Automated Labeling for Robotic Autonomous Navigation Through Multi-Sensory Semi-Supervised Learning on Big Data

Junhong Xu, Shangyue Zhu, Hanqing Guo and Shaoen Wu

***Abstract*—Imitation learning holds the promise to address challenging robotic tasks such as autonomous navigation. It however requires a human supervisor to oversee the training process and send correct control commands to robots without feedback, which is always prone to error and expensive. To minimize human involvement and avoid manual labeling of data in the robotic autonomous navigation with imitation learning, this paper proposes a novel semi-supervised imitation learning solution based on a multi-sensory design. This solution includes a suboptimal *sensor policy* based on sensor fusion to automatically label states encountered by a robot to avoid human supervision during training. In addition, a *recording policy* is developed to throttle the adversarial affect of learning too much from the suboptimal sensor policy. As a result, this solution allows the robot to learn a navigation policy in a self-supervised manner without human intervention after the initial data collection. With extensive experiments in indoor environments, this solution can achieve near human performance in most of the tasks and even surpasses human performance in case of unexpected events such as hardware failures or human operation errors. To best of our knowledge, this is the first work that synthesizes sensor fusion and imitation learning to enable robotic autonomous navigation in the real world without human supervision.**

I. Introduction

Indoor mobile robot navigation has a long research history in robotics. Many solutions have been proposed for indoor robotic navigation. One of the most widely adopted approaches is SLAM, which is a two-stage approach consisting of perception and action stage [27]. A global map is built using onboard sensors in the action stage. This map is then given to a motion planner to make predictions [18], [26]. This approach is environment-dependent and requires tremendous efforts to establish the environment maps. It also needs intensive computation to support the map based prediction.

Deep reinforcement learning (DRL) [15] is another attempted exploration to address autonomous indoor navigation. Although it has yielded many successful outcomes including games [12]–[14], [22] and robotic grasping [11] , it requires robots to learn from trial-and-error that is impractical and too expensive in real world mobile robotic tasks. Therefore, many works only employ DRL for navigation tasks in simulated settings [6] to avoid hardware damaging. The learned policy is then transferred to real world scenarios [21], [25]. However, the discrepancy between simulated environments and the real world is large and still requires fine-tuning. Meanwhile it is very likely to cause damage to robots.

Recently, end-to-end imitation learning methods have been employed to resolve complex robotic tasks such as manipulating objects [17], navigating to a target position [16], and self-driving vehicles [2]. Imitation learning converts sequential decision tasks to supervised learning problems, where a policy is trained to minimize the errors between the predicted actions and the actions taken by an expert policy. However, there are two major challenges when applying this method to sequential decision tasks: (1) data mismatch between states encountered by a trained policy and an expert policy, and (2) difficulties of querying an expert policy in real-world scenarios. The first problem is normally tackled by literature solutions with a DAgger algorithm [19] that iteratively collects training examples using both an expert and a trained policy. This approach however requires a human supervisor to provide correct control commands given an observation encountered by a trained policy, which is expensive and inaccurate [20]. Some works have attempted to address the above imitation learning problems by using a hierarchy of supervisors [8] or reducing the number of queries to the expert policy [9], [31], while other works focus on platforms and security [5], [30].

In this work, we focus on autonomous robotic navigation, we aim at the challenges of using imitation learning in real-world robotic domains where querying a human supervisor is expensive and inaccurate. We propose a solution, ***M**ulti-**S**ensory **S**emi-**S**upervised **L**earning (MS3L)*, which is based on imitation learning augmented with sensor fusion. Our solution has two key innovations:

- One is to employ various types of sensors, including both imaging and non-imaging, in a deep learning framework to minimize human involvement in that it only requires ONE iteration of human supervision to initialize a navigation policy.
- The other is that, after initializing the navigation policy, *MS3L* uses a suboptimal *sensor policy* to label the observations encountered by a mobile robot at its own, which completely eliminates the need of querying an expert policy in literature solutions. To reduce the adversarial effect of learning from the suboptimal policies, we design a *recording policy* based on the safety policy [31], which controls the degree of information learned by the navigation policy.

