factor). In reality, however, **F** is the product of an anti-symmetric matrix and a unitary matrix, so the elements of **F** are not truly independent. However, when employing the epipolar constraint, this excess freedom can be ignored, since in the end, given *physically generated* $\mathbf{v}$ and $\mathbf{v'}$, **F** will turn out to have this particular factorization. Even if the computed **F** cannot be exactly factorized in this way, a physically meaningful 3-D transform still can be extracted from **F** (see below). Therefore, the computation of **F** can proceed by treating its elements as independent variables.

The above equation is generated by applying the epipolar constraint to one tracked point. However, since multiple points are being tracked, a system of linear homogeneous equations is produced, where the unknowns are the elements of **F**. After expressing the nine elements of **F** as a vector **f**, the system of linear equations can be put into matrix form by first defining:

$$\mathbf{f} \equiv [F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33}]^T$$

$\mathbf{z_k} \equiv \mathbf{v_k} \otimes \mathbf{v'}_k$ ($\otimes$ denotes tensor product)

$$\mathbf{Z} \equiv [\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_m]$$

where m = number of point correspondences. Then the set of epipolar constraints for all points is given by: $\mathbf{Z^T f} = 0$. Since solving for **f** is equivalent to finding the null space of $\mathbf{Z^T}$, it is convenient to employ the SVD. The vector associated with a singular value of zero will be the solution to **f**. In practice, however, no singular value will be exactly zero, so the smallest one is the best approximation to the null space. The main advantage in this formalism is that even if more than eight points are used and the system is over-determined, we have a systematic way of getting a unique result for **f**. Over-determination in the system of equations is in fact desirable, since it increases robustness. The SVD is used in the following way to compute **f**:

$$\mathbf{Z}^{\mathbf{T}} = \mathbf{U\Sigma}\mathrm{V}^{\mathbf{T}} \quad , \qquad \Sigma = \begin{bmatrix} diag(\sigma_1, \sigma_2, \ldots \sigma_9) \\ 0 \end{bmatrix} \quad ,$$

$$\sigma_9 = 0 ,$$

$$\mathbf{V} = \begin{bmatrix} \mathrm{v}_1, \mathrm{v}_2, \ldots \mathrm{v}_9 \end{bmatrix} \quad , \quad \mathrm{v}_9 = \text{null} - \text{direction}$$

Once the fundamental matrix is obtained, it needs to be factored as described above. The first factor will be an anti-symmetric 3x3 matrix that expresses the translational component of the 3-D transformation. The second factor is a unitary 3x3 matrix and represents rotation.

As mentioned in the previous section, the fundamental matrix, obtained by employing a system of epipolar constraints, may in fact turn out not to be exactly factorizable into anti-symmetric and unitary components. With real-world data, this is to be expected due to noise in the feature positions. However, a unitary matrix representing the best estimation of the rotation can still be extracted from **F**. For this, we again make use of the SVD. Let $\mathbf{F} = \mathbf{TR}$ where $\mathbf{T}$ is anti-symmetric and $\mathbf{R}$ is unitary. After performing an SVD, we get:

$$\mathbf{F} = \mathbf{U\Sigma}\mathrm{V}^{\mathbf{T}} \quad \text{for some } \mathbf{U}, \ \mathbf{\Sigma} \text{ and } \mathbf{V}$$

Let **P** be the following unitary matrix:

$$\mathbf{P} \equiv \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We can insert $\mathbf{I} = \mathbf{PU}^{\mathbf{T}}\mathbf{UP}^{\mathbf{T}}$ to get:

$$\mathbf{F} = \mathbf{U\Sigma}(\mathbf{PU}^{\mathbf{T}}\mathbf{UP}^{\mathbf{T}})\mathbf{V}^{\mathbf{T}}$$

$$= (\mathbf{U\Sigma PU}^{\mathbf{T}})(\mathbf{UP}^{\mathbf{T}}\mathbf{V}^{\mathbf{T}}) \equiv \tilde{\mathbf{T}}\tilde{\mathbf{R}}$$

