

©Copyright 2019

Sivakumar Balasubramanian

# A Custom Fabricated Low Weight On-Board Vision Sensor for Insect Scale Robot

Sivakumar Balasubramanian

A dissertation  
submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

University of Washington

2019

Reading Committee:

Sawyer Fuller, Chair

Ashis Banerjee

Eric Rombokas

Program Authorized to Offer Degree:  
Mechanical Engineering

University of Washington

## **Abstract**

A Custom Fabricated Low Weight On-Board Vision Sensor for Insect Scale Robot

Sivakumar Balasubramanian

Chair of the Supervisory Committee:  
Professor Sawyer Fuller  
Department of Mechanical Engineering

Controlled flight of insect scale ( 100 mg) Micro Aerial Vehicles (MAVs) has to date required off-board sensors and computation. Achieving autonomy in more general environment will require integrating sensors such as a camera onboard, but this is a challenging task because of the small scale as the component mass and computation must be minimized.

In this work we present the design and fabrication of a low-weight camera 26 mg mounted on a flapping wing insect scale aerial and ground robot. We trained a Convolution Neural Network (CNN) with the images captured by the camera to classify flower and predator images. We show that feedback from the CNNs classification can command the robot to move toward flower and away from predator images. Our results indicate that these computations can be performed using low-weight microcontrollers compatible with the payload and power constraints of insect-scale MAVs. We also perform preliminary optic flow based position estimation experiments with the low weight camera. Many desired capabilities for aerial vehicles, such as landing site selection and obstacle detection and avoidance, are ill-defined. This work shows that Computer Vision (CV) and CNNs, which have previously been deployed only on larger robots, can now be used on insect-scale for such tasks.

## TABLE OF CONTENTS

	Page
List of Figures . . . . .	ii
List of Tables . . . . .	iv
Chapter 1:     Motivation and Previous Work . . . . .	1
Chapter 2:     Fabrication and Interface . . . . .	4
2.1   Camera Fabrication . . . . .	4
2.2   Pinhole Setup . . . . .	6
2.3   Interface . . . . .	7
Chapter 3:     CNN Image classification . . . . .	9
3.1   Training the CNN . . . . .	11
3.2   Computational Constraints . . . . .	11
3.3   Discussions . . . . .	14
3.4   On-Board Image Classification Experiments . . . . .	15
Chapter 4:     Optic Flow . . . . .	17
4.1   Block Matching . . . . .	17
4.2   Distance Formula . . . . .	18
4.3   Experiment . . . . .	20
Chapter 5:     conclusion and Future Work . . . . .	22
Bibliography . . . . .	23

## LIST OF FIGURES

Figure Number		Page
1.1	The insect scale robot (UW RoboFly) with the fully fabricated flight weight camera mounted on it. The Pinhole setup has a focal length of 2 mm and and pinhole diameter of 0.1 mm (top). Close up view of the flight weight camera (bottom). . . .	2
2.1	Diagram of the fully fabricated camera. The vision chip is adhered to an FR4 sheet that has been patterned with isolated pads of gold plated copper. Gold wire bonds connect the vision chips to the pads. A pinhole lens is mounted over the pixel array on the vision chip to focus light rays. . . . .	5
2.3	Pinhole folded from steel shim . . . . .	6
2.4	A black image of a butterfly printed on a white sheet(left), image captured by the flight weight camera with the pinhole setup of 2 mm focal length (middle), image captured by the flight weight camera with the pinhole setup of 4 mm focal length (right). The 2 mm focus pinhole setup has a wider field of view compared to the 4 mm focus pinhole setup . . . . .	7
2.5	The block diagram of the interface of the RoboFly with the flight weight camera mounted on it. The camera captures the images and transfers it to the host PC via a development board. The host PC performs the necessary computation and provides the control commands to a Target PC. The Target PC provides corresponding signals to the piezo actuators for the robot's motion. . . . .	7
3.1	Black images of the three flowers used as the samples for classification task printed on a white sheet (top). The corresponding images of the flowers as captured by the camera with pinhole setup of 4 mm focus, and used for training the CNN (bottom)	10
3.2	Black images of the three predators used as the samples for classification task printed on a white sheet (top). The corresponding images of the predators as captured by the camera with pinhole setup of 4 mm focus, and used for training the CNN (bottom) . . . . .	10

3.3	Plots relating the training accuracy, test accuracy and the classifications per second for various number of learned filters (feature length) for window sizes 3,4 and 5. For all the window sizes, the training and test accuracy increased upto 5 learned filters, after which they reached an accuracy of around 99% and 80% respectively. The classifications per second decreased with increase in number of learned filters.	13
3.4	When the picture of a flower is placed as shown, the trained CNN classified it as a flower and the robot moved towards it. (left) shows the initial position at time $T = 0$ s. (right) shows the final position at time $T = 1$ s. Forward motion was restricted to 1 s to avoid collision with the image sheet.	15
3.5	When the picture of a predator is placed as shown, the trained CNN classified it as a predator and the robot moved away from it. (left) shows the initial position at time $T = 0$ s. (right) shows the final position at time $T = 4$ s.	15
4.1	Diagram of the distance formula used for estimating the position based on optic flow from the images captured by the low weight camera.	18
4.2	Labelled sketch of the test setup for optic flow experiment	19
4.3	Test setup for optic flow experiment	20
4.4	Plot of the distance moved by the camera as captured by the motion capture arena (green) and as calculated from optic flow calculations (red)	21

## LIST OF TABLES

Table Number	Page
3.1 Table with the RAM and Flash Memory requirements for the CNN layers . . . . .	14

## **ACKNOWLEDGMENTS**

First and foremost, I would like to express my gratitude to my professor, Dr. Sawyer Fuller for his guidance for this work and his help in turning this work into a technical paper.

I would like to thank Dr. Ashis Bannerjee and Dr. Eric Rombokas for accepting to be in my thesis committee and providing valuable comments on my work.

I would like to convey my gratitude to my labmates and my graduate advisor Wanwisa Kisalang for their continued support during my graduate studies.

I would also like to thank my friends for their moral support and encouragement during tough times.

Last but not the least, I would like to convey my sincere gratitude to my parents and my brother Dr. Krishnakumar Balasubramanian, for the spiritual guidance, encouragement and support.



## **DEDICATION**

to Micro-Robotics community

## Chapter 1

### **MOTIVATION AND PREVIOUS WORK**

Autonomous flight control of insect scale MAVs has thus far required using external motion capture cameras and computation [2]. This limits the flight to be only within the arena. To make the robot compatible with a diverse set of environments, sensors and computation should be brought on-board. As robot size decreases, the low resolution of Global Positioning System (GPS), which is 1–10 m at best, makes it impractical for flight control. Vision sensors provide a better alternative because they do not have these constraints and are low-weight. They have previously been used successfully in GPS-denied environments on rotating-wing aircraft for navigation and guidance [3–6], autonomous mapping [7], and feature based stereo visual odometry [8]. So far this has not been achieved at an insect scale in full 3-D due to difficulty in down-scaling all the components. Previous work at insect scale demonstrated an integrated camera, but robot motion was constrained to one degree of freedom along guide wires [2].