In the rest of this paper, Section II reviews the related work, particularly imitation learning and reinforcement learning. Then, Section III discusses the design of our proposed solution. Next, Section IV presents the extensive performance evaluations of *MS3L* in real indoor environments. The paper is finally concluded by Section V.

II. Related Work

There are a large number of works that address mobile robotic navigation with various methodologies including

Authors are with the Department of Computer Science, Ball State University, Munice, Indiana. {jxu7,szhu,hguo,swu}@bsu.edu

robotic controls, machine learning, and reinforcement learning. Though, we present the most related works and formulate the research problems in this section.

A. Imitation Learning

Imitation learning solutions in literature are unanimously based on a pioneer DAgger algorithm proposed by Ross *et al*. [19] to iteratively train a policy that imitates a certain expert policy. DAgger works as follows. The human expert operates the robot to collect data and this data is used to initialize the robot's policy. The human expert then deploys the trained robot in the environment then run and collect by itself. The human expert corrects the data collected by the robot offline and aggregate the corrected data with the previous one. The robot is trained and deployed iteratively until a satisfying performance is reached. This algorithm has been widely used in many robotic problems [4], [8], [20]. These methods require some expert policies to be queried, which is impractical and too expensive in real environments especially when human supervision is required. Ross *et al*. use DAgger to train a drone to fly in forests and avoid collisions [20]. The human supervisors are provided with partial feedback to correct actions of trained policy offline. While this solution aims at reducing supervisor's burden, it still does not solve the problem of querying a human operator. Laskey *et al* proposes an approach to use a hierarchy of supervisors to learn grasping policy [8], which actually requires more on the burden of human supervisor. There are many works extending DAgger algorithm and focusing on the improvement of query efficiency to an expert policy [9], [31]. These works are the most relevant to ours in a way that they constrain training data with some query metrics. These methods however assume that an oracle is always available and easy to be queried, which is often unlikely in real environments. In addition, the issue of noisy sensor measurements is not addressed in these works, which definitely degrade the performance of a trained policy.

Another approach to robotic navigation using imitation learning is to set up multiple cameras to capture training samples in different directions [2], [3]. Multiple cameras are installed on a drone or car to collect training samples. Each sample is labeled according to camera positions. This method tackles the data mismatch problem by training the policy with samples from different view directions. However, this data collection strategy only works properly in the domain of lane following. In addition, installing multiple cameras on small robots is challenging or impractical.

B. Reinforcement Learning

Recently, deep reinforcement learning (DRL) [15] has been attracted a lot attention in the robotic control field. Many works based on DRL have been performed to address robotic navigation tasks. In [6], the authors train a DQN (deep q learning) agent [15] to cross an intersection in simulation. Another work proposes a simulated environment to train a DRL agent to reach a target position in indoor environments [32]. Lillicrap et al. [12] proposes deep deterministic policy gradient (DDPG) to train an agent to avoid dynamic obstacles in simulation. These methods however are all constrained in simulated settings where damage to the agents is not a concern. Applying DRL to real environments is still challenging. Many researchers attempt to address this problem by transferring learned DRL policies from simulation to real world navigation tasks [21], [28]. The difference between simulated environments and the real world settings makes this adaption difficult.

C. Learn from Noisy Labels

Our work is also closely related to the idea of learning from noisy labels in deep neural networks, where a network is trained using a dataset consisting of inaccurate labels. Most works address this problem by modeling the label noise distribution with either a neural network layer or probabilistic graphical models [24], [29]. Our work constrains learning from noisy labels by restricting to learn from sub-optimal sensor policy using a recording policy.

III. Multi-Sensory Semi-Supervised Autonomous Mobile Robotic Navigation

We consider a mobile robot navigating in indoor environments of pedestrians and obstacles. The robot's goal is to navigate rapidly and safely in indoor environments. Our goal is to minimize human supervision and allow the robot to learn from its own experience. We do not assume the robot has access to a multi-stage motion planer as a reference policy as in [16].

