Learning Features on Robotic Surgical Tools

Austin Reiter
Columbia University
New York, NY
areiter@cs.columbia.edu

Peter K. Allen
Columbia University
New York, NY
allen@cs.columbia.edu

Tao Zhao
Intuitive Surgical, Inc.
Sunnyvale, CA
tao.zhao@intusurg.com

Abstract

Computer-aided surgical interventions in both manual and robotic procedures have been shown to improve patient outcomes and enhance the skills of the human physician. Tool tracking is one such example that has various applications. In this paper, we show how to learn fine-scaled features on surgical tools for the purpose of pose estimation. Our experiments analyze different state-of-the-art feature descriptors coupled with various learning algorithms on in-vivo data from a surgical robot. We propose that it is important to be able to detect naturally-occurring features robustly in order to achieve long-term, marker-less tool tracking. We also contribute a new improvement on feature classification based on Randomized Trees.

1. Introduction

The benefits of computer-aided intervention for surgery have been evaluated quite extensively and have shown improvement over manual procedures [1]. The introduction of various designs in both the commercial [2] and research [3, 4] domains have opened the doors for computer vision applications to increase the capabilities of physicians.

One such popular paradigm is tool tracking, where applications range from automated visual servoing of motorized camera systems to virtual measurement capabilities which can provide accurate measurement of the sizes of various anatomical structures. For a robot such as the da Vinci® from Intuitive Surgical [2], tool tracking can be used to manage the tools which are off-screen, increasing patient's safety, or for overlaying virtual indicators of the status of a tool (e.g., the firing status of an electro-cautery tool).

Tool tracking has been addressed using various approaches, ranging from marker-based [5] and pixel segmentation [6] concepts to model-based [7, 8] or template matching [9] algorithms. A recent approach combined multiple features with learning at the pixel level [10], however few, if any, address interest regions using rich feature descriptors on the tip of the tool.

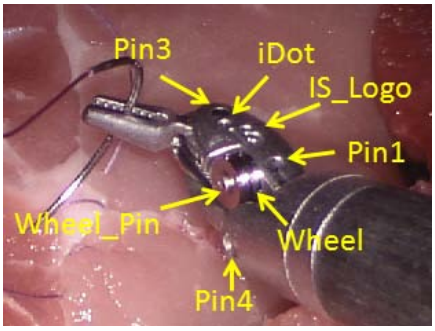


Figure 1. The feature classes we detect on the LND tool. We concentrate on 7 different types of naturally-occurring landmarks.

This paper proposes methods to learn particular 3D landmarks on a surgical tool using machine learning algorithms applied to feature descriptors within in-vivo environments, which are known to have many challenges such as: scene deformability, perspective effects, significant lighting changes, occlusions, and frequent exit-and-re-entry of the desired tool. The features we detect are small-scaled ($\sim 2\%$ of the image), vary in the amount of texture, and are observed under many different perspective views. The features are designed to be used within a marker-less pose estimation framework which fuses kinematics with vision, although this is out-of-the-scope of the current paper.

Our experiments show that we can reliably detect these important landmarks through these challenges with high accuracy. We show trade-offs between accuracy and run-time, and compare and contrast different state-of-the-art descriptors with successful classification approaches. We also contribute a new method on using Randomized Trees for classification which shows an improvement in accuracy.

2. Methods

We developed a tracking method which relies on consistently detecting known 3D locations on a robotic surgical tool. We associate these 3D observations with a kinematics model to solve for the best articulated pose over time.

978-1-4673-1612-5/12/$31.00 ©2012 IEEE

Therefore, the basis of the accuracy for the algorithm lies in being able to detect particular landmarks on the tool reliably. One common approach to this is to employ specialized fiducial markers, however these present challenges in both manufacturability and cost and may be visually distracting to the physician. As such, it is ideal to use naturally-occurring landmarks on the tool, which tend to be both small in size and difficult to capture.

This paper describes efforts towards pairing state-of-the-art feature descriptors with various successful machine learning algorithms for feature detection on a tool. Specifically, we choose to evaluate the following descriptor methods: Scale Invariant Feature Transform (SIFT) [11], Histogram of Oriented Gradients (HoG) [12], and Region Covariance (Covar) [13]. Each is coupled with a Support Vector Machine (SVM), a Randomized Tree (RT) [14] classifier, and a modified approach to RTs we developed called *Best Weighted Randomized Trees* (BWRT), which we describe in further details in Sec. 2.2. We consider both accuracy and run-time in our evaluations and concentrate on real surgical data collected from the da Vinci® robot.