Here we are interested in achieving full visual flight control at insect scale, which starts with a characterization of our physical robot. The University of Washington (UW) RoboFly is a 75 mg flapping wing robot shown in Fig. 1.1 [9–11]. It is designed and fabricated at the Autonomous Insect Robotics (AIR) Laboratory at the University of Washington, Seattle. It is fabricated using a Diode Pumped Solid State (DPSS) laser to cut an 80 mm thick carbon fiber composite which is then folded into shape. It uses bimorph piezo actuators for flapping its wings at high frequency (140 Hz) for generating the required lift. The RoboFly can perform aerial as well as ground locomotion by flapping its wings, due to lowered center of mass as compared to earlier versions of the insect robots [12].

Because of its scale, angular acceleration rates of the RoboFly are much higher than for larger drones [13]. For example, a 0.5 kg quadrotor-style helicopter, the Ascending Technologies X-3D,

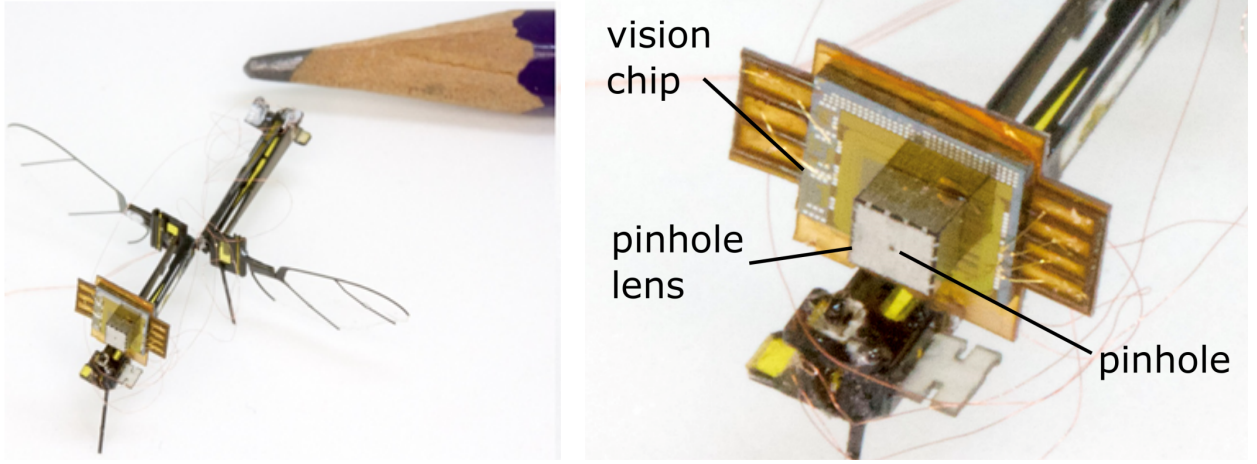


Figure 1.1: The insect scale robot (UW RoboFly) with the fully fabricated flight weight camera mounted on it. The Pinhole setup has a focal length of 2 mm and pinhole diameter of 0.1 mm (top). Close up view of the flight weight camera (bottom).

can perform angular accelerations up to approximately  $200 \text{ rads/s}^2$  for performing multi-flip maneuvers [14], while a much smaller quadrotor, the Crazyflie 2.0 at about 30 g can do so at a higher angular acceleration of  $500 \text{ rads/s}^2$  [15]. By comparison, UW RoboFly can achieve approximately  $1400 \text{ rads/s}^2$  and  $900 \text{ rads/s}^2$  around the roll and pitch axes respectively [12]. A light weight visual sensor compatible with these speeds is needed to perform flight control.

We show the design and fabrication of a low-weight camera with a 2-D vision sensor integrated onto the RoboFly. It has a pixel resolution of  $64 \times 64$  and a weight of 26 mg. It is used to classify images as predators or flowers, and the robot's motion is determined based on the classification feedback. The image classification is performed using a CNN. Neural networks have been shown to match human performance in image recognition [16] and also perform better than other classifiers [17]. Our approach minimizes layers and features of the CNN to reduce computation so that it is compatible with limited on-board computation capability owing to limited battery power and weight constraints of insect scale. We also try preliminary optic flow based position estimation experiments with a test camera similar to the low weight camera's specifications. Ultimately our goal will be to use this camera for flight control. Here we use our robot's ability to move on the

ground as a first step to validate our sensors and algorithms.

In section II, the fabrication and interface of the low weight camera is discussed. Analysis of the CNN classification task performed using the camera is provided in section III. Section IV describes the optic flow experiments performed with the test camera.

## Chapter 2

### FABRICATION AND INTERFACE

The camera consists of a bare die vision sensor, the circuits that interface the data from the vision sensor to a development board, and a pinhole lens. The vision sensor is manufactured by Centeye Inc. (Whiteoak model, Washington, DC), and is designed specifically for the use in insect-scale flight control. It consists of a  $64 \times 64$  pixel array of  $12.5 \mu\text{m}$  sized photo-diodes that capture light intensity values. A 5-wire interface provides power, control/reset commands, and an analog pixel reading. We use a pinhole lens to eliminate the mass of an ordinary lens. Fig. 1.1 shows the RoboFly with the camera mounted on-board and Fig. 1.1 shows a closeup view of the camera.

The RoboFly shown in Fig. 1.1 consists of several structural and functional components such as airframes, transmissions, actuators and wings. In this design, the airframe and transmission are all assembled from a single laminate in order to improve the accuracy and precision of folds. With the help of specially designed folding, this design limits the error to only the rotation about the folding axis. The details about the folding procedure involved are presented in

#### **2.1 Camera Fabrication**

A layout of the camera is shown in Fig. 2.1 and the step by step fabrication process is shown in Fig 2. The vision chip is adhered to a flexible printed circuit board made of copper-clad FR4 plated with gold. The printed circuit was fabricated by first electroplating copper-clad FR4 with gold and then ablating these metals using DPSS laser. We connected the pads on the chip to the substrate using a ball bonder and a  $25 \mu\text{m}$  gold wire. Gold wires are used as they provide good corrosion resistance and high thermal conductivity. The gold plating provides better bondability for gold wires.

