IMAGE-BASED OBJECT STATE MODELING OF A TRANSFER TASK IN SIMULATED SURGICAL TRAINING

Kuo Shiuan Peng
Electrical Computer Engineering
University of Arizona
Tucson, AZ 85721, USA
kspeng@email.arizona.edu

Minsik Hong
Electrical Computer Engineering
University of Arizona
Tucson, AZ 85721, USA
mshong@email.arizona.edu

Jerzy Rozenblit
Electrical Computer Engineering
University of Arizona
Tucson, AZ 85721, USA
jr@email.arizona.edu

ABSTRACT

This paper proposes a real-time, image-based training scenario comprehension method. This method aims to support the visual and haptic guidance system for laparoscopic surgery skill training. The target task of the proposed approach is a simulation model of a peg transfer task, which is one of the hands-on exam topics in the Fundamentals of Laparoscopic Surgery certification. In this paper, a simulation process of an image-based object state modeling method is proposed. It generates a system object state of the transfer task to support the guidance system. A rule-based, intelligent system is used to discern the object state without the aid of any object template or model. This is the novelty of the proposed method.

Keywords: medical simulation, simulation-based surgical training, laparoscopy, image understanding.

1 INTRODUCTION

Computer Assisted Surgical Trainer (CAST) (Rozenblit et al. 2014) is a simulation-based training device designed for laparoscopic surgery skill training. Laparoscopic surgery is a popular and advanced technique which offers patients benefits of minimal invasiveness and fast recovery time. It requires extensive, simulation-based training before operating on patients. CAST can be considered as a simulation-based training device with visual and force guidance to its users.

The system allows the users to work on several task scenarios. It has a visual guidance module (*opt*Viz) which generates a *reference path* laid over the live camera image. The force (haptic) guidance module (*opt*Guide) (Hong and Rozenblit 2016a) provides *force* assistance in moving surgical instruments. To support these two modules, the optimal and collision free path generator (*opt*MIS) was designed by (Napalkova et al. 2014). The generated paths are the reference inputs for the controller of the *opt*Guide and the graphical generator of the *opt*Viz (Shankaran and Rozenblit 2013).

To further develop the *opt*MIS module, a state modeling method of a peg transfer task was designed in a simulated environment (Hong and Rozenblit 2016b). This is one of the important, hands-on training tasks

SpringSim-MSM 2017, April 23-26, Virginia Beach, VA, USA

for laparoscopic surgery training. This state modeling method proposes an object state model, the XML based task description and Bezier curve based guidance path generation methods. However, the object state detection method is not discussed in (Hong and Rozenblit 2016b). Utilizing live-camera image is one of the object state detection methods.

In this paper, we propose an image-based object state modeling method of a peg transfer task to support the CAST guidance system. This method aims to simulate the object states of the peg transfer task using the live camera image. The simulated object states will serve *opt*MIS to understand the system states and generate the dynamic optimal path. Also, the collection of these simulated object states will be used to model the user behavior, which is another useful reference information to support the guidance system.

The rest of this paper is organized as follows. Section 2 reviews the state description model of the peg transfer task and the related object tracking methods. Section 3 represents the framework for the proposed image feature-based object state modeling method. Section 4 is the simulation result and discussion. Section 5 concludes this paper and indicates the future research.

2 OBJECT STATE TRACKING OF PEG TRANSFER TASK

2.1 Peg Transfer Tasks

The peg transfer task is the first hands-on exam in the Fundamentals of Laparoscopic Surgery (FLS) (FLS 2017) program. FLS program was developed by the Society of American Gastrointestinal and Endoscopic Surgeons (SAGES). This program has multiple hands-on exams for trainees to learn the basic laparoscopic skills. There are five tasks of the exams: peg transfer, precision cutting, ligating loop, suture with extracorporeal knot, and suture with intracorporeal knot. For better understanding of the exam procedure for the peg transfer task, we suggest learning with a visual example **(**Sjhsurgery 2017).

The basic components of the peg transfer task include two surgical instruments (e.g., graspers), one pegboard with 12 pegs, and six rubber ring-like objects (triangles) as shown in Figure 1(a). Trainees need to manipulate an instrument to grasp a triangle from a peg, carry this triangle in mid-air, transfer it to the other instrument, and place the triangle on a peg on the opposite side of the board. In this task, trainees learn how to manipulate grasper-type instruments.