We propose and implement a system, ***M**ulti-**S**ensory **S**emi-**S**upervised **L**earning (MS3L)*, to enable the autonomous robotic navigation. Our system combines four policies π_h, π_s, π_{θ_r}, and π_{θ_n}, which respectively represent human policy, sensor policy, recording policy, and navigation policy. By initializing π_{θ_n} with π_h and constraining learning from a suboptimal policy π_s with a recording policy π_{θ_r}, the robot is able to surpass the suboptimal policy and achieve near human performance in a self-supervised manner.

In this section, we present the detail design and framework of *MS3L*, including the policies and the training protocol.

A. MS3L Policies

The *MS3L* solution has four policies including *human policy (π_h), sensor policy (π_s), navigation policy (π_{θ_n})* and *recording policy (π_{θ_r})*, designed for the multi-sensory imitation learning. Among those, *human policy* and *sensor policy* are also referred as *reference* policies (π_{ref}) to guide the navigation.

1) Human Policy: Human policy π_h refers to a human operation that guides the robot to navigate through environments and avoid collision. The operator provides initial demonstrations using a joystick to tele-operate the robot remotely. Each human demonstration refers to the data collection in one human operation of the robot. is Note that the human policy is used only ONCE in data collection for each of the navigation and recording policy.

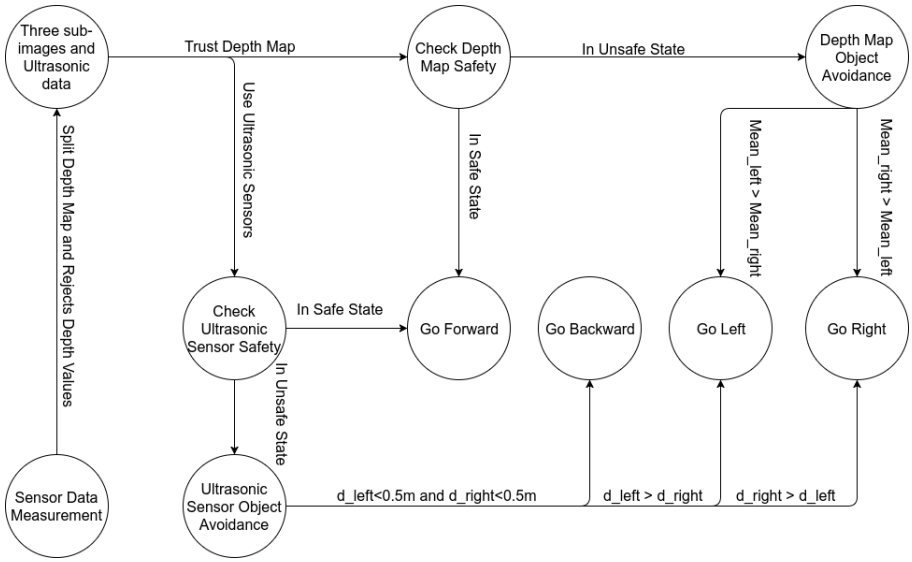


Fig. 1: Finite state machine for sensor policy.

2) Sensor Policy: Sensor policy π_s is used in the self-supervised learning phase to generate suboptimal action labels. This policy generates a set of basic robot control commands: turning, acceleration, and deceleration. Unlike in simulated or controlled environments where dynamics are always known, it is not possible to obtain an accurate model of an unstructured real indoor environment. Our goal is not to use an expensive accurate range sensor like a laser device used in [16] to create an optimal depth control policy, but to build a suboptimal sensor policy π_s upon the coarse measurements of those unreliable cheap commodity sensors.