2.1. Feature Descriptors

SIFT [11] has been shown to be one of the most effective descriptors for feature point recognition/matching [15] and is often used as a benchmark against which all other feature descriptors are compared. It has been shown that SIFT can be well approximated using integral images for more efficient extraction [16], and we use ideas based on this method for classifying densely at many pixels in an image.

HoG [12] descriptors describe shape or texture by a histogram of edge orientations quantized into discrete bins (we use 45) and weighted on gradient magnitude, so as to allow higher-contrast locations more contribution than lower-contrast pixels. These can also be efficiently extracted using integral histograms [17].

Region Covariance [13] uses the covariance matrix of d features in a small image region to serve as the feature descriptor. Given an image I of size $[W \times H]$, we extract d=*11* features, resulting in a $[W \times H \times d]$ feature image:

$$\mathbf{F} = [x \quad y \quad \mathbb{H} \quad \mathbb{S} \quad \mathbb{L} \quad I_x \quad I_y \quad I_{xx} \quad I_{yy} \quad \sqrt{I_x^2 + I_y^2} \quad \arctan(I_y/I_x)] \tag{1}$$

where x, y are the pixel locations; $\mathbb{H}, \mathbb{S}, \mathbb{L}$ are the hue, saturation, and luminance values at pixel location (x, y); I_x, I_y are the 1^{st}-order spatial derivatives; I_{xx}, I_{yy} are the 2^{nd}-order spatial derivatives; and the latter two features are the gradient magnitude and orientation, respectively. The covariance matrix $C_{\mathbf{R}} \in \mathbb{R}^{d \times d}$ of an arbitrary rectangular region $\mathbf{R}$ within $\mathbf{F}$ then becomes the feature descriptor.

In [13] it is shown that the covariance matrix of *any rectangular region* can be extracted in $O(d^2)$ time using integral images. However, the d-dimensional nonsingular covariance matrix descriptors cannot be used *as is* to perform classification tasks directly because they do not lie on a vector space, but rather on a connected Riemannian manifold, and so the descriptors must be post-processed.

[18] describes an in-depth mathematical overview for post-processing the covariances. Here we briefly summarize. Symmetric positive definite matrices, of which our nonsingular covariance matrices belong, can be formulated as a connected Riemannian manifold. A manifold is locally similar to a Euclidean space, and so every point on the manifold has a neighborhood in which a homeomorphism can be defined to map to a tangent vector space.

Our goal is to map our $[d \times d]$ dimensional matrices to a tangent space at some point on the manifold, which will transform the descriptors to a Euclidean multi-dimensional vector-space for use within the classifier. Given a matrix $\mathbf{X} \in \mathbb{R}^{d \times d}$, we define the manifold-specific exponential and logarithmic mappings at the point $\mathbf{Y} \in \mathbb{R}^{\mathbf{d} \times \mathbf{d}}$ as:

$$\exp_{\mathbf{X}}(\mathbf{Y}) = \mathbf{X}^{\frac{1}{2}} \exp(\mathbf{X}^{-\frac{1}{2}} \mathbf{Y} \mathbf{X}^{-\frac{1}{2}}) \mathbf{X}^{\frac{1}{2}} \tag{2}$$

$$\log_{\mathbf{X}}(\mathbf{Y}) = \mathbf{X}^{\frac{1}{2}} \log(\mathbf{X}^{-\frac{1}{2}} \mathbf{Y} \mathbf{X}^{-\frac{1}{2}}) \mathbf{X}^{\frac{1}{2}} \tag{3}$$

In these formulations, exp and log are the ordinary matrix exponential and logarithmic operations. Next, we define an orthogonal coordinate system at a tangent space with:

$$\mathbf{vec}_{\mathbf{X}}(\mathbf{Y}) = upper(\mathbf{X}^{-\frac{1}{2}} \mathbf{Y} \mathbf{X}^{-\frac{1}{2}}) \tag{4}$$

where $upper$ extracts the vector form of the upper triangular part of the matrix. In the end, we are left with a vector space with dimensionality $q = d(d+1)/2$.

