Vision Guidance for Automated Harvesting of Malaria Vaccine from Live Mosquitos

Final Report

Team Members

Akash Chaurasia
achaura1@jhu.edu

Alan Lai
alai13@jhu.edu

Parth Vora
pvora4@jhu.edu

Mentors

Dr. Russell Taylor
rht@jhu.edu

Balazs Vagvolgyi
balazs@jhu.edu

May 5, 2020

Acknowledgements

We would like to thank our mentors Dr. Russell Taylor and Balazs Vagvolgyi, as well as other members of the LCSR-Sanaria team, for their support with this project. Their help and guidance were instrumental in achieving our goals. We would also like to acknowledge the role of Sanaria, Inc. (Rockville, MD) in this project, and thank them for their support.

Purpose

The goal of this project is to create a ROS-integrated computer vision system for mosquito detection, mosquito orientation classification, and mosquito pose estimation to guide an automated mosquito dissection robotic system for live malaria vaccine production. The key points on the mosquito to be detected via pose estimation include the proboscis (tip and end), the head, and the neck. The relevant parts of the mosquito are labeled in figure 1. below.

Background & Relevance

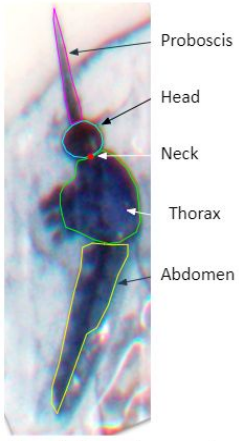

Figure 1. Parts of a mosquito

Malaria is a mosquito-borne disease that affects humans and is caused by a single-celled organism of the Plasmodium group. Spread by mosquitoes carrying the parasite, symptoms include fever, tiredness, vomiting, headaches, seizures, and eventually death. In 2017, there were over 200 million clinical cases of malaria, causing over 435,000 deaths, and over $12 billion USD loss in Africa alone.[1] Despite the impact that malaria has and the clear need, there currently exists no effective malaria vaccine in the market. However, Sanaria, a biotechnology company based in Rockville MD, has recently been successful in developing a live malaria vaccine that has shown to be up to 100% effect in clinical trials. These vaccines are made from attenuated Plasmodium falciparum sporozoites (PfSPZ), which is the most common parasite that causes malaria. Because of the nature of these live vaccines, they must be cultivated within live mosquitoes, and hence must also be extracted from mosquito salivary glands before being able to be used as a vaccine.

Currently, the workflow to create this vaccine is tedious and processing is slow, requiring manual extraction of the attenuated PfSPZ from salivary glands using syringes. In order to scale up production, manual operations that constrain the production of the vaccine must be replaced by automatic systems. Previous work has been done by Schrum et al. to create a semi-automatic system for mosquito dissection.[2] Though training time decreased substantially and processing capacity increased by at least two fold, the process was still manually tedious and labour intensive. Hence, an autonomous robotic system is currently being developed by LCSR in order to automate the process. The full workflow for automated PfSPZ extraction is outlined in Figure 2.