Sensor policy π_s is high bias and low variance due to the hand engineered control and noisy measurements in sensor data. It is regulated through a finite state machine as shown in **Fig. 1**. It takes the current depth map M and ultrasonic sensor measurements d_{left} and d_{right} as inputs and generates a control command $a = \pi_s(M, d_{left}, d_{right})$. At the first stage, it splits the depth map M into three sub-image. Then, it rejects depth values in each sub-image which has the value larger than two standard deviation of the entire depth image This checks whether we can trust the current depth estimation. If the number of rejected data is large, the robot distrusts the depth map, and rather uses the two ultrasonic sensor measurements (d_1, d_2) to calculate robot control actions. This usually happens when the robot is facing towards a plain wall or objects only appear in one camera. If the depth image is used, mean depth value of the middle sub-image is used, e.g. the value is smaller than 0.8m in our experiments, to determine whether the robot needs to take object avoidance actions. If the ultrasonic sensor is used, d_{left} and d_{right} are used to check whether the robot is in the safety position. Go back action is only taken when ultrasonic sensor is used because ZED stereo camera is not valid in detecting distance within 0.5 meters.

It should be noted that, π_s is not robust and we expect the recording policy described later to constrain the robot to only learn a subset of data generated by π_s

3) Navigation Policy: Navigation policy π_{θ_n} outputs an action to avoid obstacles based only on the current RGB image observed from ZED stereo camera. To handle high dimensional observations i.e. images in our case, we parameterize θ_n as a 5-layer convolutional neural network (CNN) as shown in Fig.2. We adopt VGG-like architecture [23] in our design. All convolution layers have a 3×3 kernel size. The first two convolution layers have 64 channels followed by a max pooling layer. The last two convolution layers have 128 channels followed by a fully-connected layer with 256 neurons. ReLU activation functions are applied to all layers. We normalize iRobot Create2 linear and angular velocities between $[-1, 1]$ when collecting dataset. To bound with the this normalized scale, we add a $tanh$ activation function before the output. The policy outputs 2-dimensional actions in forms of linear velocity and angular velocity. Input images are resized to 128×128 for real-time inference.

We define the navigation policy dataset as $D_{\pi_{\theta_n}} = \{(x_1, \pi_{ref}(x_1)), ..., (x_i, \pi_{ref}(x_i)), ..., (x_t, \pi_{ref}(x_t))\}$, where x_i represents the observation at timestep t, and $\pi_{ref}(x_i)$ refers to the output of reference policies consisting of human policy π_h that is used only in the first iteration and sensor policy π_s that is used in the rest of training iterations. The imitation objective is defined as

$$C_i(D_{\pi_{\theta_n}}, \pi_{\theta_n}) = 1/N \times \sum_{i=1}^{i=N} ||\pi_{\theta_n}(x_i) - \pi_{ref}(x_i)||_2^2, \quad (1)$$

where N is the mini batch size for training. With this objective function, the navigation policy is trained to imitate from both human and sensor policies: π_h and π_s.

4) Recording Policy: Recording policy π_{θ_r} is crucial to our learning framework, where θ_r is parameterized by a 2-layer fully-connected neural network. It takes a feature vector from the last fully-connected layer $fc5 = \pi_{\theta_n}^5(x)$ of the navigation policy as input and generates the probability $p_r = \pi_{\theta_r}(fc5)$ that the current observation is needed for the navigation policy to learn, as shown in Fig.2, where x is observation and θ_n^5 represents the parameters of last fully-connected layer of the navigation policy. This design choice of using shared convolutional layers for navigation and recording policies is based on two factors. First, the limited GPU memory on Jetson TX1 is not powerful enough for two CNNs: one for the