The manifold point at which we construct a tangent space is the mean covariance matrix $\mu_{\mathbf{C_R}}$ of the training data used to train the classifiers, computed in the Riemannian space by minimizing the sum of squared distances:

$$\mu_{\mathbf{C_R}} = \underset{\mathbf{Y} \in \mathbf{M}}{\operatorname{argmin}} \sum_{i=1}^{N} d^2(\mathbf{X}_i, \mathbf{Y}) \tag{5}$$

This can be computed using the following update rule in a gradient descent procedure:

$$\mu_{\mathbf{C_R}}^{t+1} = \exp_{\mu_{\mathbf{C_R}}^t} [\frac{1}{N} \sum_{i=1}^{N} \log_{\mu^t}(\mathbf{X_i})] \tag{6}$$

We use the logarithmic mapping of $\mathbf{Y}$ at $\mu_{\mathbf{C_R}}$ to obtain our final vectors as in [18].

2.2. Classifier Algorithms

Next we describe the different classification algorithms which are paired with each of the feature descriptors. The task at hand is a multi-class problem, where each feature is

a separate class (labeled in Fig. 1). Later on we consider both accuracy and run-time and the trade-offs in between.

Support Vector Machines: An SVM constructs a set of hyperplanes which seek to maximize the distance to the nearest training point of any class. The vectors which define the hyperplanes can be chosen as linear combinations of the feature vectors, called *Support Vectors*, which has the effect that more training data may produce a better overall result, but at the cost of higher computations. We use Radial Basis Functions as the kernel during learning.

Randomized Trees: Next we adapt *Randomized Trees* (RTs) [14] to perform our multi-class classification. In addition to providing feature labels, we would like to retrieve confidence values for the classification task which will be used to construct class-conditional likelihood images.

RTs naturally handle multi-class problems very efficiently while retaining an easy training procedure. The RT classifier Λ is composed of a series of L randomly-generated trees $\Lambda = [\gamma_1, \ldots, \gamma_L]$, each of depth m. Each tree γ_i, for $i \in 1, \ldots, L$, is a fully-balanced binary tree composed of internal nodes, each of which contains a randomly-generated test that splits the space of data to be classified, and leaf nodes which contain estimates of the posterior distributions of the feature classes.

To train the tree, the training features are dropped down the tree, performing binary tests at each internal node until a leaf node is reached. Each leaf node contains a histogram of length equal to the number of feature classes b. The histogram at each leaf counts the number of times a feature with each class label reaches that node. At the end of the training session, the histogram counts are turned into probabilities by normalizing the counts at a leaf node by the total number of hits at that node. A feature is then classified by dropping it down the trained tree, again until a leaf node is reached. At this point, the feature is assigned the probabilities of belonging to a feature class depending on the posterior distribution stored at the leaf from training.

Because it is computationally intractable to perform all possible tests of the feature, L and m should be chosen so as to cover the search space sufficiently and avoid randomness. Although this approach has been very successful for matching keypoints [14], traditionally the internal node tests are performed on a small patch of the gray image by randomly selecting 2 pixels and applying a binary operation ($\leq$) to determine which path to take to a child. In our problem, we are using feature descriptor vectors rather than image patches, and so we must adapt the node tests to suit our problem.

To this end, we use a similar approach to [19] in creating node tests for feature descriptors. For each internal tree node we randomly construct a linear classifier $h_i(\mathbf{x})$ on feature vector $\mathbf{x}$ to split the data:

$$h_i(\mathbf{x}) = \begin{cases} \mathbf{n^T x} + z \leq 0 & \text{go to right child} \\ otherwise & \text{go to left child} \end{cases} \quad (7)$$

where $\mathbf{n}$ is a randomly generated vector of the same length as feature $\mathbf{x}$ with random values in the range $[-1, 1]$ and $z \in [-1, 1]$ is also randomly generated. This node test allows for robust splitting of the data and is efficiently utilized as it is simply a dot product per tree node. In this way, we train the tree with the descriptor vectors and build up probability distributions at the leaf nodes. The results from each γ_i are *equally-averaged* across the L trees.