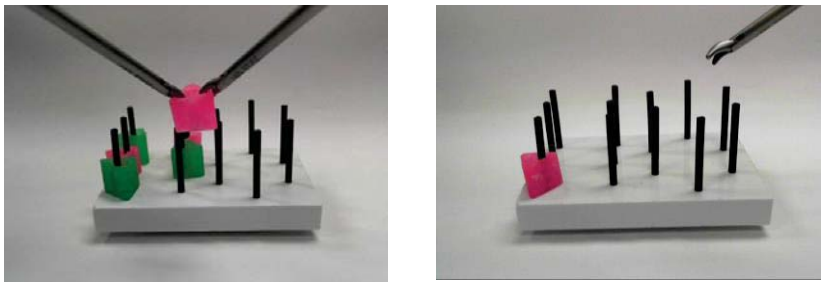
In the peg transfer task, CAST provides trainees visual and haptic guidance based on the system object's state (Hong and Rozenblit 2016b). Hong and Rozenblit generalized the movements of grasper-type instruments and modeled these actions as object states as follows:

- *Move* a grasper to a specific position.
- *Grasp* (pick up) an object using a specific grasper.
- *Carry* an object to a specific position using a specific grasper.
- *Transfer* an object from a specific instrument to the other grasper.
- *Place* (put down) an object at a certain location.

However, there are two states not considered in this setting. We supplement these two states into this action set as follows:

- *Stop* all the other actions.
- *Drop* the triangle from the grasper.

The action *stop* is the case that all objects are not moving. The action *drop* is the case that the triangle is dropped from the grasper. Adding these two supplemental actions models the states of the peg transfer task comprehensively.



(a) (b)

Figure 1: (a) FLS peg transfer task. (b) Simplified peg transfer task.

2.2 Object State Tracking

The topic of this paper is to detect the object states of the peg transfer task. In CAST, the live camera image contains the most comprehensive information of all the objects. We are interested in using this live camera image to track a moving object and understand its state. More specifically, this process is often called object tracking (Athanesious and Suresh 2012; Hu et al. 2004; Romero-Cano, Agamennoni, and Nieto 2015). In our application, we care about the relationship between the moving objects. For example, we can infer the *move* state when only the grasper is moving in the live camera image. Based on the survey done in (Athanesious and Suresh 2012), Silhouette Tracking is the effective way to extract the object region to model the state space model. To achieve real-time application, we prefer contour evolution rather than shape matching or learning, which needs the 3D template and complicated matching or learning process (Romero-Cano, Agamennoni, and Nieto 2015).

Object tracking is the process to detect the object features, which is a critical role in a successful tracking process. The unique features of an object can be distinguished easily in the image. Athanesious and Suresh (Athanesious and Suresh 2012) summarized several common features: color, edge, centroid, and texture. In our application, the color and centroid of the moving objects are the salient features that can help identify the objects and their relationships. Based on these features, the temporal differencing method reviewed in (Hu et al. 2004) is used to track the object motion. This method makes use of the binary result of the pixel-wise absolute difference between the previous and current frame images. Then the motion regions are clustered by the connected component analysis. The temporal differencing method is very efficient to detect the motion pixels but not robust to noise. We have set up a well-controlled environment for our peg transfer task to limit noise; hence, the temporal differencing method is effective in our application.

After detecting the object motion, the simulated object states can be modeled using an intelligent system. An Expected Association (EA) algorithm was proposed (Romero-Cano, Agamennoni, and Nieto 2015) to learn and classify the multiple moving objects. The EA algorithm is robust but computational. The model-based tracking reviewed in (Hu et al. 2004) effectively used a rule-based system to model the object state and then inferred the object states using the detected object behaviors. When we have a prior knowledge of the object behaviors, the object states are able to be modeled as a set of rules. The peg transfer task has predefined object actions and matches the criteria of the model-based tracking method.

2.3 System Configurations

To simplify the problem, only one right-hand grasper and one magenta triangle are considered in this paper. The user (trainee) needs to use a grasper to grab the triangle from one peg and then transfer it to another peg. During this operation, the user may accidentally drop the triangle. In this setting, we only consider the following object states: *stop, move, carry, and drop*. Besides, the image understanding process is sensitive to camera noise caused by the dark scene, background color, and environment

illumination. To provide a stable environment, we set up a uniform and sufficient light source for illumination with the color temperature of daylight. The background color of the environment is set to white. This simplified peg transfer task is shown in Figure 1(b).

3 OBJECT STATE DETECTION

To provide the object states for the peg transfer task in CAST, an object state model is required. This is inferred from the object actions detected in the live camera image. The Image Feature Based Object State Modeling (IFBOSM) defines the object states of the peg transfer task using image features extracted from the live camera image. In this section, we discuss the modeling method and the image processing algorithm for the peg transfer task.

3.1 Image Feature Based Object State Modeling (IFBOSM)

The object states are modeled using the image features. In the beginning, an image feature set is designed to represent each object. Then, the object states are modeled by the image feature set. This process is named Image feature Based Object State Modeling (IFBOSM). The IFBOSM provides a skeleton to develop the object state detection system. The block diagram is presented in Figure 2. In the following sections, we discuss the details of the implementation of this system.