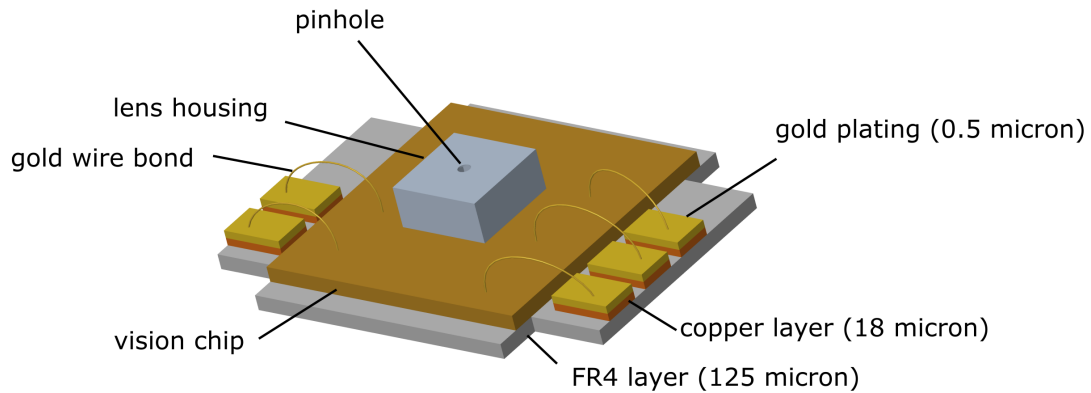


Figure 2.1: Diagram of the fully fabricated camera. The vision chip is adhered to an FR4 sheet that has been patterned with isolated pads of gold plated copper. Gold wire bonds connect the vision chips to the pads. A pinhole lens is mounted over the pixel array on the vision chip to focus light rays.

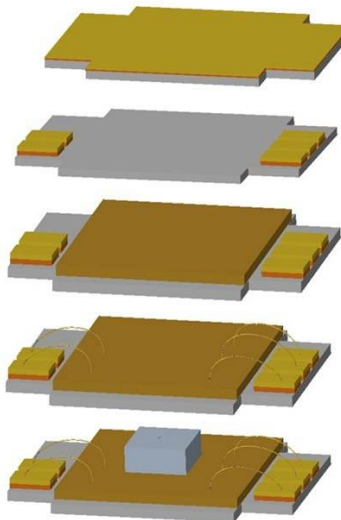


Figure 2.2: steps: i) A copper clad FR4 sheet is plated with a 0.5 micron layer of gold. ii) A DPSS laser is used to ablate the gold plated copper layer into pads. iii) The vision chip is adhered to the substrate. iv) The pads on the chip are connected to the pads on the substrate using wire bonding technique. v) A pinhole lens is adhered onto the pixel array.

## 2.2 Pinhole Setup

The pinhole lens was cut using the DPSS laser from a 50  $\mu\text{m}$  thick stainless steel shim and folded into shape as shown in Fig 2.3. The inner surface of the pinhole lens was painted black to avoid reflections from the steel surface.

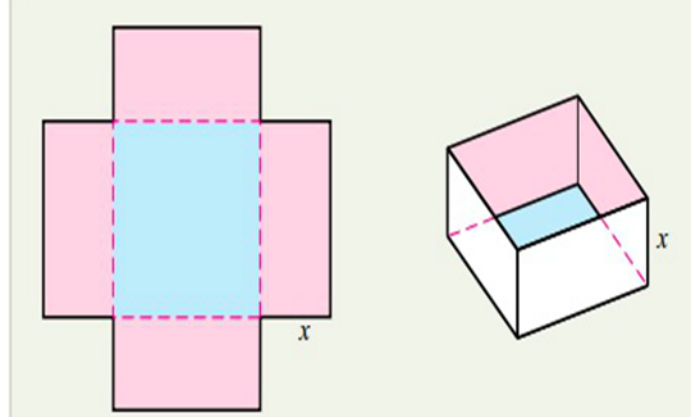


Figure 2.3: Pinhole folded from steel shim

After folding the shim to the desired cubical shape, the edges were covered with black paint to eliminate light entering through the gaps on the edges. The height of the lens determines the focal distance, and we used the following formula for determining the optimal pinhole diameter for a given focal length

$$D = 2\sqrt{\lambda F} \quad (2.1)$$

where,  $D$  is the optimal pinhole diameter,  $F$  is the focal length of pinhole setup, and  $\lambda$  is the wavelength of light (500 nm).

Initially, a pinhole setup of 2 mm focal length was fabricated for which the optimal diameter given by Eq. 2.1 is 0.01 mm. This diameter does not allow enough light to pass through and thus we increased the diameter to 0.1 mm for allowing light at the cost of image sharpness. We performed the experiments with this setup. Next, we fabricated a setup with 4 mm focal length with 0.1 mm pinhole diameter. This has an optimal diameter very close to 0.1 mm and gives better

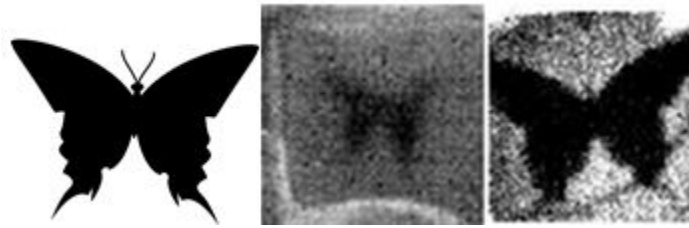


Figure 2.4: A black image of a butterfly printed on a white sheet(left), image captured by the flight weight camera with the pinhole setup of 2 mm focal length (middle), image captured by the flight weight camera with the pinhole setup of 4 mm focal length (right). The 2 mm focus pinhole setup has a wider field of view compared to the 4 mm focus pinhole setup

image sharpness. This setup has narrower field of view compared to the previous setup. Fig. 2.4 shows the images taken with the two pinhole setups.

### 2.3 Interface

Fig. 2.5 shows a block diagram of the closed loop connection of the overall system that controls the RoboFly. Copper wires interface the chip to a development board with ARM Cortex M0+

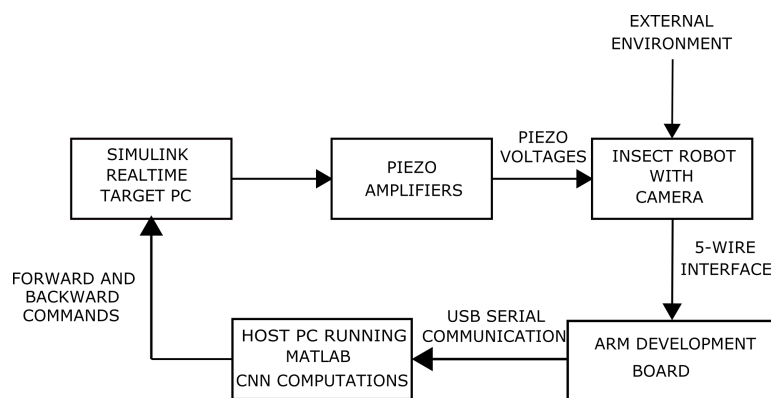


Figure 2.5: The block diagram of the interface of the RoboFly with the flight weight camera mounted on it. The camera captures the images and transfers it to the host PC via a development board. The host PC performs the necessary computation and provides the control commands to a Target PC. The Target PC provides corresponding signals to the piezo actuators for the robot's motion.



micro-controller. The board is programmed to retrieve the pixel values using an analog to digital converter with 16-bit resolution. The values are sent to MATLAB running on a PC (Host PC) using USB serial communication for visual analysis. For situations in which a high frame rate is desired, such as during aggressive flight maneuvers, the chip also allows for only sampling a subset of the pixels, by quickly incrementing the selection register past the other pixels. The analog values are stored as an array in MATLAB, and converted to normalized gray-scale values which can be displayed and processed further using in-built MATLAB functions. High level commands are sent to a second PC running Simulink real-time, which generates signals for the piezo actuators.