Best Weighted Randomized Trees: We developed a method which is able to improve on the standard RT approach which we call *Best Weighted Randomized Trees* (BWRTs). The modification lies in two observations:

- Each tree γ_i is essentially a weak classifier, but some may work better than others, and we can weight them according to how well they behave on the training data
- We can show improvement by initially creating a *randomized tree bag* Ω of size $E \gg L$, evaluating each tree in Ω on the training data, and selecting the best L trees for inclusion in the final classifier $\Lambda \subseteq \Omega$ according to the Root-Mean Squared (RMS) error.

The latter point allows us to consider more of the parameter space while retaining the computational efficiency of RTs by only selecting the best performers. In order to evaluate a particular tree on the training data, we look at the posterior distributions at the leaf nodes. First, all trees in Ω are trained as usual. Next, given a candidate tree $\tilde{\gamma_i} \in \Omega$, we drop each training sample through $\tilde{\gamma_i}$ until a leaf node is reached. Given training feature $\mathbf{X_j}$ and feature classes $1, \ldots, b$, the posterior distribution at the leaf node contains b conditional probabilities $p_{\tilde{\gamma_i}}(y|\mathbf{X}_j)$ where $y \in 1, \ldots, b$. To evaluate the goodness of tree $\tilde{\gamma_i}$ on $\mathbf{X_j}$, we compare $p_{\tilde{\gamma_i}}(y_j|X_j)$ to the desired probability $\mathbf{1}$ of label y_j, and accumulate the RMS error of all training features $\mathbf{X_j}$ across all trees in Ω. The top L trees (according to low RMS) are selected for the final classifier Λ.

In addition to selecting the best trees in the bag, we use the error terms as weights on the trees so that trees that label better have a larger say in the final result. As such, for each $\gamma_i \in \Lambda$ we compute an associated weight w_i such that:

$$w_i = 1/rms_i \quad (8)$$

where rms_i is the accumulated RMS error of tree γ_i. At the end, we normalize the weights w_i to unit-sum.

3. Experiments and Results

Experiments were performed on in-vivo data collected from a da Vinci® robot [2]. We concentrate on the Large

Figure 2. Examples of appearance changes of the *IS_Logo* feature

Needle Driver (LND) tool, keeping in mind this technique may be applied to many other types of tools. 7 naturally-occurring landmarks are manually selected and shown in Fig. 1 overlaid on an image of the LND. The features chosen are of the pins that hold the distal clevis together, the **IS** logo in the center, and the wheel and wheel pin. Each feature is a class in our multi-class classification experiments. Fig. 2 shows example appearance changes typically encountered of the IS Logo feature through different lighting and perspective effects, to motivate the need for a robust descriptor.

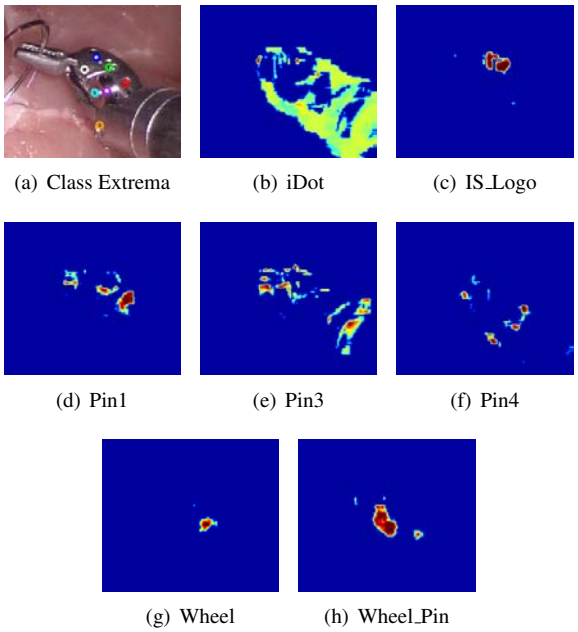
3.1. Training

To train our classifiers, we use 5 different video sequences which span various in-vivo experiments, to best cover a range of appearances. For each frame in the ground truth procedure, we manually drag the best encompassing bounding-box around each feature of interest. To obtain as large a dataset as possible with reasonable effort, we coast through small temporal spaces using Lucas-Kanade optical flow to predict ground truth locations between user clicks. Overall, we use $\sim 15,000$ total training samples across the seven feature classes. For each training sample, we compute each of the three feature descriptors described in Sec. 2.1. Next, for each set of feature descriptors we train classifiers using each of the methods described in Sec. 2.2. This results in a total of nine possible descriptor/classifier combinations. For convenience, we notate these as: SIFT/SVM, SIFT/RT, SIFT/BWRT, HoG/SVM, HoG/RT, HoG/BWRT, Covar/SVM, Covar/RT, and Covar/BWRT.