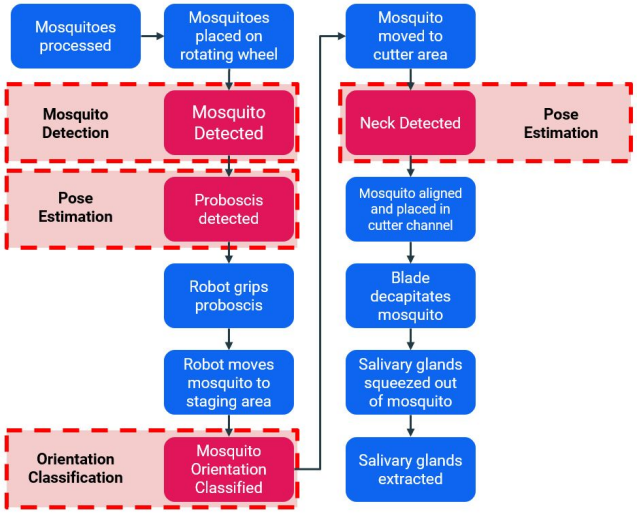

Figure 2. Process flow diagram for mosquito keypoint localization

The red states in the diagram demonstrate areas in the process flow where our computer vision system will be used to support the robotic system. The mosquitoes are first processed and placed on the rotating wheel, where the mosquito detection and pose estimation algorithms will be used to locate the mosquito and the proboscis (the magenta outlined area in figure 1). The robot grabs the mosquito by the proboscis, and moves it to the staging area, before the cutter. The orientation of the mosquito is classified, so that the cutters will be able to move in the correct direction to maximize yield. Once the mosquito is moved to the cutter area, the neck is detected, and the robot aligns the neck with the cutter, ensuring that when decapitation occurs, the salivary glands are exposed. Finally, the salivary glands will be squeezed out of the mosquito, and the vaccine is then produced.

Our project focuses on leveraging computer vision techniques to allow the robot to autonomously detect keypoints on the mosquito. Locating the position of the proboscis will enable the robotic system to grab the mosquito to manipulate and move it, while locating the position of the neck will enable the robotic system to align the mosquito correctly with the cutter. This will allow the robot to optimally position the mosquito for decapitation and salivary gland extraction while minimizing human input.

Previous work has been done by previous members of our lab, reported in the paper by Wu et al., that used deep learning and image processing for the detection of mosquito and mosquito pose estimation.[3] With regards to deep learning, convolutional neural networks such as Mask R-CNN and DeeperCut were used for mosquito detection and pose estimation, while for the image processing pathway, a multi-step process was used. Though the problem definition that the authors addressed is largely similar to our current problem, the change in robotic system setup and lack of reproducibility of results due to lack of documentation meant that our project had to essentially start from scratch. Despite this, the paper was able to provide us with some reference to the techniques used, and some evaluation results that we were able to compare our algorithms with.

Technical Approach

Image Processing vs Deep Learning

Both image processing and deep learning methods were used for this project. While both methods are established in the computer vision field and are capable of solving each of the vision problems within our project, each has advantages that we leverage to achieve optimal results.Image processing algorithms are typically fast and able to solve well specified imaging problems, but are often variant to varying conditions such as lighting and scale. On the other hand, deep learning algorithms are more flexible and are able to solve problems that cannot easily be described by rules and have more uncertainties, but require large amounts of both data and computational resources to run. Throughout this project, we have been constantly comparing these two methods to determine the optimal combination of both image processing and deep learning algorithms to create a robust and accurate computer vision system.

Image Processing

The main goals of the model-based approaches are to (1) detect and locate mosquitos, (2) locate the center of the neck, and (3) locate the center of the proboscis. The goals of each of these steps is to find where the mosquitos are on the rotating wheel, determine how far the robot must drag the mosquito to align it with the blade cutter, and determine where to grab the mosquito respectively. Each of these steps will be performed using traditional computer vision methods. Algorithms will be implemented in C++ and integrated with the robotic system using ROS.

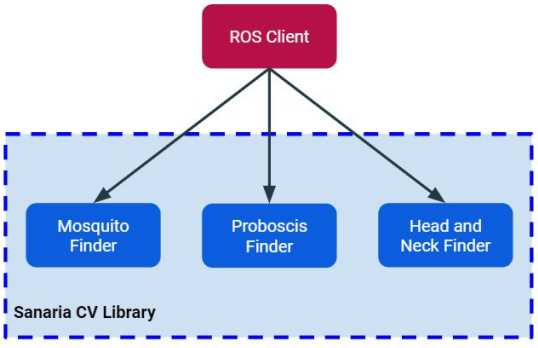

Figure 3. Process flow diagram for mosquito keypoint detection

Mosquito Finder

The mosquito finder algorithm uses binary image processing and connected component labelling to identify objects in the image. Mosquitos are filtered based on their orientation and size.