## Chapter 3

### CNN IMAGE CLASSIFICATION

To demonstrate the use of the camera to control the insect robot, we implemented an image classification task using a CNN. Using a neural network learning method for classification helps in countering image noise and serves as a test case for tackling ill-defined identification tasks for which they are suited [19] [20]. Pictures of three different flowers and predators were placed in front of the camera. The pictures were captured using both the pinhole setup. We first used the images captured with the pinhole setup of 2 mm focal length and performed the experiments as explained in the experiments section. A shallow CNN with just one convolution layer was used for classifying the images into two classes, either predators or flowers. The layers of the CNN are as follows,

1. A 2-D image input layer that receives a raw captured image as an input to the CNN.
2. A convolution layer of stride length 1.
3. A Rectifier Linear Unit (ReLU) layer as the activation function.
4. A maximum pooling layer with a  $2 \times 2$  window and a stride length of 2.
5. A fully connected layer with outputs equal to number of classes.
6. A softmax layer for normalising the outputs from the fully connected layer.
7. A classification layer that classifies the image as a flower or predator.

Next, the pictures were captured with the setup of 4 mm focal length. The captured pictures are as shown in Fig. 3.1 and Fig. 3.2. The same CNN layers were used for the classification. With this

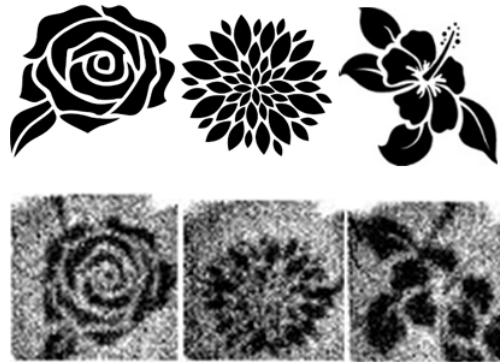


Figure 3.1: Black images of the three flowers used as the samples for classification task printed on a white sheet (top). The corresponding images of the flowers as captured by the camera with pinhole setup of 4 mm focus, and used for training the CNN (bottom)



Figure 3.2: Black images of the three predators used as the samples for classification task printed on a white sheet (top). The corresponding images of the predators as captured by the camera with pinhole setup of 4 mm focus, and used for training the CNN (bottom)

setup, we get higher test classification accuracy than the previous one (95% vs 85%) using fewer learned filters (5 vs 50).

We also used the 4 mm focal length pinhole setup to classify our dataset into 6 classes (3 different flowers and 3 different predators). The subsequent sections give details of the training and test classification accuracy of the CNN for this task.

### ***3.1 Training the CNN***

The CNN was trained with black images printed on a white sheet. A light source was placed in front of the sheet and the reflection of the image was captured by the camera. The gray-scale imagery reduces the number of channels required for the CNN thereby decreasing the computation to a third of what it will take for a RGB image. Each image was captured by varying illuminations and light source angles. The obtained images were also rotated to different angles to overcome rotational variance. A total of 1080 images were generated as training data. The images were taken in the raw form without any filters for noise correction and used for training the CNN. The CNN was trained using back propagation [21] with stochastic gradient descent algorithm with a learning rate of 0.001. All the operations were performed in MATLAB using the Neural Network toolbox. Another set of 360 images of the same flowers and predators were captured and used as test data for testing the accuracy of the CNN. Fig 3.3 shows the training and test classification accuracy for different number of learned filters for window sizes of 3, 4 and 5.

### ***3.2 Computational Constraints***

Implementing neural networks on insect scale robots give us insights into how insect nervous systems might use visual feedback to control flight, a process that is still not fully understood. The main constraints are the computational expense when implementing these tasks with on-board micro-controllers. For future on-board processing, we target an ARM Cortex M4 class processor, which is available at clock speeds up to 120 MHz in an 8 mg Wafer Level Chip Scale Package (WLCSP) with 512 kB flash memory and 176 kB RAM (Atmel SAMG5).

The micro-controller includes Digital Signal Processing (DSP) resources like a 32 bit, 1 cycle Multiply and Accumulate (MAC) unit. The main multiplication and addition tasks for the CNN are the convolution layer and the fully connected layer. Other operations include comparing and storing values in the ReLU and maximum pooling layers. We assume that pre-fetch libraries will be used for reading operations from flash. Thus we assume 1 cycle each for reading operations from flash and RAM. This allows us to estimate the number of cycles required for a single classification with a particular number of learned filters and convolution window size as shown in Eq. 3.1.

$$\begin{aligned}
 L1 &= p^2 \times K^2 \times f + p^2 \\
 L2 &= p^2 \times f \\
 L3 &= (p/2)^2 \times 3 \times f + (p/2)^2 \times f \\
 L4 &= (p/2)^2 \times N \times f + (p/2)^2 \times f \\
 \text{Total cycles} &= L1 + L2 + L3 + L4
 \end{aligned} \tag{3.1}$$

where  $L1$ ,  $L2$ ,  $L3$ ,  $L4$  are the cycles required for convolution, ReLU activation, max pooling and fully connected layers respectively;  $f$  is the number of learned filters in the convolution layer;  $p$  is the number of pixels along the side of the square pixel array;  $K$  is the convolution window size; and  $N$  is the number of classes in the fully connected layer. This is proportional to the number of such classifications made per second as shown in Eq. 3.2.

$$C \propto \frac{(1 \text{ sec}) \times (120 \times 10^6 \text{ MHz})}{\text{Total cycles}} \tag{3.2}$$

where  $C$  is the number of classifications per second. Fig 3.3 shows the relationship between the number of learned filters and the number of classifications per second for window sizes of 3, 4, and 5.