Given that $\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}$ we get:

$$\tilde{\mathbf{T}} = \mathbf{U\Sigma PU}^{\mathbf{T}} = \mathbf{U}\begin{bmatrix} 0 & -\sigma_1 & 0 \\ \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}\mathbf{U}^{\mathbf{T}} \qquad \text{and}$$

$$\tilde{\mathbf{R}} = \mathbf{UP}^{\mathbf{T}}\mathbf{V}^{\mathbf{T}}$$

Clearly, $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{R}}$ can be identified as the anti-symmetric $\mathbf{T}$ and the unitary $\mathbf{R}$ if and only if $\sigma_1 = \sigma_2$ and $\sigma_3 = 0$. Finally, we obtain:

$$\mathbf{R} = \mathbf{UP}^{\mathbf{T}}\mathbf{V}^{\mathbf{T}} = [\mathrm{u}_1\mathrm{u}_2\mathrm{u}_3]\mathbf{P}^{\mathrm{T}}\begin{bmatrix} \mathrm{v}_1^{\mathrm{T}} \\ \mathrm{v}_2^{\mathrm{T}} \\ \mathrm{v}_3^{\mathrm{T}} \end{bmatrix} = \mathrm{u}_1\mathrm{v}_2^{\mathrm{T}} - \mathrm{u}_2\mathrm{v}_1^{\mathrm{T}} + \mathrm{u}_3\mathrm{v}_3^{\mathrm{T}}$$

which is the rotation matrix we seek.

The above derivation assumes ideal conditions where **F** can be decomposed into an anti-symmetric factor and a unitary factor. In a real-world scenario, this condition will not be exactly satisfied. However, by selecting σ_3 to be the *smallest* singular value, we can still make use of the result for $\mathbf{R}$ with the guarantee that it is unitary.

Third stage: image rectification

Finally, once $\mathbf{R}$ is determined, the orientation (in the camera's coordinate system) of a virtual "up" vector is updated. After projecting the vector onto the x-y plane (the screen), the angle of deviation from the y-axis can be determined. Using this information, the image is rotated to compensate for this deviation angle. This puts the vector (and everything else in the image) in the "up" position as seen on the screen.

As a final point, it should be mentioned that for the ideal case, the expression from above $\mathbf{Z} \equiv [\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_m]$ can be replaced with no side-effects by the expression $\mathbf{Z} \equiv [\alpha_1\mathbf{z}_1, \alpha_2\mathbf{z}_2, \ldots \alpha_m\mathbf{z}_m]$, where the α's are arbitrary numbers. This follows from the fact that each epipolar constraint equation is homogeneous. In effect, a particular α will act as a weight that that point correspondence has on obtaining the fundamental matrix. Therefore, for real-world data, it makes sense to choose a value for an α that reflects the confidence in that particular point correspondence. Several different criteria can be used here, including using the value of the 2-D feature overlap function (eg. as part of a Boltzmann weighting factor). Another way is to compute **f** in two passes. In the first pass, all the α's are 1 and we obtain a tentative $\tilde{\mathbf{f}}$. For the second pass, the following selection is made: $\alpha_i = h((\mathbf{z}_i^T\tilde{\mathbf{f}})^2)$, where h(x) is a monotonically decreasing function. In our application, we selected h(x) to be a downward step-function, such that the p worst point-correspondences are given a weight of 0 and the rest are given 1. Or we suppress the potential outliers with zero weight. We used m = 18 and p = 2.