Detection threshold: Intersection over Union (IOU) of 0.4
Target success rate (% detected): ≥95%

Neck Finder

The neck finder algorithm first uses template matching to locate the head using two manually extracted and processed templates. Then, it identifies the neck as a preset distance below the center of the head. It is assumed that mosquitos will be oriented vertically with the proboscis pointed towards the top of the image, as the mosquito in Fig. 1 is oriented. Since the neck finder is used to find the vertical displacement required to align the mosquito neck with the blade, only the error in the y-axis is considered for successful detection. This algorithm has implemented checks

for error cases, such as missing head, primarily by checking the normalized cross-correlation score used from template matching.

Detection threshold: y-axis error ≤30 pixels
Target success rate: ≥95%

Proboscis Finder

The proboscis finder algorithm first uses template matching to locate the head. Then, it takes a region around the head and warps it to polar coordinates. The angle of the proboscis is identified by the calculating row averages in the polar warped region (each row is one degree, radially from the center of the head). Using this angle and a preset radius, the image coordinates of the proboscis are then computed. The proboscis finder has multiple checks for error cases, including missing proboscis to improve detection accuracy and prevent false positives. These checks are implemented using averages over the searched region (to ensure that the dark mosquito parts are indeed present in the image) and checks over the values used in the local optimization.

Detection threshold: lateral projection error ≤30 pixels
Target success rate: ≥95%

Deep Learning

The main goals of the deep learning approaches for this project are to (1) classify the orientation of the mosquito, (2) detect and locate the mosquitoes, and (3) locate keypoints of the mosquito, which include the proboscis tip, the proboscis end, the head, the neck, the thorax, and the abdomen. The process of creating deep learning neural network models and using them in practice is outlined below in figure 4.

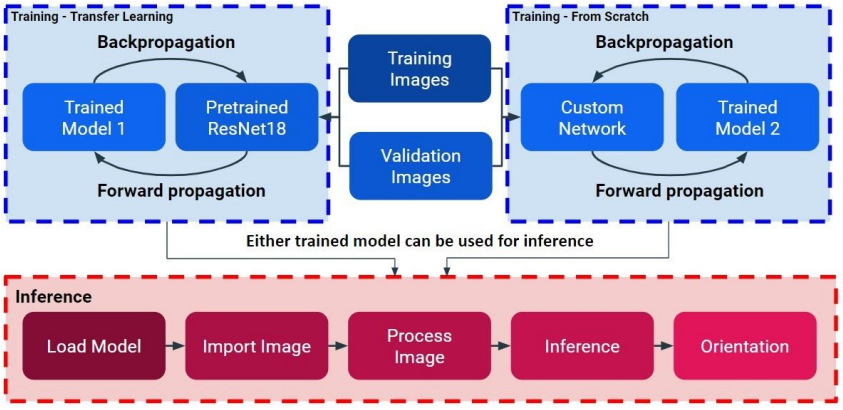

Figure 4. Flow diagram for deep learning training and usage

Deep learning can occur via either training from scratch, or transfer learning. Training from scratch (shown in figure 4 as the right light-blue shaded box) occurs when a network model is initialized from random weights, and training images and validation images are provided to the network, where a series of forward propagation

and backpropagation are performed so that the network is able to learn features of the image that it uses for classification. Though this specializes the model for our purposes, training is slow, and because of our limited dataset, accuracy is low.

Therefore, transfer learning was used for this project, which involves taking a pretrained network (shown in figure 4 as the left light-blue shaded box) and fine-tuning it. By pretraining the network on other images, the neural network already has a representation of features of images that it can use for classification, speeding up the training process. The fine tuning process outlined in the image by using our training and validation dataset allows us to modify the model weights such that it can be used for our purposes.

Once the model has been trained, inference can occur, as shown in the light-red shaded box in the bottom of figure 4. The model is loaded, and the relevant image is imported and processed. Inference occurs when the image is fed into the trained neural network, where it will output the classification or other predictions.