We assume that the training will be performed offline. The main on-board storage requirement is for the weights of the convolution layer and fully connected layer. We assume that we perform

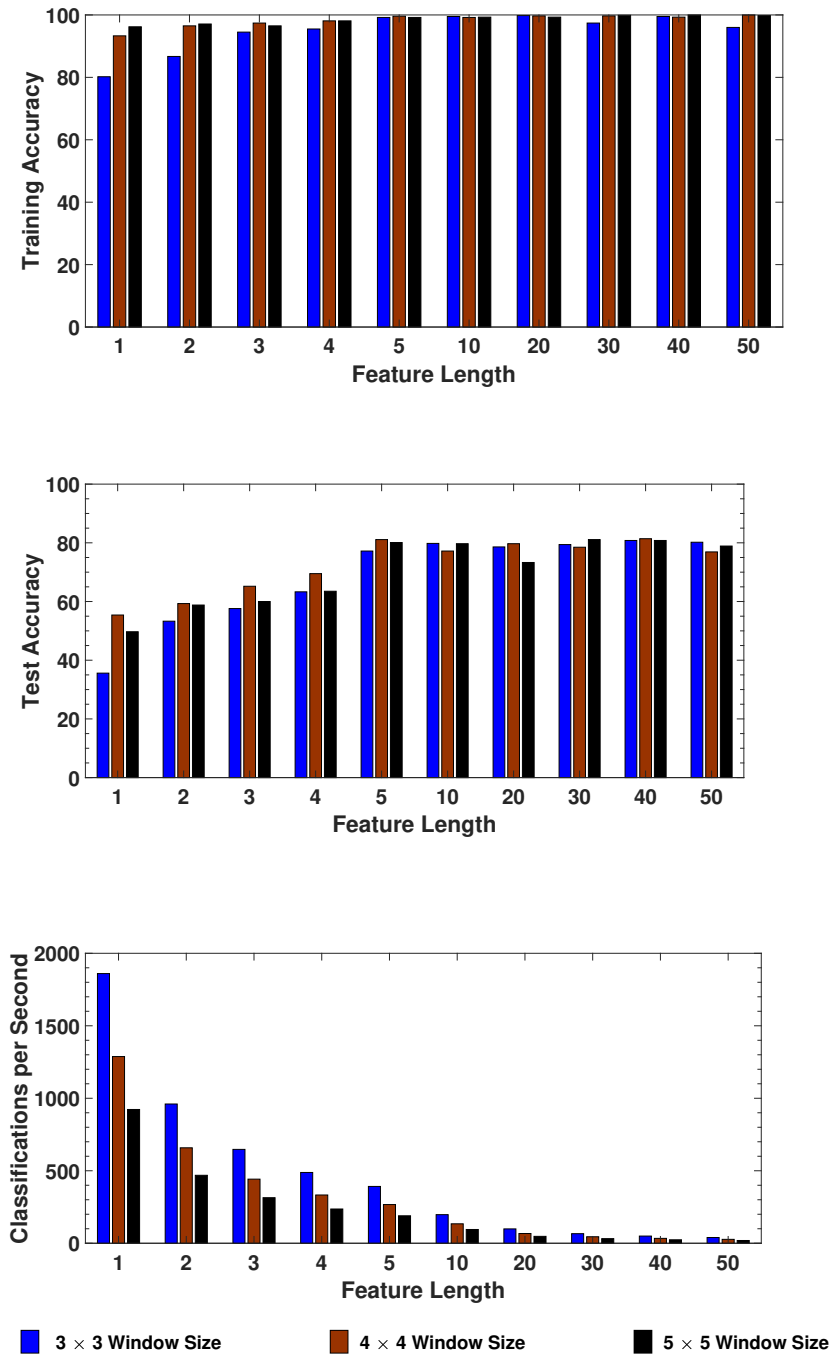


Figure 3.3: Plots relating the training accuracy, test accuracy and the classifications per second for various number of learned filters (feature length) for window sizes 3,4 and 5. For all the window sizes, the training and test accuracy increased upto 5 learned filters, after which they reached an accuracy of around 99% and 80% respectively. The classifications per second decreased with increase in number of learned filters.

Table 3.1: Table with the RAM and Flash Memory requirements for the CNN layers

Layer	Flash (kB)	RAM (kB)
Convolution Layer	2	0
Max Pooling Layer	0	100
Fully Connected Layer	480	0
Total Memory	482	100

a convolution, ReLU activation and max pooling operation simultaneously and store these into the RAM memory. Thus the RAM is utilized mostly by the output of max pooling layer. The other layers do not require significant storage and contribute only towards computation. Table 3.1 shows that flash and RAM memory used by the three layers for a convolution window size of 5 and 40 number of learned filters are compatible with the target micro-controller. More than 40 learned filters do not fit into the RAM memory. The weights are assumed to be stored at 16 bits per weight.

### 3.3 Discussions

From Fig 3.3 we can see that the CNN with number of learned filters of 1 to 4 do not capture all the features very well, which is evident in the training accuracy. There is a gradual increase in the test accuracy. For more than 5 learned filters, the CNN captures the features well and has training accuracy around 99% while the test accuracy reaches 77-80%. Since the amount of training data is very small, the models tend to overfit after a particular number of learned filters. Getting more comprehensive training data would provide better performance, but that is not the emphasis for this work.

The number of the classifications per second goes down as the number of learned filters increases. The latency of image capture increases the time taken for each classification. Thus there is a trade-off between the classification accuracy and the number of classifications that can be made per second. Choosing an optimal classification rate and accuracy is important for high speed tasks

as mentioned in Chapter 1. The present study is concerned primarily with a system-level proof of principle that a CNN could be used on an insect-sized robot, and less concerned with the specific characteristics of our implementation. We therefore leave the analysis of the learned filters for future work.

### 3.4 On-Board Image Classification Experiments

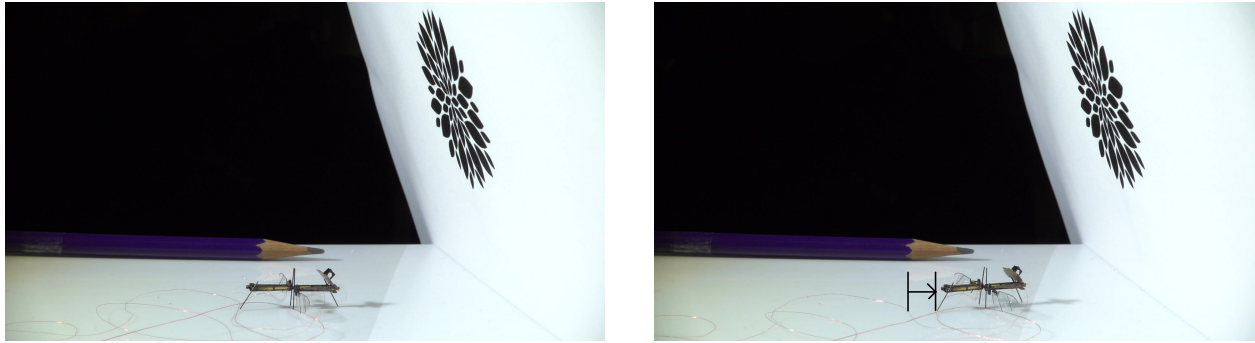


Figure 3.4: When the picture of a flower is placed as shown, the trained CNN classified it as a flower and the robot moved towards it. (left) shows the initial position at time  $T = 0$  s. (right) shows the final position at time  $T = 1$  s. Forward motion was restricted to 1 s to avoid collision with the image sheet.