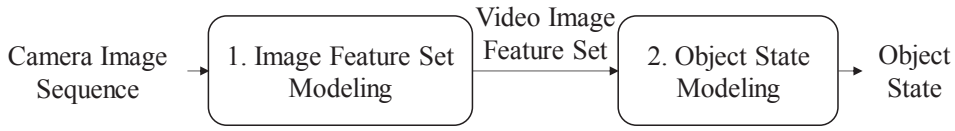


Figure 2: IFBOSM block diagram.

Before modeling the object states, we define a set of image features to represent each object and the relationship between two objects. When objects are moving, the temporal differences of the luma and chroma (Schalkoff 1989) between the previous and current image frames are the significant features that can be used to identify each object. The image features are presented as follows:

- Triangle: The key image feature of the triangle is the significant temporal chroma difference image (*dC*) of two continuous images. The triangle's color is magenta in this task and has more significant chroma information comparing to the grasper, whose color is silver. The *dC* can be used to represent the moving flag of the triangle (*Tm*). The *Tm* is formulated as the following equation.

$$Tm=f_t\,(dC)$$

  where, function f_t is the logistic function which will be elaborated later.

- Grasper: The key image feature of the grasper is the significant temporal luma difference image (*dY*) of two continuous images. Because the grasper's color is silver, the chroma information is very less (Schalkoff 1989). Although the triangle also has the temporal luma difference when moving, the temporal luma difference of the triangle can be removed by using the *dC* of the triangle. The *dY* without the triangle information ($dY \cap dC$) can be used to represent the moving flag of the grasper (*Gm*). The *Gm* is formulated as the following equation

$$Gm=f_g(dY - dY \cap dC)$$

  where, function f_g is the logistic function which will be elaborated later.

- Connection: When the triangle and grasper are both moving, they are connected when the distance between them is lower than a predefined threshold, which is determined by the CAST

system. The connection flag (*Cn*) of the triangle and grasper are defined by distance (*dist*) between the motion triangle and motion grasper. The feature *Cn* is formulated as the following equation

$$Cn=f_c(dist)$$

where, function f_c is the logistic function which will be elaborated later.

Based on the above definition, we define an image feature set *IFS(Tm, Gm, Cn),* to represent the movement of the triangle and grasper. However, the information of each image frame pair might be unstable due to the noise or tremble in the user's hand. The stabilized features in the video frame image are needed to be considered. Therefore, the video image feature set *VIFS(vTm, vGm, vCn)* is generated by the moving filtering process, which is a temporal noise reduction method (Schalkoff 1989).

Using *VIFS*, the object states are modeled as follows.

- *Stop*: Both the triangle and grasper are not moving.
- *Move*: The grasper is moving and the triangle and grasper are not connected.
- *Carry*: The triangle and grasper are both moving and the triangle and grasper are connected.
- *Drop*: The triangle is moving and the triangle and grasper are not connected.

3.2 Image Feature Set Modeling

An Image Feature Set Modeling method (IFSM) is proposed in this section. There are two major steps of this process – pre-processing and post-processing. The block diagram of IFSM is shown in Figure 3.

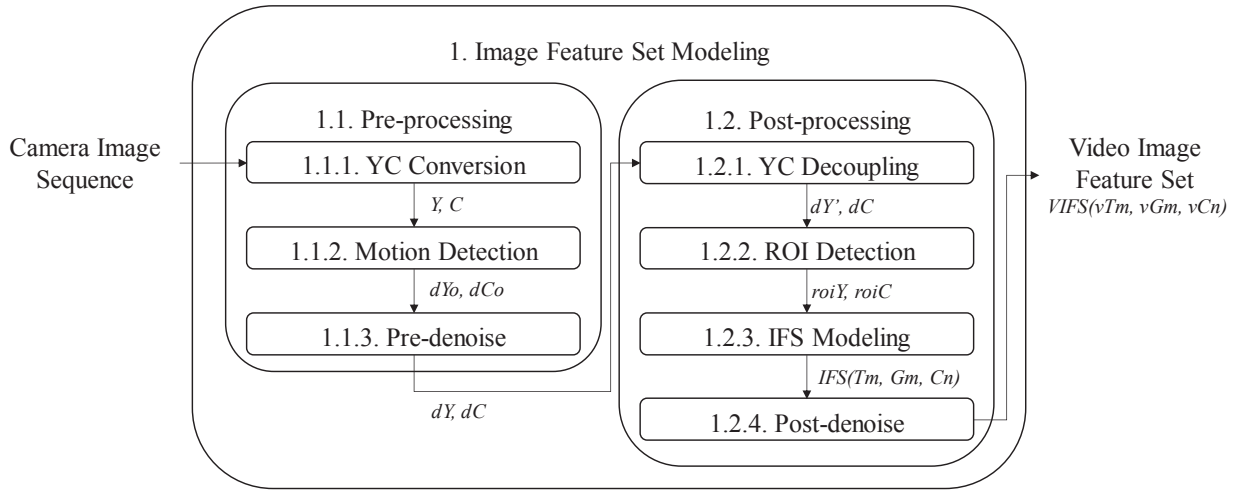


Figure 3: IFSM block diagram.