3.2. Evaluation

For the purposes of evaluation, we look at confidence values resulting from the classifiers rather than the discrete labelings. The reason is that a classification of label l arises because its confidence is greater than all other $b-1$ classes in the classifier, however a confidence of 95% for one pixel location means more than a confidence of 51% for that same labeling at a different location. In this case, we would choose the feature with the higher probability. For this reason, we evaluate accuracy in the *likelihood-space* rather than in the *labeling-space*.

Our method for evaluation using the likelihood-space



(a) Class Extrema (b) iDot (c) IS_Logo

(d) Pin1 (e) Pin3 (f) Pin4

(g) Wheel (h) Wheel_Pin

Figure 3. Example likelihood images along with 6/7 successfully detected feature classes correctly located as extrema in (a). The *Pin3* (e) feature is incorrectly localized (white circle). The color-coding for the circles in (a) is: **Blue** for iDot, **Green** for IS_Logo (c), **Red** for Pin1 (d), **Orange** for Pin4 (f), **Purple** for Wheel (g), and **Cyan** for Wheel_Pin (h).

works as follows: given a test image, we run the multi-class classifier through the entire image, resulting in b probabilities at each pixel for each feature class. This yields b different likelihood images. In each likelihood, we perform non-maximal suppression to obtain the 3 best peaks in the likelihood. Then, we mark a feature classification correct if any of the 3 peaks in the likelihood is within a distance threshold (we used 1% of the image size) of the ground truth *for that feature type*. We choose this method because it is often the case that there is a local peak at the correct location for a feature, but it is not always the global peak. Therefore, in a full tracking system a temporal coherence filter can eliminate these outliers. Fig. 3 shows sample likelihoods on the tip of the LND tool overlaid with extrema locations. Six of the seven features are correctly detected as peaks in the class-conditional likelihoods, where the *Pin3* (Fig. 3(e)) feature is incorrectly detected. This was produced using the Covar/RT approach.

Accuracy: To evaluate accuracy, we tested on video which was specifically not used in the training stage. We tested 1500 frames from in-vivo sequences, which resulted in ~ 4500 possible feature detections which were ground-truthed. The accuracy against the ground truth is shown in Fig. 4 for each individual feature type, separately. It is clear that different features are more reliably detected than others, which we attribute to differences in size, texture, and

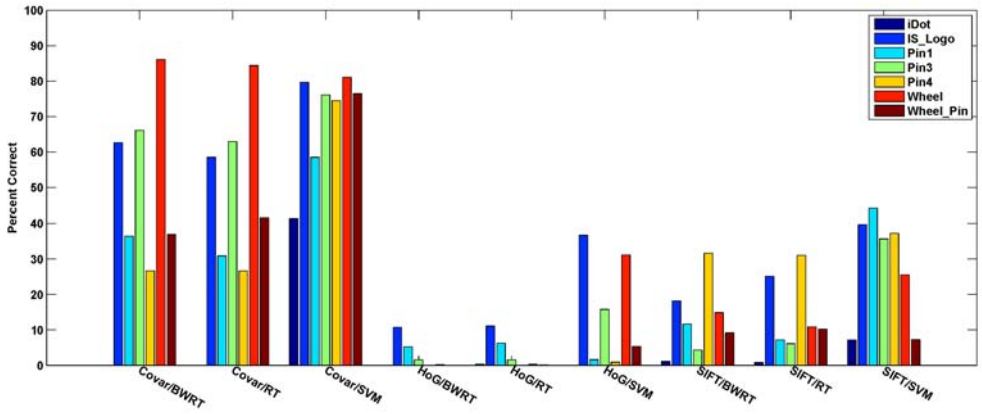


Figure 4. Accuracy graph of test features against ground truth, shown for each individual feature type, separately. We tested with 1500 frames of an in-vivo surgical sequence, resulting in ~ 4500 possible feature detections.

uniqueness. However, it is obvious from this graph the the **Region Covariance** out-performs both the SIFT and HoG descriptors, regardless of the learning algorithm.