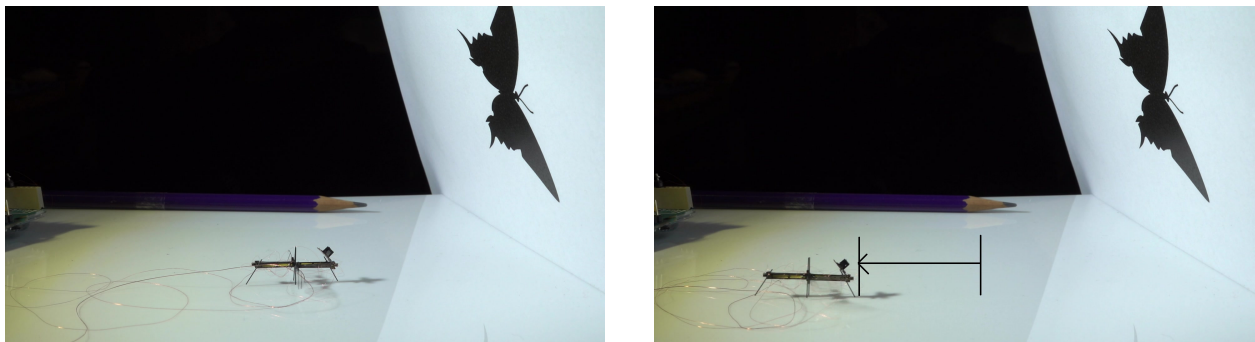


Figure 3.5: When the picture of a predator is placed as shown, the trained CNN classified it as a predator and the robot moved away from it. (left) shows the initial position at time  $T = 0$  s. (right) shows the final position at time  $T = 4$  s.



We also performed a test of the camera mounted on the RoboFly. For these tests, we used an earlier design of our camera with a 2 mm focal length providing lower accuracy. The insect robot was placed in front of the images of the flowers and the predators. The onboard camera captured the images, and the trained CNN classified them.

The robot moved toward the flower images and away from the predator images based on the feedback provided by the CNN in real-time. Fig. 3.4 shows initial and final time instances of the insect robot moving forward toward a flower image and Fig. 3.5 shows initial and final time instances of the insect robot moving backward away from a predator image. Forward motion was restricted to 1 s to prevent the insect robot from colliding with the image sheet.

## Chapter 4

### OPTIC FLOW

Optic flow is the apparent motion of objects relative to an observer. It can be used to determine the current direction of travel of a body from the movement of the objects around that body. Studies such as [27–29], show that optic flow is an important component in insect’s flight.

In computer vision, the shift of an object/pattern in subsequent images is defined as the optic flow in the image sequence. When images are captured from a moving object, the optic flow in the images can be used for determining position and velocity of the object.

Optical flow has been used for flight tasks in aerial robotics. It has been used for altitude control [2], hovering [22–24], and landing [25]. Compared to other techniques such as feature based visual Simultaneous Localization And Mapping (SLAM), optic flow has far lower computational requirements. For example, it was shown that a hovercraft robot with fly like dynamics could visually navigate a 2-D corridor using optic flow in a way that only required 20 KFLOPS [26]. Extensions of this to 3-D should fall within the computational constraints of insect scale.

Several techniques are used to calculate optic flow efficiently such as Lucas Kanade method [30], Horn Schunck method [31]. In this work we use the simple block matching technique for motion estimation and optic flow calculations. The details of the techniques and preliminary position estimation experiment are explained in the following subsections.

#### **4.1 Block Matching**

The underlying principle for block matching algorithm is that, blocks of objects or patterns in one image frame in a image sequence move within the image frame in subsequent images. Thus by finding the closest matching blocks between subsequent images we can estimate the motion of objects in the image sequence. The steps for calculating optic flow of objects/ patterns between two

subsequent images using this algorithm is as follows.

1. Divide the two image into Macro blocks (MB) of required size.
2. Compare light intensity values of each MB in current image to adjacent MBs in the previous image using mean squared error or mean absolute error.
3. For every MB in one image compare for  $s$  pixels around the MB in the other image and find closest matching MB.

Step 3 is the most computationally costly step. This step can be performed by searching for every MB within an area to find the closest match. This is called Exhaustive Search (ES). There are other techniques such as three step search, diamond search and adaptive rood pattern search, that reduce the number of computations in step 3.

#### 4.2 Distance Formula

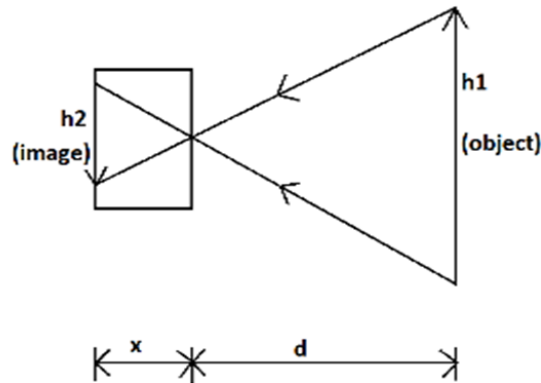


Figure 4.1: Diagram of the distance formula used for estimating the position based on optic flow from the images captured by the low weight camera.

$$h1 = \frac{h2 \times d}{X} \quad (4.1)$$

where  $h_1$  is the object distance and  $h_2$  is the pixel distance moved by the object in subsequent captured image.  $X$  is the focal length of the pinhole camera and  $d$  is the distance between the camera and object.