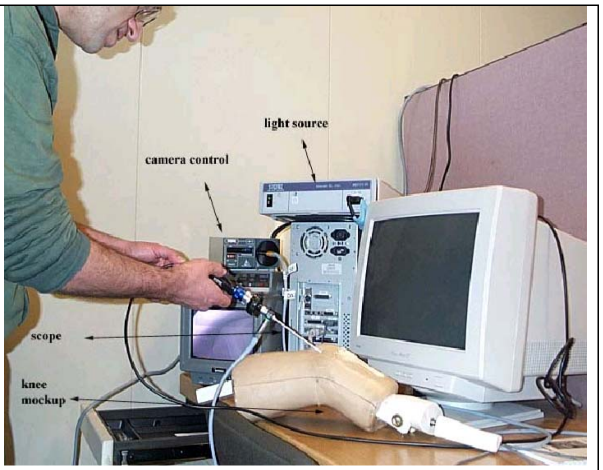

Figure 3: Knee mockup with camera and scope

III. Experimental Set-up and Results

Figure 3 shows the setup used to conduct testing. A knee mockup (provided by Karl Storz Imaging) was used as the working environment for the endoscope. The camera itself was a Karl-Storz Telecam model 20212130U hooked up by S-video to an ATI capture card. The S-Video was then converted to AVI files that were then parsed frame by frame. After rectification, new images were generated and then concatenated into a new AVI stream.

To validate the algorithm, we proceeded in the following stages. The first stage involved testing on a hybrid of real and synthetic video, where a real image was rotated synthetically by known amounts. The purpose is to obtain precise ground truth data so that comparison between the algorithm's rectification angle and the synthetic angle of rotation could be made. Finally, completely real and long video sequences with general camera movement were tested. We described the results below.

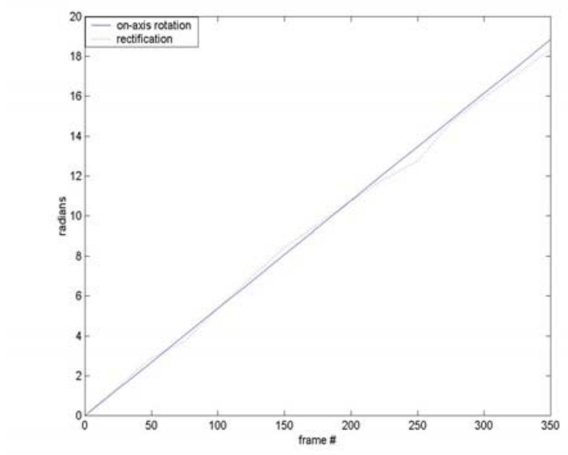

Figure 4: Actual deviation of "up" direction (unrectified) and computed "up": for on-axis rotation

Stage 1: Hybrid of synthetic-real video using a knee mock-up

For this stage, a surgical knee mockup supplied by Karl Storz Imaging (18" long with holes for inserting the camera) was used (Figure 3). The scope was inserted into a cavity in the knee and a typical image was obtained. This image was then rotated synthetically by known amounts. In typical tests, video executing five rotations was processed by the algorithm. As an initializing step, the software selects features of interest by use of an interest operator. The 2-D tracking then is performed for each feature at each frame. Two adjustable parameters are the number of features used (at least eight) and the frequency with which the 3-D algorithm is executed. More specifically, the 3-D algorithm does not need to run at every new frame. Rather, a frame separation greater than one can be used between executions. We call this number of frames the "inter-3D" parameter. The reason for not selecting the value of this parameter as one is that some error may accrue at each execution. Therefore, we wish to run this algorithm only when enough difference in the camera's coordinates has accumulated. On the other hand, it is important not to set this parameter too high, since in that case features may become occluded or fall off the field of view. Additionally, some frequency in running the 3-D algorithm has to be maintained to ensure real-time updates to the rectification angle. For the frames lying in-between, a linear predictor can be used for obtaining the rectification angle. Again, if the inter-3D