A more detailed analysis reveals that the SVM evaluates best overall, although both RT and BWRT are certainly comparable as different features perform differently. For example, Covar/SVM classifies the *Wheel* feature with 81% accuracy, whereas Covar/RT classifies that same feature at 84% and Covar/BWRT at 86%. Contrastly, Covar/SVM classifies the *IS_Logo* feature at 80% against a classification rate of 59% for Covar/RT and 63% for Covar/BWRT.

The maximum achieved accuracy using the SIFT descriptor was 44% using SIFT/SVM on the *Pin1* feature. Using the HoG descriptor, the best achieved accuracy was 37% using HoG/SVM on the *IS_Logo* feature.

Descriptor	Classifier	Unmasked	Masked
Covar	**SVM**	60185.4	4431.18
	RT	8672.4	1171.01
	BWRT	8685.57	1086.8
SIFT	**SVM**	204696	13704.8
	RT	11914.3	915.163
	BWRT	12325.9	990.732
HoG	**SVM**	55634.6	4216.58
	RT	5231.53	551.321
	BWRT	5388.07	557.324

Table 1. Average msecs/frame

Timing: In addition to accuracy, we also consider the per-frame processing time of each algorithm. As mentioned previously, SVMs tend to become more complex and time consuming as more support vectors are added, which arises due to more training data. Conversely, the tree approaches are designed to be efficient as the node tests are low-cost and only m tests-per-tree across all L trees are needed to classify a feature (in our experiments, $m = 10$ and $L = 90$). In the case of the BWRTs, we began with an initial tree bag of 1000 and selected the best 90 from this bag.

During testing, for a given 640x480 image we classify *every other* pixel using the descriptor/classifier combinations. This amounts to 76,800 descriptor extractions and classifications *per-frame* for each algorithm. We used a constant-size window for each descriptor (21 pixel diameter, empirically determined) for all descriptor types. We analyzed the average run-time per-frame and the results are shown in the third column of Table. 1 in msecs/frame. The higher-dimensional feature vectors required more time, especially in the case of the SVMs. Therefore, SIFT (d=128) had the largest run-time and HoG (d=45) had the smallest. Note that the run-time for the RT and BWRT (d=66) cases should be very close as they are equivalent in terms of behavior, only differing in the values for the weights.

The fastest algorithm was HoG/RT and HoG/BWRT, with the smallest complexity. We note that an increase in speed can be applied to all cases if an initial mask prior were present, which would limit which pixels to analyze in the image. One such algorithm is presented in [10], which we used to confine the classifications to pixels only on the metal tip of the tool. The runtime results (including the time to compute the masks) are shown in the fourth column of Table 1, where we can observe a significant reduction in processing. This gets us closer to a real-time solution, where, for example, the Covar/BWRT approach is reduced to a little over 1 sec/frame. Finally, we analyzed the *percent decrease* in run-time from the SVM case to the RT/BWRT cases for each descriptor. If we are able to sacrifice some accuracy performance, this showed that we can achieve a reduction of up to 80% using Covar, and 90% and 94% using the HoG and SIFT descriptors, respectively. These are not trivial speed-ups, and should be considered in the final choice of the feature detection algorithm.

4. Discussions

We note that although some feature types are not always detected, we only need a minimum of 3 on a given frame to recover the articulated pose (because our algorithm fuses kinematics with vision), and so across the 7 chosen landmarks preliminary experiments show that the percent correct achieved in this paper is sufficient for long-term tracking. Because we have confidences we can reject features with low probabilities. An example of applying these features into our tracking framework is shown at: http://vimeo.com/40102648, where yellow indicates the raw kinematics estimate, blue represents the *fixed* kinematics from our tracker, and the green dots are the features described in this paper (using Covar/RT). An interesting question arises when considering tracking 2 tools simultaneously. In this case, we can use the kinematics as a prior on geometric constraints to assign features to the most likely tool pairing. This is a topic of future research.