Fig 4.1 shows the sketch of the distance formula in Eq 4.1 used for calculating the position of insect robot using optic flow values obtained from low weight camera. The Z-axis is fixed to a certain distance ( $d$ ) and the optic flow is calculated only at the 2-D level for the experiments. For

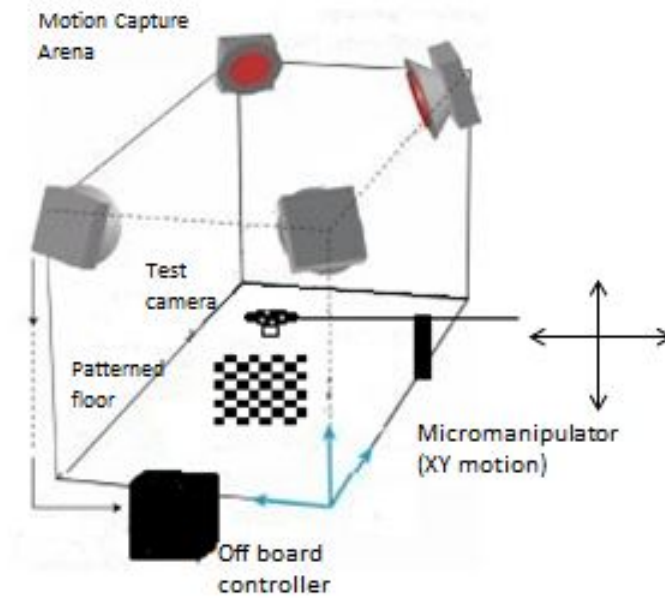


Figure 4.2: Labelled sketch of the test setup for optic flow experiment

the experiments we fix  $d = 220$  mm to mimic the hovering behaviour of insects. We know that  $x$  is the focal length of the pinhole lens which is 4mm. ' $h_2$ ', is the distance moved by pattern or object in the image captured by camera and is equal to the number of pixels moved multiplied by the distance between the centers of two consecutive pixels in the pixel array (pixel pitch). From the specifications of the Centeye vision sensor, we get the pixel pitch to be equal to 0.025 mm. Thus the least motion the camera can capture when placed 220 mm from the pattern or object is  $1 \text{ pixel} \times 0.025 \text{ mm}$  which corresponds to  $h_2 = 0.025 \text{ mm}$ . The distance moved by the pattern relative to

the camera  $h_1 = 1.3$  mm. We restrict the motion of the pattern between subsequent motion to be not more than 8 pixels. This gives a maximum  $h_1$  of 11 mm.

### 4.3 Experiment

We perform 2-D optic flow experiments with a test camera in a motion capture arena. Fig 4.3 shows the experimental setup and Fig 4.2 shows a labelled sketch of the setup. A micro-manipulator is used to hold the test camera setup at a distance of 220 mm from the ground level. It is also used to move the camera for required distances in the X and Y plane. A checkered board pattern is placed at the ground level facing the camera.

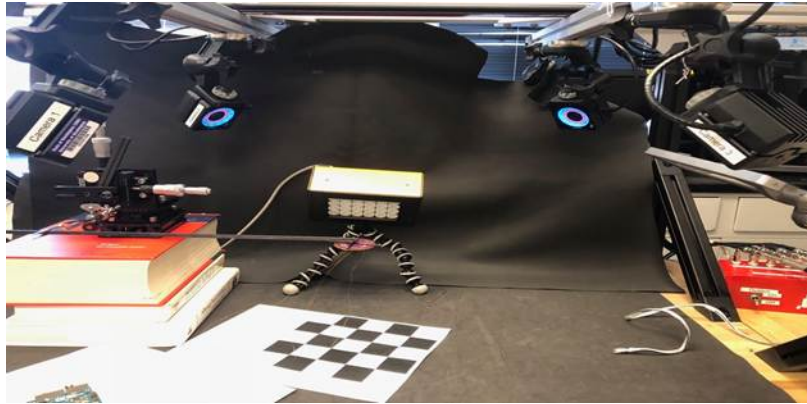


Figure 4.3: Test setup for optic flow experiment

The test camera is moved for specific distances using the micro-manipulator in both X and Y directions. The motion capture camera is used to get the distance moved by the test camera and this is used as the benchmark. The test camera also captures the image of the pattern at each step. The block matching formula is used to find the pixel distance moved. The  $64 \times 64$  images are divided into 4 macroblocks. An ensemble of the ES and ARPS algorithm is used to find the pixel distance moved by each macroblock and an average of these distances are used to give the pixel distance moved between the images as a whole. This is used in the distance formula to get the relative distance moved by test camera between two steps.

Fig 4.4 shows the plot of the distance measured between the steps from both motion capture cameras and from the optic flow calculations . We see that the optic flow calculations are in line with the motion capture camera readings. The inaccuracies arise mainly due to the least count of the optic flow calculations. As mentioned previously, the test camera setup needs the relative motion of the camera and the pattern to be at least 1.3 mm for perceiving the change. This can be corrected with a camera with more resolution.

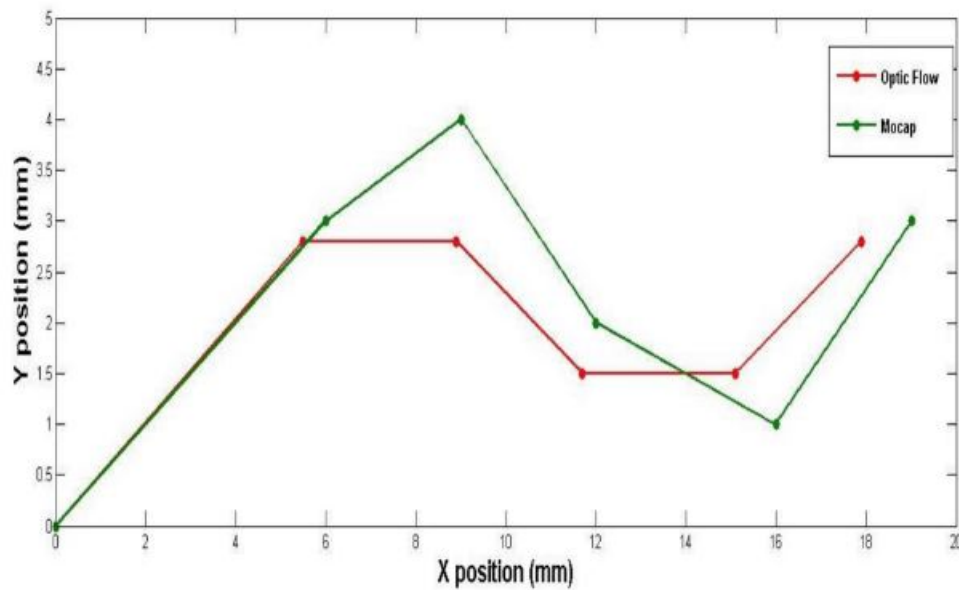


Figure 4.4: Plot of the distance moved by the camera as captured by the motion capture arena (green) and as calculated from optic flow calculations (red)

## Chapter 5

### CONCLUSION AND FUTURE WORK

The work presents the fabrication and interface of a low-weight camera onto an insect scale robot, the RoboFly. Compared to a previous implementation of a vision sensor on an insect scale robot [2], the work here increased the resolution ( $64 \times 64$  vs  $4 \times 32$ ) and reduced the weight (26 mg vs 33 mg). The camera is used for implementing visual control tasks. As a demonstration, image classification using CNN is performed with the images captured by the camera to make the insect robot recognize a flower and a predator image and move toward or away. We also discuss the possibility of implementing the computation on board the insect robot. Our results indicate that current ultra-light embedded micro-controllers are capable of the necessary computation at the necessary speed. The results can be seen as a step towards performing flight control tasks such as landing site detection and obstacle avoidance using only components carried on-board. We believe such tasks, which are hard to explicitly specify, are well-suited to the model-free type of operations performed by neural networks.