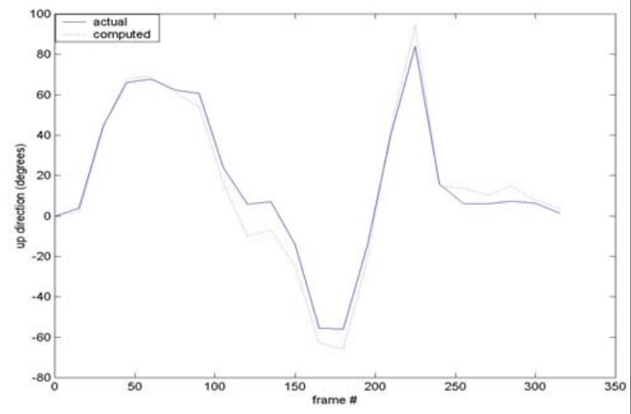

Figure 5: Actual deviation of "up" direction (unrectified) and computed "up": for general camera motion

parameter is not excessively high, then a low-order predictor can be used with good accuracy. Typical values used for these parameters have been: 16 features and 5 frames between 3-D executions.

The purpose of testing this hybrid video was to test the simple case of on-axis rotation and compare with the synthetic ground-truth. Figure 4 shows typical results obtained. As can be seen, the computed angle of rectification is a close fit to the known ground-truth.

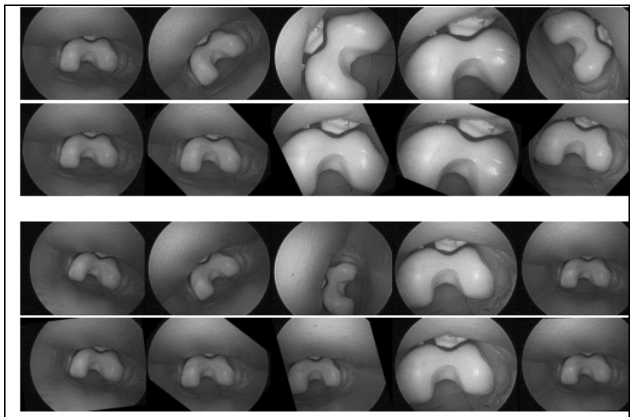

Figure 6: original and rectified sequences (top and bottom rows, respectively) for general camera motion: knee mockup

Stage 2: Real video with arbitrary camera motion

In this stage, video was obtained by inserting the scope into the knee mockup. In this case however, the camera was moved about with a general movement in which all degrees of freedom were varied. In these sequences, the camera was allowed to approach the cavity wall significantly, thus simulating real surgical situations. Also, performance of the interest operator was tested as old features disappear from the field of view and new ones are acquired. Figure 6 shows an original and rectified sequence using the knee mockup. Figure 7 displays the same comparison, but here the interior of the mouth has been used as the environment. Figure 5 compares the computed angle of rectification with the amount obtained by (human) analysis of the original sequence (sets of landmarks were carefully examined frame by frame). Given the arbitrary nature of the camera movement (zooming and panning), the correspondence is very good.

Discussion

From our experiment, we concluded that the source of overall error in the algorithm comes from the 2-D tracking. More specifically, over long sequences, a bounding box surrounding a feature will inevitably have some drift with respect to the true location of the feature. By over-determining the system of equations, we achieve some robustness, especially if these drifts are random in direction. Our results show that minimizing these drifts will be helpful to improving rectification for very long sequences. Preprocessing the images by various image processing techniques is currently being examined. In particular, counter-acting the effect that motion blurring may have on tracking accuracy is being investigated.