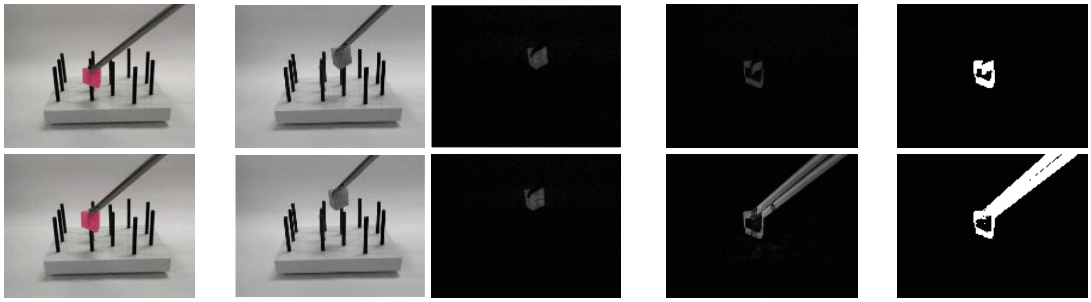
3.2.1 Pre-processing

The main purpose of preprocessing is to generate the *dY* and *dC* from the live camera images. There are two steps in pre-processing. First, the input image is converted to the luma (*Y*) and chroma (*C*) image pair. Then the *dY* and *dC* are calculated from the previous and current *YC* image pairs.

In the step of YC Conversion, the *YC* image pair is converted from the input image using YCbCr format (Schalkoff 1989). The *Y* image of *YC* image pair is the same as the *Y* image converted from the input image as shown on the left side in Figure 4(b). The *C* image of YC image pair is the combination of the *Cb* and *Cr* (Schalkoff 1989). The *C* image is defined by the following equation.

$$C=((Cb-128)^2+(Cr-128)^2)^{0.5} \quad (1)$$

The C image is shown on the right side in Figure 4(b). In the step of the motion detection, using the *YC* image pair, the original temporal difference image of luma (*dYo*) and chroma (*dCo*) are calculated using absolute difference between previous and current frame as shown in Figure 4(c). Then, the *dYo* and *dCo* are converted from the gray image to the binary image by the binary threshold (thr_{bin}) in the pre-denoise step. The thr_{bin} is selected as 8, which is the commonly-used maximal image noise level (Schalkoff 1989). The binary temporal difference images of *YC* image are as shown in Figure 4(d).



(a) (b) (c) (d)

Figure 4: IFSM Pre-processing results. (a) Camera image sequence, upper: previous image frame, bottom: current image frame; (b) *YC* image pairs after YC conversion, upper: previous *YC* image pair, bottom: current *YC* image pair; (c) Primitive motion image difference after Motion detection, upper: *dCo*, bottom: *dYo*; (d) Motion image difference after Pre-denoise, upper: *dC*, bottom: *dY*.

3.2.2 Post-processing

After the processed *dY* and *dC* image pair is generated, the image feature set – *IFS(Tm, Gm, Cn)* is extracted in the post-processing. The main process in the post-processing includes four steps – YC decoupling, region of interests detection, image feature set (IFS) modeling, and post-denoise.

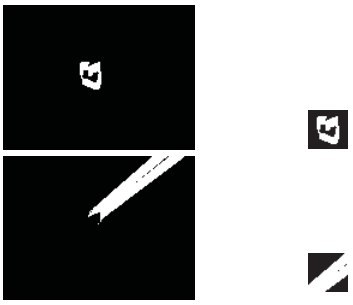
The final goal of the post-processing is to use *dY* and *dC* to represent the moving grasper and triangle. However, *dY* contains the motion information of both moving objects. The YC decoupling process is needed to remove the information of the moving triangle in *dY*. Next, in order to reduce the computation, we define a region of interest of *dY* (*roiY*) and a region of interest of *dC* (*roiC*). The *roiY* and *roiC* have the compact and complete information of the grasper and triangle. This information includes the motion speed and location of the moving objects. The motion speed of the grasper and triangle are referred by the summation of *dY* and *dC*. The location of the grasper and triangle are referenced by the centroid of *dY* and *dC*. Using the locations of these two objects, the distance between them are also available.