Orientation Classification

The orientation classification model aims to classify the mosquito as either missing, lying on its back or stomach, lying on its left side, or lying on its right side, as seen in figure 5. Transfer learning of neural network models was performed via PyTorch. Pretrained models such as ResNet18, ResNet152, VGG16, and DenseNet121 were loaded from PyTorch, and trained on our orientation image dataset. Conditions such as data augmentation, learning rate, epoch number, and optimizers were varied to determine the best combination that would yield the highest accuracy.

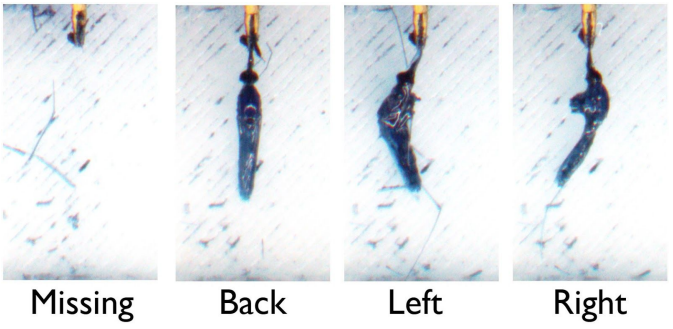

Figure 5. Orientation classification classes

- Target accuracy: 95%
- Target running time: < 1000 ms

Mosquito Detection

Transfer learning was also performed for our mosquito detection task, which involves locating the mosquito in the image. The Detectron2 library framework, made by Facebook AI and based on Mask R-CNN, was used for training, as it hosts a large number of pretrained models, and supports object detection, as required for this project. To use this model, Detectron2 models were loaded from the library, and training and validation images from the old system setup were used to train the model via the framework. Once training was completed, testing images were then provided to the model, where evaluation was performed.

- Target mAP@IoU=0.5: 0.96
- Target mAP@IoU=0.75: 0.84
- Target running time: < 400 ms

Mosquito Pose Estimation

For mosquito pose estimation/keypoint detection, the Detectron2 library was also used, as it supported human pose estimation, which was re-purposed to be applicable for mosquito keypoints. Again, Detectron2 models were loaded from the library, and training and validation images from the old system setup were used to train the model via the framework. Once training was completed, testing images were then provided to the model, where evaluation was performed.

- Target average RMSE for all body parts: 4 pixels
- Target running time: < 400 ms

Materials & Methods

Image Processing

Image processing algorithms were validated on 250+ mosquitos placed on a turntable, with manually annotated bounding boxes, necks, and proboscis. The bounding boxes generated by the mosquito finder algorithm were validated by computing the intersection over union (IOU) relative to the ground truth, with IOU of 0.4 our threshold for success. The proboscis finder algorithm was validated by computing the lateral error relative to the line of the proboscis in the ground truth, with an error of 30 pixels our threshold for success. The neck finder algorithm was validated by computing the error in the vertical axis relative to the ground truth, with an error of 30 pixels our threshold for success.

Deep Learning Datasets

Three different datasets were used for deep learning methods. The first dataset was used for orientation classification, which included 433 images of single mosquitoes in the staging area (shown also in figure 5). These images were annotated by previous members of our lab according to their orientation classes. The images were split 70-30 for training and validation, respectively.

The second dataset included the same 126 images used for image processing validation, where single or multiple mosquitoes were placed onto the turntable. The mosquito bounding boxes, and the proboscis and neck positions were manually annotated. Though ideally this dataset would have served as the training and validation set for the mosquito detection and pose estimation deep learning models, the dataset was limited by COVID-19. Because of the limited number of images, training was done with the third dataset of old system setup images, and testing was done using this dataset.