The other component of the work is optic flow based position estimation. We show experiments to demonstrate how optic flow of patterns in subsequent images captured by low weight camera can be used to estimate motion of the robot.

Future work will involve implementing a fast control loop for the above techniques. As mentioned in Chapter 1, at insect scale the motion of the robots are swift and it is very important to have a low latency control loop. Thus we have to address the trade off between computational cost and accuracy for any task we perform on the insect robot.

## BIBLIOGRAPHY

- [1] Balasubramanian, S., Chukewad, Y. M., James, J. M., Barrows, G. L., Fuller, S. B. (2018, August). An insect-sized robot that uses a custom-built onboard camera and a neural network to classify and respond to visual input. In 2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob) (pp. 1297-1302). IEEE.
- [2] P.E. Duhamel, N. O. Perez-Arancibia, G. L. Barrows, and R. J. Wood, “Biologically inspired optical-flow sensing for altitude control of flapping-wing microrobots”, *Mechatronics, IEEE/ASME Transactions on*, vol. 18, no. 2, pp. 556568, 2013.
- [3] M. Bloesch, S. Weiss, D. Scaramuzza, and R. Siegwart, “Vision based MAV navigation in unknown and unstructured environments”, in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2010.
- [4] R. Moore, K. Dantu, G. Barrows, and R. Nagpal, “Autonomous MAV guidance with a lightweight omnidirectional vision sensor”, in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.
- [5] S. Ahrens, D. Levine, G. Andrews, and J. P. How, “Vision-based guidance and control of a hovering vehicle in unknown, gps-denied environments”, in *Proc. IEEE International Conference on Robotics and Automation ICRA 09*, 2009, pp. 26432648.
- [6] L. Minh and C. Ha, “Modeling and control of quadrotor mav using vision-based measurement”, in *Strategic Technology (IFOST), 2010 International Forum on. IEEE*, 2010, pp. 7075.
- [7] F. Fraundorfer, L. Heng, D. Honegger, G. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, “Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV”, in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, oct 2012.
- [8] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, “Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments”, in *Proc. SPIE Unmanned Systems Technology XI*, 2009.
- [9] A. T. Singh, Y. M. Chukewad, and S. B. Fuller, “A robot fly design with a low center of gravity folded from a single laminate sheet”, in *workshop on Folding in Robotics, IEEE conference on Intelligent Robots and Systems* (2017).



- [10] J. James, V. Iyer, Y. Chukewad, S. Gollakota, S. B. Fuller, “Liftoff of a 190 mg Laser-Powered Aerial Vehicle: The Lightest Untethered Robot to Fly, in *Robotics and Automation (ICRA)*, 2018 *IEEE Int. Conf. IEEE*, 2018.
- [11] Y. M. Chukewad, A. T. Singh, and S. B. Fuller, “A New Robot Fly Design That is Easy to Fabricate and Capable of Flight and Ground Locomotion, in *Intelligent Robots and Systems (IROS)*, 2018 *IEEE/RSJ International Conference on. IEEE*, 2018. (accepted)
- [12] K. Ma, S. Felton, and R. Wood, “Design, fabrication, and modeling of the split actuator microrobotic bee”, in *Intelligent Robots and Systems(IROS), IEEE/RSJ International Conference on. IEEE*, 2012.
- [13] V. Kumar and N. Michael, “Opportunities and challenges with autonomous micro aerial vehicles”, *Int. J. Robot. Res. (IJRR)*, vol. 31,no. 11, pp. 12791291, 2012.
- [14] S. Lupashin, A. Schollig, M. Sherback, and R. D. Andrea, “A simple learning strategy for high-speed quadcopter multi-flips”, in *Proc. of the IEEE Int. Conf. on Robotics and Automation, Anchorage, AK*, May 2010, pp. 16421648.
- [15] G. P. Subramanian, “Nonlinear control strategies for quadrotors and CubeSats”, *Masters thesis, University of Illinois at Urbana-Champaign*, 2015
- [16] D. C. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column Deep Neural Networks for Image Classification”, in *Proceedings of Proceedings of Computer Vision and Pattern Recognition*, 2012, pp. 3642-3649.
- [17] H. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection”, in *IEEE Patt. Anal. Mach. Intell.*, volume 20, pages 2238, 1998.
- [18] J. W. Strutt, “On Pin-hole Photography”, *Phil. Mag.*, v.31, pp 87-99, 1891.
- [19] M. E.-Petersen, D. de Ridder, and H. Handels, “Image processing with neural networks a review”, *Pattern Recognition*, vol. 35, pp.22792301, 2002.
- [20] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks”, in *NIPS*, 2012.
- [21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation”, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart (AAAI-91), July 1991, pp. 762-767. and James L. McClelland, Eds., vol. 1, ch. 8, pp. 318-362. Cambridge, MA: MIT Press, 1986.

- [22] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications", in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2013.
- [23] V. Grabe, H. H. Bulthoff, D. Scaramuzza, and P. R. Giordano, "Nonlinear ego-motion estimation from optical flow for online control of a quadrotor UAV", *The International Journal of Robotics Research*, vol. 34, no. 8, pp. 1114-1135, 2015.
- [24] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "MAV navigation through indoor corridors using optical flow", in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010.
- [25] B. Herisse, F.-X. Russotto, T. Hamel, and R. Mahony, "Hovering flight and vertical landing control of a vtol unmanned aerial vehicle using optical flow", in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pp. 801-806, 2008.
- [26] S. B. Fuller and R. M. Murray, "A hovercraft robot that uses insect inspired visual autocorrelation for motion control in a corridor", in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Karon Beach, Phuket, pp. 1474-1481, 2011.
- [27] Baird, Emily, et al. "Nocturnal insects use optic flow for flight control." *Biology letters* 7.4 (2011): 499-501.
- [28] Collett, Thomas S. "Insect vision: controlling actions through optic flow." *Current Biology* 12.18 (2002): R615-R617.
- [29] Egelhaaf, Martin, and Roland Kern. "Vision in flying insects." *Current opinion in neurobiology* 12.6 (2002): 699-706.
- [30] Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." (1981): 674-679.
- [31] Horn, Berthold KP, and Brian G. Schunck. "Determining optical flow." *Artificial intelligence* 17.1-3 (1981): 185-203.