3.2.2.1 YC decoupling

The first step of the post-processing is YC decoupling of *dY* and *dC* image pair. As we mentioned in Section 3.1, the triangle moving (*Tm*) is generated from *dC*, while the grasper moving (*Gm*) is generated from *dY*, from which the triangle information ($dY \cap dC$) has to be removed. The *dY* is updated to *dY'* by the YC decoupling process shown as following equation.

$$dY' = dY - dY \cap dC \quad (2)$$

After YC decoupling, the *dY'* is the binary image containing the luma change of the moving grasper, and the *dC* is the binary image containing only the chroma change of the moving triangle. When the *dY'* and *dC* image pair is ready, the region of interest (*roi*) detection is able to be done in the next step.



(a) (b)

Figure 5: IFSM Post-processing results. (a) Results of YC Decoupling, upper: *dC*, bottom: *dY'*. (b). Region of interests after ROI Detection, upper: *roiC*, bottom: *roiY*.

3.2.2.2 Region of interests detection

The *roiY* and the *roiC* are the predefined size of area where the objects are located. The motion of the triangle is in a small range of *dC*, while the motion of the grasper is in a larger range of *dY'*. The region we are interested in is the grasper's front part, which directly interacts with the triangle. Then, we define the same size of *roiY* and *roiC*. When the grasper and triangle have similar motion behavior, *roiY* and *roiC* have the similar moving area.

The *roiY* and *roiC* are detected by using the contour detection method in OpenCV (Bradski and Kaehler 2008). There might be some noise or multiple regions of contours in *dY'* or *dC*, but we only select the contour with the largest area to be the region of interest. The moving area size of the selected contour also must be larger than a moving object contour threshold (thr_{mvc}), which is set as 50 in our application. The most-likely object contour of dY (*mlocY*) and dC (*mlocC*) are detected in this process. This method is the most-likely object contour detection (MLOCD).

$$mlocY = \text{MLOCD}(dY') \tag{3}$$

$$mlocC = \text{MLOCD}(dC) \tag{4}$$

The location and size of the minimum enclosing rectangle of *mlocY* (*rectY*) and *mlocC* (*rectC*) are also available. For the triangle, the whole object contour is important in our application, so we set the centroid of *rectC* as the centroid of the *roiC*. For the grasper, we are only interested in the motion of the grasper's front part. We select the left bottom corner area as the *roiY* because the grasper is moved from the right upper side in our application. This is the process of the region of interest extraction (ROIE). The *roiC* and *roiY* detection process are shown in Figure 5.

$$roiC = \text{ROIE}(mlocC) \tag{5}$$

$$roiY = \text{ROIE}(mlocY) \tag{6}$$

If there is no most-likely object contour detected, the centroid location of region of interest is set to be (0, 0) and its area size is set as 0. After the ROIE process, the *roiC* and *roiY* are available and we can model the image feature set *IFS(Tm, Gm, Cn)*.

3.2.2.3 Image feature modeling

The *IFS(Tm, Gm, Cn)* includes three logistic image features: moving triangle (*Tm*), moving grasper (*Gm*), and connection (*Cn*) between the triangle and grasper. The *roiC* and *roiY* provide the location and motion (white) pixel amount of the *dC* and *dY'* in the region of interests. The location of region of interest is the centroid of it. Using this information, the three image features are modeled as follows.

The *Tm* represented in the *roiC* and the *Gm* represented in the *roiY* are mentioned in Section 3.1. If the motion area size of the region of interest is larger than the motion pixel amount threshold (thr_{mpa}), the object is in the motion state. The logistic equation is defined as follows.

$$Tm = f_t\,(dC) = area(roiC) > thr_{mpa}\ ?\ True : False \tag{7}$$

$$Gm = f_g\,(dY - dY \cap dC) = area(roiY) > thr_{mpa}\ ?\ True : False \tag{8}$$

We use the same thr_{mpa} for both the *roiC* and *roiY* because they have the same dimension. The thr_{mpa} is defined as 100. The connection (*Cn*) between the triangle and grasper is defined by the distance (*distYC*) between the centroids of the *roiC* and *roiY*. When the *distYC* is less than the connection threshold (thr_{cn}), defined as 100 here, then *Cn* is set to be true. The logistic equation is defined as follows.

$$Cn = f_c(dist) = distYC < thr_{cn}\ ?\ True : False \tag{9}$$

In the case of any of the *area(roiC)* and *area(roiY)* being 0, then *Cn* is set as *False*.