The third dataset included 996 images of the old system setup, of which 787 was used for training, and 209 was used for validation. These images were taken by previous members of our lab, and were also annotated by previous members. The annotations include the bounding boxes, and the six keypoints to be predicted (proboscis tip, proboscis end, head, neck, thorax, abdomen). These were also the images used for training and validation by Wu et al, and hence provides a unique opportunity to compare results with previous work done.

Deep Learning Evaluation

Evaluation of orientation classification was done by computing the number of correct classifications compared to the total number of images. For mosquito detection, bounding boxes were evaluated via mean average precision, with IOU threshold to be at 0.5 and 0.75 when comparing with the ground truth. With regards to mosquito pose estimation, the root mean squared error distance was used to compute the errors between the predicted and ground truth keypoints of the mosquito for the validation and testing images. Further computation of the lateral error of the proboscis and the vertical axis error for the neck (the same metrics used for image processing) was performed for testing images to allow for a fair comparison between image processing and deep learning approaches.

Results & Discussion

Image Processing

The results for the mosquito finder, neck finder, and proboscis finder algorithms are summarized in Table 1, and visualized in figures 6 and 7. Overall, a success rate of >95% was achieved for each algorithm on the turntable image set.

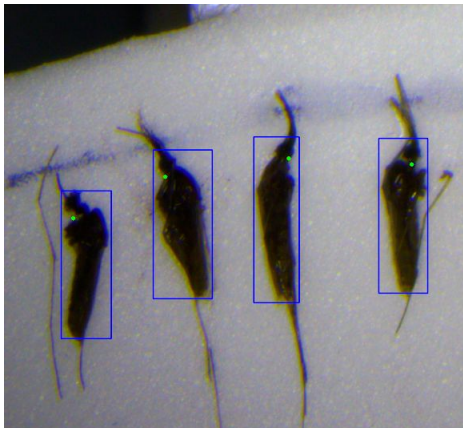

Figure 6. Neck finder results (green dots show point of detected neck)

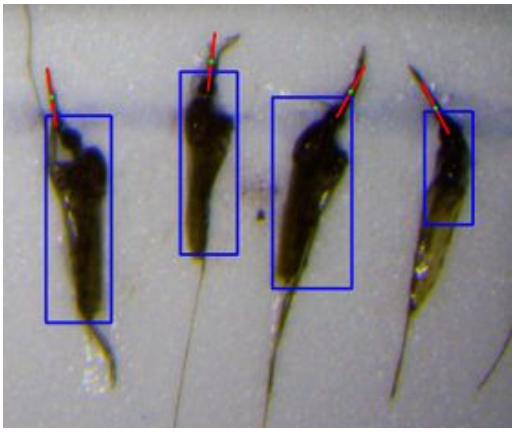

Figure 7. Proboscis finder results (red line overlaid on detected proboscis, green dots show midpoint of proboscis)

Table 1. Results for image processing algorithms.

Mosquito Finder		**Proboscis Finder**		**Neck Finder**	
# Mosquitos	306	# Proboscis	261	# Necks	272
# Found	291 (95%)	# Found	251 (96%)	# Found	259 (95%)

Avg. IOU	0.736	Avg Lateral Error (pixels)	3.46	Avg y-axis Error (pixels)	10.91
STDEV	0.126	STDEV	3.39	STDEV	7.62

All three image processing algorithms were able to achieve the target success rates of 95% with reasonable errors for each case. Visually inspecting the results show that the algorithms are fairly capable of detecting mosquitos and finding their neck/proboscis. Furthermore, the checks implemented for error cases were used during validation, since missing proboscises were reported as such, which increased the overall success rate and reduced the average errors.

For proboscis finder, the error threshold was set to 30 pixels since this is approximately equal to the width of the open jaws of the gripper which grasps the proboscis, and thus serves as an upper bound for the error. We see that the algorithm was able to achieve an average error far lower than this upper bound.