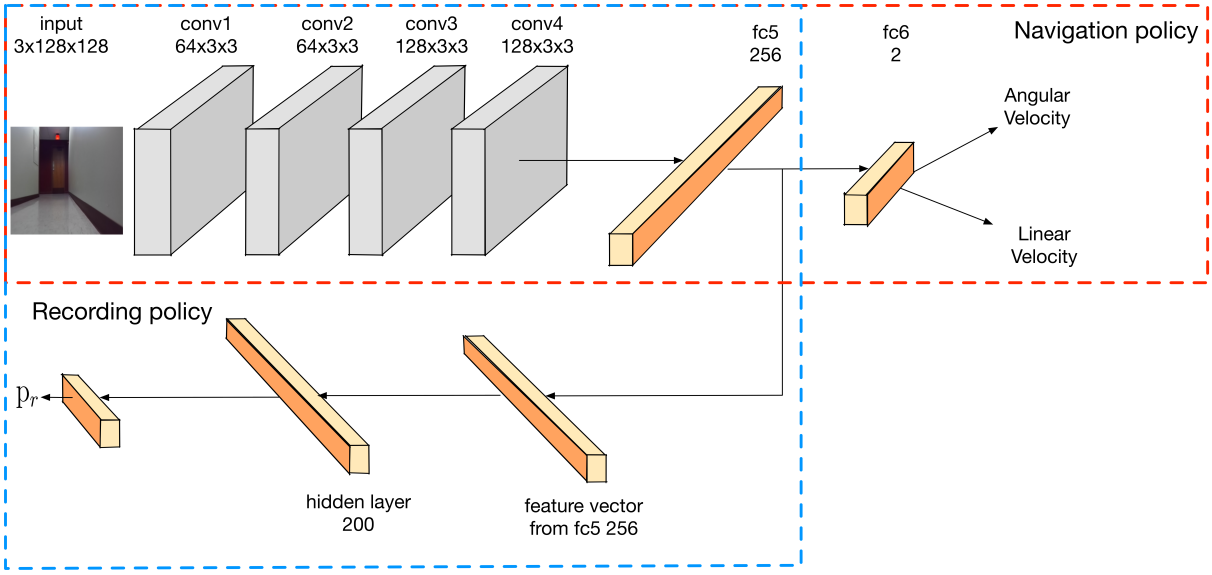


Fig. 2: Our model consists of two policies: navigation policy π_{θ_n} and recording policies π_{θ_r}

navigation policy and the other for recording policy. Second, the feature extracted by the CNN at layer fc_5 can be shared and reused by both the navigation policy and the recording policy to improve its learning speed [10].

We use the sensor policy π_s as a suboptimal policy to self label data after the human operated pre-training stage. In order to throttle the propagation of learning errors from π_s, the recording policy only keeps those labelled data from π_s if π_{θ_n} deviates far from both reference policies. We define the deviation as the squared difference between the output of the navigation policy π_{θ_n} and that of reference policies (π_h and π_s) as

$$\begin{aligned} e(x, \pi_{\theta_n}, \pi_h, \pi_s) = \gamma ||\pi_{\theta_n}(x) - \pi_h(x)||_2^2 \\ +(1-\gamma)||\pi_{\theta_n}(x) - \pi_s(x)||_2^2, \end{aligned} \tag{2}$$

where γ is a constant between 0 and 1 that weights the relative importance of the two reference policies, e.g. if γ is closer to 1, the sensor policy is less accounted for the deviation. With this deviation metric, a binary function indicating whether to record is defined as

$$\epsilon(x, \pi_{\theta_n}) = \begin{cases} 1, & \text{if } e(x, \pi_{\theta_n}, \pi_h, \pi_s) > \tau \\ 0, & \text{otherwise} \end{cases}, \tag{3}$$

where τ is a scalar indicating the degree of error tolerated by the recording policy.

π_{θ_r} is trained on a recording dataset $D_{\pi_{\theta_r}} = \{\pi^5_{\theta_n}(x_1), ..., \pi^5_{\theta_n}(x_n)\}$, which is collected using extra human demonstrations, from the last fully-connected layer $fc5$. The objective function to be minimized by π_{θ_r} is a binary cross-entropy loss function defined as

$$C_r(D_{\pi_{\theta_r}}, \epsilon, \pi_{\theta_r}\pi_{\theta_n}, \pi_{ref}) = -\frac{1}{N} \times$$
$$\sum_{n=1}^{N} \epsilon(x_n, \pi_{\theta_n}) log(\pi_{\theta_r}(x_n)) + (1 - \epsilon(x_n, \pi_{\theta_n})) log(1 - \pi_{\theta_r}(x_n)),$$

where π_{ref} is the reference policies consisting of π_h and π_s.

After the pre-training stage, observations are recorded only if $\pi_{\theta_r}(x) > \beta$, where β is a recording threshold controlling the degree to which the robot trusts π_{θ_r}. We refer the observations recorded by π_{θ_r} as hard samples. Some examples with different thresholds are shown in Fig.3. Fig.3a shows the images recorded in dangerous states i.e. turning or being close to walls. In contrast, Fig. 3b shows the images of going forward, which are not necessary for the navigation policy to learn. A larger β places more constraints to the number of observations recorded and labeled by π_s.