3.3 Video image feature modeling

After the post-processing, the *IFS(Tm, Gm, Cn)* are modeled from the clear *dY* and *dC* image pair, but the *IFS* is unstable because of the noise from camera or hand tremor of the user. A temporal filter is applied on multiple frames (videos) to stabilize the *IFS* to get the video image feature set - *VIFS(vTm, vGm, vCn)*.

The temporal filter we use to stabilize the *IFS* is the moving median filter, which is a voting mechanism. We apply an odd number of frames of *IFS* queue and use the median value of this queue to replace the current *IFS*. All the image features of the *IFS* are logical values, and the median value of the data in the queue is the value of the majority. The *VIFS(vTm, vGm, vCn)* is formulated as follows

$$VIFS(vTm, vGm, vCn) = (\text{median}(\text{queue}(Tm)), \text{median}(\text{queue}(Gm)), \text{median}(\text{queue}(Cn))) \tag{10}$$

3.4 Object State Recognition

The *VIFS* provides the stable image features of the objects, and the object states defined in Section 3.1 can be represented using logic expression of *VIFS*. The rule of each state is expressed as follows.

$$Stop: \neg vTm \cap \neg vGm \tag{11}$$

$$Move: vGm \cap \neg vCn \tag{12}$$

$$Carry: \text{v}Tm \cap vGm \cap vCn \tag{13}$$

$$Drop: \text{v}Tm \cap \neg vCn \tag{14}$$

4 SIMULATION RESULTS

The proposed algorithm is simulated using C++. To evaluate the performance of our method, we use a practical case of image sequence which has 1000 frames. This image sequence covers all kind of states of the simplified peg transfer task: *stop*, *move*, *carry*, and *drop*. To quantitatively measure the result, we use the positive predictive value of each state in frame-based count and action-based count. We present the detailed settings and evaluation results in this section.

4.1 Evaluation configuration and metrics

The hardware configuration of the simulation system includes Intel i3-4160T 3.1 GHz CPU, Intel HD Graphic 4400 CPU, 16 GB DDR3 RAM, and 512 GB SSD. The camera is the Microsoft LifeCam Studio. The functions of auto focus, auto exposure, and auto white-balance are all disabled. The white-balance is fixed to 3500 Kelvin and illumination of the environment is set to a fixed brightness.

The software configuration of our method is described here. The resolution of the test image sequence from the camera is 1280 by 720 pixels and the frame rate is 30 frame per second (fps). To reduce the computation power, we downscale the image to 160 by 90 pixels to be the system input. In the ROI-detection process, the dimensions of *roiC* and *roiY* are both 32 by 32 pixels. Based on the experimental results, this dimension covers most cases of the moving object. In the Post-Denoise process, the lengths of the three queues are set as 5. In the Image Feature Set Modeling process, all the thresholds are determined by experimental results. All these parameters are fixed in the simulation process.

The test image sequence of the simplified peg transfer task is designed to cover all the object states. There are 1000 frames in this sequence. The frame count is the total frame amount of each object state, while the action count is the total action amount. For example, an action of the drop state may have around 4 frames and in total we perform 10 actions of the drop state. In the drop state, the action count is 10, and the frame count is around 40. The frame count and the action count of each object state in the test image sequence are identified manually and summarized in Table 1.

The quantitative evaluation is done by computing the positive predictive value (*PPV*) as follows.

$$PPV = \frac{TP}{TP+FP} \tag{15}$$

where *TP* is the total amount of true positive, and *FP* is the total amount of false positive. We consider the *PPV* of the frame count (*PPV_fc*) and the action count (*PPV_ac*) both. In the next section, we present the evaluation results of the test image sequence.

Table 1: Positive predictive value (PPV) of the test image sequence.