Deep Learning

Orientation Classification

Four different PyTorch models were used for orientation classification. Different training parameters were varied, including feature extraction (all weights except for last layer fixed during training) vs fine tuning (no fixed weights during training), data augmentation schemes (either no augmentation, or random crop and normalization), learning rate schemes, and optimizers. The optimal training parameter settings were fine tuning, data augmentation with random crop and normalization, stochastic gradient descent (SGD) optimizer with momentum, and a step learning rate scheme with step size of 7 and gamma of 0.1. The results of the different models trained using these parameters are shown below in table 2.

Table 2. Results for Mosquito Orientation Classification via Deep Learning

Model	Accuracy	System	Inference (s)	Inference + Processing (s)
ResNet18	96%	CPU	0.102	0.104
ResNet18	96%	GPU	0.004	0.005
ResNet152	97%	CPU	0.588	0.590
ResNet152	97%	GPU	0.028	0.029
VGG16	94%	CPU	0.614	0.616
VGG16	94%	GPU	0.016	0.017
DenseNet121	96%	CPU	0.297	0.299
DenseNet121	96%	GPU	0.022	0.023

Overall, we see that ResNet152 (red text), with 97% accuracy, performed the best for the task of orientation classification. When running on the GPU, the total

inference time (time for network to classify image) and processing time (time taken to normalize the image) for the best model took 29 ms. Both our targets for the orientation detection in terms of accuracy and processing speed were achieved.

Mosquito Detection

Multiple Mask R-CNN and Faster R-CNN models were trained via Detectron2. Validation results were compared to those from Wu et al., shown below in table 3.

Table 3. Results for Mosquito Detection via Deep Learning

Model	mAP@IoU=0.75	mAP@IoU=0.5	Inference (s)
Wu et al.	0.84	0.96	0.400
Mask R-CNN R50 FPN	0.894	0.975	0.100
Faster R-CNN R50-C4 VOC	0.895	0.981	0.200
Faster R-CNN R50 FPN	0.895	0.970	0.060

The best model trained was Faster R-CNN R50-C4 VOC with 15000 epochs, achieving an mAP@IoU=0.75 of 0.895, and an mAP@IoU=0.5 of 0.981, outperforming the model trained by Wu et al. in both metrics. mAP@IoU=0.5 for this problem can be thought of as the proportion of mosquitoes detected, and the mAP@IoU=0.75 can be thought of as how close the bounding box is to the ground truth. Our model also has a faster processing speed, and was able to achieve both targets for mosquito detection.

Mosquito Pose Estimation

One pretrained model was used from Detectron2. The root mean squared error metric was used for each of the keypoints, and the results were compared to the results reported by Wu et al. with regards to the validation set, shown in table 4.

Table 4. Results for Mosquito Orientation Classification via Deep Learning

Model	Wu et al.	R50-FPN Detectron2	
Dataset	Validation	Validation	Testing
Proboscis Tip Error	7.1	7.17	22.15
Proboscis End Error	2.2	1.91	16.64
Head Error	1.6	1.40	N/A
Neck Error	2.0	1.89	44.70
Thorax Error	2.6	2.71	N/A
Abdomen Error	3.0	3.68	N/A
Average Error	3.08	3.12	N/A

We see that the proboscis tip error was the highest at 7.17 pixels, while the other keypoints averaged around 2 to 3 pixels. This reflected the results reported by Wu et al., and we see that the average error across all keypoints were extremely similar, with 3.08 for Wu et al., and 3.12 for our Detectron2 model. For the validation images, one pixel represents around 0.05 mm, and hence the proboscis tip error

represents around 0.35 mm error. With regards to the testing set, we see that the average errors are much higher, with around 20 pixel error for the proboscis and 45 pixel error for the neck.

Sample images with the predicted bounding boxes (blue), ground truth bounding boxes (green), and predicted keypoints connected and drawn are shown below.