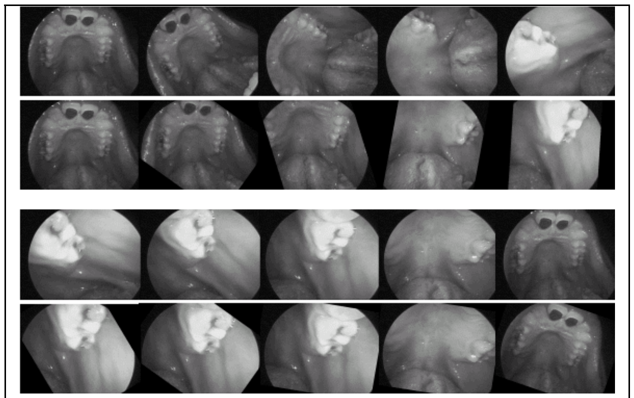

Figure 7: original and rectified sequences (top and bottom rows, respectively) for general camera motion: actual human tissue (mouth)

The timing results for the 2-D tracking using the FFT-convolution method are very positive. Using another method for evaluating the objective (feature overlap) function as a reference, we achieved a 26-fold increase in speed. This reference method utilized a hierarchical approach in finding the minimum of the objective function. It can be briefly described as follows. In the first iteration, the full possibility window is searched, but using a course granularity. In the next iteration, the size of the window is decreased, the center is placed on the previous iteration's solution and the resolution is increased. Eventually, the resolution reaches the pixel level and the algorithm terminates. In addition, before the start of the algorithm, the center of the window can be predicted from the past movements of the bounding box. Using a predicted center allows for employing a smaller initial window, with the net effect of speeding up the algorithm. It is this prediction-based hierarchical method that we used as a comparison to calculate the mentioned speedup. Additionally, it is clear that the FFT method is truly an exhaustive search (within the 2-D affine subspaces), while the reference method is not and can on occasion miss the true minimum.

The 3-D component of the algorithm yields good timing performance as well. It consumes only 1/5 of the duty cycle required for real-time processing.

In the sequences we have analyzed, the deviation of the computed rectification from the known correct amount is approximately 10%. Our current work is focused on reducing this figure.

IV. Concluding Remarks

In this paper, we present our research on enhancing visual feedback to the surgeon in minimally-invasive surgery. A method was presented for estimating the amount of rotation necessary to rectify images forming a video stream to help alleviate the dis-orientation problem in endoscopy. By estimating the "up" direction as seen in the camera's coordinate system, we can obtain the deviation from the screen's "up" direction. This deviation tells us the amount of rotation we need to render a rectified image to maintain the surgeon's sense of head-up direction. An efficient method of computing 2-D feature tracking was presented. The 3-D portion of the algorithm makes use of the epipolar constraint and employs a set of these constraints to generate a matrix equation. With the help of the SVD, we are able to get a optimal approximation to the fundamental matrix. After this step, we again use the SVD to decompose the fundamental matrix into an anti-symmetric factor and a unitary factor. The latter is of interest to us, since it represents the rotation of the camera between two successive frames and can therefore be used to update the camera parameters used for rectification.

By expressing the 2-D tracking objective function as a convolution, we were able to obtain a large speedup over straightforward evaluation. This formulation allows for DSP hardware implementation as well as parallelization (in evaluating 2-D FFTs).

Next steps for us include processing the image stream so that moving objects, such as scalpels, which are not attached rigidly to the rest of the scene, can be segmented out. In this way, only features forming part of the rigid whole will be selected for tracking and the basic assumptions of the algorithm are maintained.

References:

1. J.D. Foley, A. van Darn, S.K. Feiner, and J.F. Hughes. Computer Graphics: Principles and Practice, 2nd ed. Addison-Wesley, Reading, MA, 1990
2. J.F. Hulka and H. Reich. Textbook of Laparoscopy, 2nd Ed. W.B. Saunders, Philadelphia, PA, 1994
3. J.G. Hunter and J.M. Sackier (eds.). Minimally Invasive Surgery. McGraw-Hill, New York, 1993
4. H.C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene from Two Projections," Nature, vol. 293, pp. 133-135, Sept 1981.
5. G. Xu and Z. Zhang, "Epipolar Geometry in Stereo, Motion and Object Recognition", Kluwer Academic Publishers, 1996