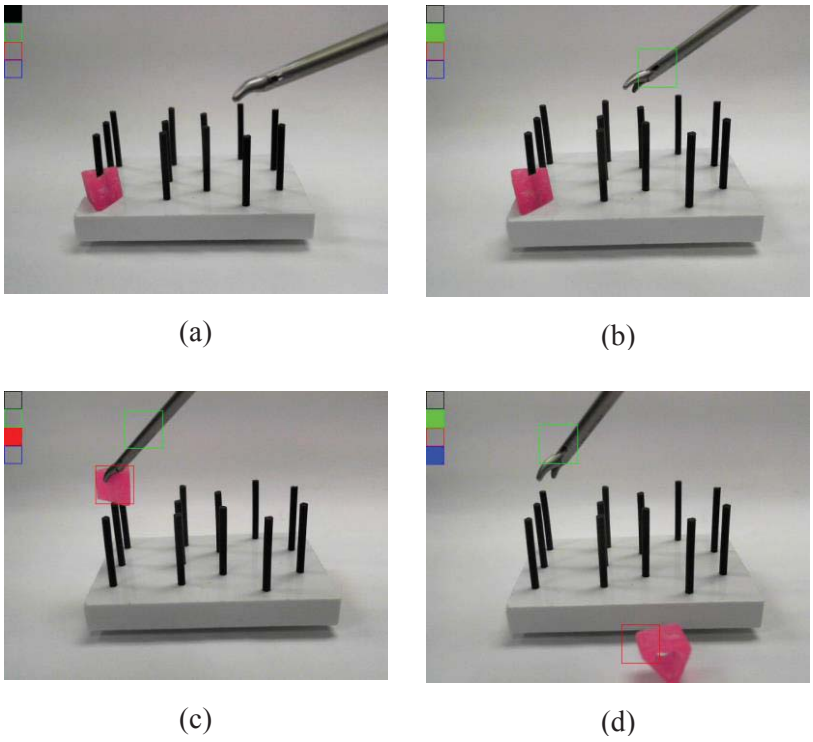
States		**Stop**	**Move**	**Carry**	**Drop**
Frame count	*Truth*	20	348	594	38
	TP_fc	20	288	557	20
	FP_fc	35	10	58	12
	PPV_fc	0.364	0.966	0.906	0.625
Action count	*Truth*	3	22	22	10
	TP_ac	3	22	22	8
	FP_ac	13	6	1	2
	PPV_ac	0.188	0.789	0.917	0.8

4.2 Evaluation result and Discussion

The computational efficiency is evaluated by monitoring the computation time of each frame in our system. The evaluation image sequence with the visualized label of each state is available on our website. (http://mbdl.arizona.edu/projects/computer.assisted.minimally.invasive.surgery.html). In our result, the average computation time is less than 20msec per frame, which is less than the camera capturing time (33 msec of 30 fps). Our method is feasible for real-time implementation.

In the simulation, the simulated object states are visualized as shown in Figure 6. There are 4 color boxes to be used as the indicators of the object states. The black, green, red, and blue boxes are fulfilled when the object states are *stop, move, carry*, and *drop*. The *stop, move,* and *carry* are mutually exclusive. The *drop* and *move* may happen at the same time. Using these visual indicators, the user and the supervisor can easily monitor the object state.

The quantitative evaluation of the performance is represented in Table 1, which shows the *PPV_fc* and *PPV_ac* from the result of simulation. The detection result of the carry state is over 90% in both frame count and action count, while result of the stop state is less than 40%. The *PPV_fc* of the move state has the best performance of the frame count, but the *FP_fc* causes significant effect on the *PPV_ac*. On the contrary, although the *PPV_fc* of the drop state is only 62.5%, the *FP_fc* only causes 2 counts in *FP_ac* and the *PPV_ac* is 80%. In the *action* count, our method is able to correctly identify the *move, carry,* and *drop* state, but the detection of the *stop* state has false positive cases, also known as false alarms.



(a) (b)

(c) (d)

Figure 6: IFBOSM simulation results of each object state. (a) Stop: The black box is fulfilled as the indicator. (b) Move: The green box is fulfilled as the indicator. (c) Carry: The red box is fulfilled as the indicator. (d) Drop: The blue box is fulfilled as the indicator.

The major issue in performance is the false alarms. The main factor identified for this issue is the operational noise, which is the hand tremor of the user. Although we have applied the temporal filter to stabilize this effect, the performance is limited due to the length of the queue. Long queue causes longer delay, but the filtering effect is weak in the short queue. Based on our experiment, the optimal length of the queue is 5. However, the case of the slow moving object suffers the effect of hand tremor. The image feature set may get cross the threshold back and forth frequently. It jumps in and out of the stop state in the detection. This is the root because of the false positive in the *stop* state. A possible solution is to consider the dynamic queue and threshold to adapt the case of the slow moving object.

To improve the performance of other states, false alarm cases have to be reduced. The proposed method is a rule-based intelligent system using the image feature sets which are based on the fixed thresholds from the experimental results. The fixed thresholds have their limitation to fit all cases of the states. For example, the image features of near and far away the camera are different. The thresholds should be able to adapt to such differences. Although we would like to use an adaptive method to improve the accuracy of the detection, we do not have any information of the exact location of the objects in the live camera image to modify the thresholds. The possible solution to improve the quality of image feature sets is to consider the learning-based method to learn the object location and the associated threshold. False alarms during detection can be improved with this.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose an IFBOSM method to simulate the system object state of the simplified peg transfer task in the FLS. The major benefit of this method is to provide the framework of the simulated object state model for CAST. This method detects the object states based on the image feature sets and the rule-based intelligent system. The temporal difference of the moving objects in the image are modeled as the image feature sets and this method uses no object. The computation time of the proposed system is less than 20msec per frame and is able to be applied to the real-time application. The *PPV* of the state detection of move and carry state in frame count is over 90%, and in action count, it is around 80%. The weak points are the cases of false alarms. The root causes are the operational noise of the slow motion and the lack of the feasible threshold values when the objects are in different locations. In the future, we will focus on dynamic filtering and learning-based object state detection to improve the current weakness. Although the IFBOSM method is currently implemented in the simplified peg transfer, it has great potential to be extended to the complete peg transfer task and other tasks in the FLS.