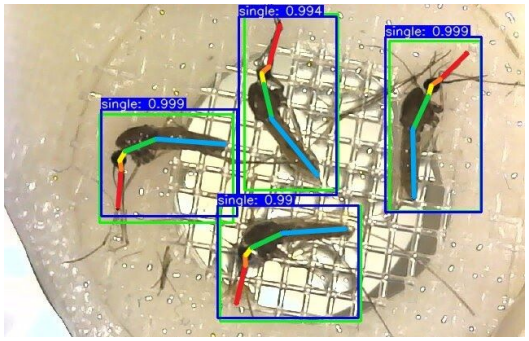

Figure 8. Pose Estimation results on old setup image with keypoints connected

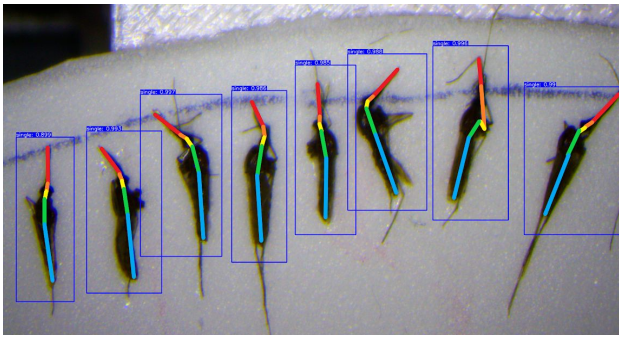

Figure 9. Pose Estimation results on new setup image with keypoints connected

Comparison between Image Processing and Deep Learning

Table 5. Comparison of Results for Deep Learning and Image Processing

	Image Processing	Deep Learning
Mosquito Detection		
# Found (threshold = 0.7 IoU)	291 (95%)	306 (100%)
Average Runtime (ms)	29	200
Proboscis Detection (261 total)		
# Found (threshold = 30 pixels)	251 (96%)	229 (88%)
Average lateral error (pixels)	3.46	8.25
Average Runtime (ms)	41	106
Neck Detection (272 total)		
# Found (threshold = 30 pixels)	259 (95%)	248 (92%)
Average y-axis error (pixels)	10.91	8.49
Average Runtime (ms)	4.7	106

We see with regards to mosquito detection, deep learning outperforms image processing, as it was able to detect all mosquitoes within the testing images, despite being trained on images of the old system. However, with proboscis and neck detection, image processing outperforms deep learning, achieving a higher number of keypoints found within the threshold, and having a lower lateral error for proboscis detection. Y-axis error for neck detection was around the same for both methods. This suggests that deep learning pose estimation requires more work to be able to match the results of image processing. Finally, image processing runtime outperforms that of deep learning, on the order of tens of milliseconds instead of hundreds of milliseconds, which was expected. However, given the problem context, both models

are fast enough to be integrated with the robotic system without being a bottleneck for the process.

Conclusion and Significance

Overall, our project was able to support the vision tasks required for the automated robot mosquito dissection system. With the developed algorithms and models, the steps of finding mosquitos on the turntable, finding the mosquito's proboscis, detecting the mosquito's orientation, and finding the mosquito's neck, which all previously required manual input, can now be done in fractions of a second without human interaction. As shown by the results, the methods achieve strong success rates of 100% mosquito detection, 96% proboscis detection, and 95% neck detection (best case across both approaches). Thus, with full system integration, the robotic system should be able to rely on these vision methods to minimize human input needed. As indicated by our project advisors, there are more steps in the workflow which require vision guidance, such as verifying decapitation and cleaning. Though these tasks were not included in the scope of this project, our group will work to address these challenges as well to further streamline the autonomous workflow.

COVID-19 posed a large barrier to portions of our project, specifically testing our methods directly on the robotic system and capturing new images to be used for model training and algorithm validation. Thus, once we are able to capture new images, we hope to include these in new, larger datasets which reflect the updated robotic system as well as provide real-world validation by testing the efficacy of our methods when used on the system on real mosquitos.