Finally, we compared the performance of the RT algorithm to our proposed BWRT modified approach. We showed that the SVM is noticeably more accurate, but is also slower. If we are able to trade-off some accuracy to speed-up the runtime, the tree approach is ideal due to its efficiency in multi-class classification. Looking at *percent change*, on average our BWRT approach improved the accuracy from the RT approach by $\sim 10\%$. Again this effect differs depending on the feature type. For the Covar descriptor, we see an increase in accuracy of 18% for the *Pin1* feature and an improvement of 63% using SIFT.

5. Conclusions

In this paper we have presented an examination of state-of-the-art feature descriptors paired with supervised machine learning algorithms for detecting natural features on the tip of a robotic surgical tool. We have shown high accuracy and robustness, as well as trade-offs in computational run-time efficiency towards a long-term, marker-less tool tracking system. We also proposed a novel approach to improving the standard Randomized Tree model of multi-class classification called Best-Weighted Randomized Trees using feature descriptor vectors instead of image patches.

References

[1] G. Hubens, H. Coveliers, L. Balliu, M. Ruppert, and W. Vaneerdeweg, "A performance study comparing manual and robotically assisted laparoscopic surgery using the da vinci system," *Surgical Endoscopy*, vol. 17, pp. 1595–1599, 2003. 1

[2] "Intuitive Surgical, Inc.." http://www.intuitivesurgical.com/. 1, 3

[3] J. Shang, D. Noonan, C. Payne, J. Clark, M. Sodergren, A. Darzi, and G. Yang, "An articulated universal joint based flexible access robot for minimally invasive surgery," in *IEEE Intl. Conf. on Robotics and Automation*, 2011. 1

[4] A. Bajo, R. E. Goldman, L. Wang, D. Fowler, and N. Simaan, "Integration and preliminary evaluation of an insertable robotic effectors platform for single port access surgery," in *IEEE Intl. Conf. on Robotics and Automation*, 2012. 1

[5] M. Groeger, K. Arbter, and G. Hirzinger, "Motion tracking for minimally invasive robotic surgery," in *Medical Robotics, I-Tech Education and Publishing*, pp. 117–148, 2008. 1

[6] C. Doignon, P. Graebling, and M. de Mathelin, "Real-time segmentation of surgical instruments inside the abdominal cavity using a joint hue saturation color feature," in *Real-Time Imaging*, 2005. 1

[7] S. Voros, J. Long, and P. Cinquin, "Automatic detection of instruments in laparoscopic images: A first step towards high-level command of robotic endoscopic holders," *Intl. J. of Robotics Research*, vol. 26, pp. 1173–1190, Nov. 2007. 1

[8] R. Wolf, J. Duchateau, P. Cinquin, and S. Voros, "3d tracking of laparoscopic instruments using statistical and geometric modeling," in *Medical Image Computing and Computer-Assisted Intervention*, 2011. 1

[9] D. Burschka, J. J. Corso, M. Dewan, W. Lau, M. Li, H. Lin, P. Marayong, N. Ramey, G. D. Hager, B. Hoffman, D. Larkin, and C. Hasser, "Navigating inner space: 3-d assistance for minimally invasive surgery," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2004. 1

[10] Z. Pezzementi, S. Voros, and G. Hager, "Articulated object tracking by rendering consistent appearance parts," in *IEEE Intl. Conf. on Robotics and Automation*, 2009. 1, 5

[11] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Intl. J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. 2

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2005. 2

[13] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *European Conf. on Computer Vision*, 2006. 2

[14] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1465–1479, 2006. 2, 3

[15] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1615–1630, 2005. 2

[16] M. Grabner, H. Grabner, and H. Bischof, "Fast approximated sift," in *Asian Conf. on Computer Vision*, 2006. 2

[17] F. Porikli, "Integral histogram: A fast way to extract higtograms in cartesian spaces," in *IEEE. Conf. on Computer Vision and Pattern Recognition*, 2005. 2

[18] O. Tuzel, F. Porikli, and P. Meer, "Human detection via classification on riemannian manifolds," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007. 2

[19] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *ACM Intl. Conf. on Image and Video Retrieval*, 2007. 3