ACKNOWLEDGMENTS

This work has been supported by the National Science Foundation grant no. 1622589, "Computer Guided Laparoscopy Training".

REFERENCES

Athanesious, J.J. and Suresh, P. 2012. “Systematic Survey on Object Tracking Methods in Video.” *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, pp.242-247.

Bradski, G. and Kaehler, A. 2008. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc.

Sjhsurgery 2017. “FLS Expanded Video Tutorial Series: Task 1 - Peg Transfer.”. Accessed January 03, 2017.https://www.youtube.com/watch?v=gAQPXHWqdXQ&list=PLAW27YzXUvRtxhCqs2uEB9xWMBCajol_X

FLS 2017. “Fundamentals of Laparoscopic Surgery.” Fundamentals of Laparoscopic Surgery. Accessed January 03, 2017. http://www.flsprogram.org/.

Hong, Minsik, and Jerzy W. Rozenblit. 2016a. "A haptic guidance system for Computer-Assisted Surgical Training using virtual fixtures." In *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*, pp. 2230-2235.

Hong, M. and Rozenblit, J.W. 2016b. “Modeling of a Transfer Task in Computer Assisted Surgical Training.” *Proc. of the 2016 Spring Simulation Conference, Modeling and Simulation in Medicine*, pp. 794-799.

Hu, W., Tan, T., Wang, L., and Maybank, S. 2004. “A survey on visual surveillance of object motion and behaviors.” *IEEE Transactions on Systems, Man, and Cybernetics*, Part C, vol.34, no. 3, pp. 334-352.

Napalkova, L., Rozenblit, J.W., Hwang, G., Hamilton, A.J. and Suantak, L. 2014. ”An Optimal MotionPlanning Method for Computer-Assisted Surgical Training." *Applied Soft Computing* vol. 24, pp. 889-899.

Nikodem, J., Wytyczak-Partyka, A., and Klempous, R. 2015. “Application of Image Processing and Virtual Reality Technologies in Simulation of Laparoscopic Procedures.” *International Conference on Computer Aided Systems Theory*, pp. 463-470

Romero-Cano, V., Agamennoni, G. and Nieto, J. 2015. “A variational approach to simultaneous multi-object tracking and classification.” *The International Journal of Robotics Research*, pp.654-671.

Rozenblit, J.W., Feng, C., Riojas, M., Napalkova, L., Hamilton, A.J., Hong, M., Berthet-Rayne, P., Czapiewski, P., Hwang, G., Nikodem, J. and Shankaran, A. 2014. "The Computer Assisted Surgical Trainer: Design, Models, and Implementation." *Proc. of the 2014 Summer Simulation Conference, Modeling and Simulation in Medicine*, pp. 211-220.

Shankaran, Akash, and Jerzy W. Rozenblit. "Augmented reality visualization for computer assisted surgical training." In *International Conference on Computer Aided Systems Theory*, pp. 183-190. Springer Berlin Heidelberg, 2013.

Schalkoff, R.J. 1989. *Digital image processing and computer vision.* vol. 286. New York: Wiley.

AUTHOR BIOGRAPHIES

KUO SHIUAN PENG is a Ph.D student of Electrical and Computer Engineering at the University of Arizona. He earned a MS in Power Mechanical Engineering from National Tsing Hua University in Taiwan. His research interests lie in image processing, computer vision, artificial intelligence and machine learning, especially in intelligence modeling. His email address is kspeng@email.arizona.edu.

MINSIK HONG is a Ph. D. candidate at the University of Arizona. He received a Master of Science degree in Electrical and Computer Engineering from POSTECH, Republic of Korea. His research interests are robotics, control system, fuzzy theory, and modeling and simulation for medical devices. His email address is mshong@email.arizona.edu.

JERZY ROZENBLIT is University Distinguished Professor, Raymond J. Oglethorpe Endowed Chair in the Electrical and Computer Engineering (ECE) Department, and Professor of Surgery in the College of Medicine at the University of Arizona. From 2003 to 2011, he served as the ECE Department Head. During his tenure at the University of Arizona, he established the Model-Based Design Laboratory with major projects in design and analysis of complex, computer-based systems, hardware/software codesign, and simulation modeling. His email address is jr@email.arizona.edu.