With the ultimate goal of mass-producing Sanaria's live malaria vaccine in mind, we believe that our progress with the vision components of the dissection system represents a large step towards a fully/semi-automated system. The vision algorithms are extremely easy to scale with any number of systems and can operate in parallel with correct hardware and software implementations (ie. doing mosquito detection while performing other tasks simultaneously), and ultimately reduce the amount of required human input. Thus, we hope that the outputs of this project will undergo continuous improvement and be implemented in an automated dissection system which results in the mass production of the malaria vaccine, saving millions of lives worldwide as well as a multi-billion dollar burden on the healthcare system.

Management Summary

Team Contributions

Akash Chaurasia	Image processing algorithms (mosquito finder, proboscis finder, neck finder), data collection/labeling, algorithm validation, function documentation, ROS integration, version control & git management
Parth Vora	Image processing algorithms (proboscis finder, neck finder), data collection, validation scripts (mosquito finder, proboscis finder, neck finder), function documentation

Alan Lai	Deep learning algorithms (Orientation classification, mosquito detection with deep learning, pose estimation with deep learning), deep learning documentation, deep learning training framework library

Deliverables

	Minimum	Expected	Maximum
Image Processing	• Mosquito finder algorithm • Neck finder algorithm • Proboscis finder algorithm • Algorithm + design documentation	In addition to minimum... • Library creation • Library documentation • Integration with ROS	In addition to expected... • Validation of CV algorithms • Validation methods + results documentation
Deep Learning	• Orientation algorithm • Framework for training pose estimation and mosquito detection • Documentation of design choices, algorithms, and code usage	In addition to minimum... • Implementation of DL orientation algorithm in a ready-to-be-integrated class-based model	In addition to expected... • Creation and validation of library for training mosquito pose estimation

The deliverables are shown in the table above, with the completed items in green. All planned deliverables were achieved. Note that the initial goals of integrating deep learning models with ROS and validation on the robotic system with real mosquitos were made impossible due to Covid-19, and hence those deliverables were modified to the ones above in March.

Future Work

With regards to image processing, more algorithms can be implemented for dissection pipeline validation. This would enable verification of certain crucial steps (e.g. decapitation), to ensure that they successfully occur, or report error cases otherwise. With regards to deep learning, more work needs to be done on pose estimation, which would involve further annotation of images into Detectron2 compatible format to facilitate further training. Finally, the ultimate goal is to be able to integrate the deep learning models with ROS and validate them.

Lessons Learnt

- Technical skills learnt (deep learning and image processing techniques; knowledge about evaluation metrics; ROS integration)
- Evaluation metrics need to take into account the problem context
- Design specifications must also take into account problem context
- Error handling and error checking are vital for the correct operation of a integrated system

References

1. “Sanaria's Mission.” Sanaria, 2020, sanaria.com/about/.
2. M. Schrum, A. Canezin, S. Chakravarty, M. Laskowski, S. Comert, Y. Sevimli, G. S. Chirikjian, Stephen L. Hoffman, and R. H. Taylor, "An Efficient Production Process for Extracting Salivary Glands from Mosquitoes", arXIV, 2019, http://arxiv.org/abs/1903.02532.
3. Wu, H., Mu, J., Da, T., Xu, M., Taylor, R. H., Iordachita, I., & Chirikjian, G. S. (2019). Multi-mosquito object detection and 2d pose estimation for automation of PfSPZ malaria vaccine production. In 2019 IEEE 15th International Conference on Automation Science and Engineering, CASE 2019 (pp. 411-417). [8842953] (IEEE International Conference on Automation Science and Engineering; Vol. 2019-August). IEEE Computer Society. https://doi.org/10.1109/COASE.2019.8842953

Technical Appendix

Documentation

Documentation for image processing can be found on the following GitLab wiki page.
Documentation for deep learning can be found on the following GitLab wiki page.
Please note that only mentors for our project and the team members have access to these Wiki pages.

Source Code

Source code for image processing can be found here.
Source code for deep learning can be found here.
Please note that only mentors for our project and the team members have access to these Git repositories.