(a) $\beta = 0.99$ **(b)** $\beta = 0.5$

Fig. 3: Images are uniformly sampled from datasets that have different β values.

B. Training Protocol

With these policies, the training procedure of *MS3L* is performed as: the navigation policy π_{θ_n} is initialized by human policy π_h and iteratively trained by a suboptimal internal policy π_s constrained on a recording policy π_{θ_r}. Our training procedure consists of two stages with five iterations in total, which is shown in Algorithm 1. The first stage is a pre-training stage, where a human operator controls the robot using a PlayStation wireless controller to navigate through the environments and collect the initial recording and navigation policies datasets: $D_{\pi_{\theta_r}}$ and $D^0_{\pi_{\theta_n}}$ to initialize the navigation policy: π_{θ_r} and π_{θ_n}. These two datasets are collected in opposite directions shown in Fig.6. to ensure that the recording policy correctly classifies hard observations. While the navigation policy is trained to have a basic understanding of the environment, it does not learn any complex actions such as turning or decelerating. A self-supervised learning

Algorithm 1 Multi-Sensory Self-Supervised Learning

procedure *MS3L* TRAINING PROTOCOL

Randomly initialize π_{θ_r} and π_{θ_n}.

Pre-training

Collect $D_{\pi_{\theta_r}}$ and $D^0_{\pi_{\theta_n}}$ using π_h

$\pi_{\theta_n^0} = arg\,min_{\theta_n} C_i(D^0_{\pi_{\theta_n}}, \pi_{\theta_n})$

$\pi_{\theta_r^0} = arg\,min_{\theta_r} C_r(D_{\pi_{\theta_r}} \cup D^0_{\pi_{\theta_n}}, \epsilon, \pi_{\theta_r}, \pi_{\theta_n^0}, \pi_{ref})$

Self-supervised learning

for $i = 1$ **to** k **do**

Execute $\pi_{\theta_n^{i-1}}$, collect and label observations into D using π_s and $\pi_{\theta_r^{i-1}}$.

Only keep $(x, \pi_{ref}(x_i))$ pair from D if $\pi_{\theta_r^i}(\pi^5_{\theta_n^{i-1}}(x)) > \beta$

$D^i_{\pi_{\theta_n}} = D \cup D^{i-1}_{\pi_{\theta_n}}$

$\pi_{\theta_n^i} = arg\,min_{\theta_n} C_i(D^{i-1}_{\pi_{\theta_n}}, \pi_{\theta_n^{i-1}})$

$\pi_{\theta_r^i} = arg\,min_{\theta_r}$
$C_r(D_{\pi_{\theta_r}} \cup D^i_{\pi_{\theta_p}}, \epsilon, \pi_{\theta_r^{i-1}}, \pi_{\theta_n^{i-1}}, \pi_{ref})$

return $\pi_{\theta_n^4}$ and $\pi_{\theta_r^4}$

stage is proposed to further improve the policy without human operators. At this stage, π_s is used to generate labels for the collected data. In addition, π_{θ_r} is used to constrain the data collection to remove largely deviated data. Specifically, an observation x and its corresponding label $\pi_s(M, d_1, d_2)$ are only recorded when $\pi_{\theta_r}(\pi^5_{\theta_n}(x)) > \beta$. This ensures that the deficient labeling resulted from noisy measurements of π_s is minimized. Then, the navigation policy is trained upon the aggregation of the initial dataset and the selected observations. The trained navigation policy and the aggregation of recording and navigation datasets are used to update the recording policy.

IV. PERFORMANCE EVALUATION

A. Robotic Platform and Implementation Details

The robot is built based on commodity supplies as shown in Fig. 4. The base of the robot is an iRobot Create2[1], which has a linear velocity of [-0.5m/s, 0.5m/s] and an angular velocity of [-4.5rad/s, 4.5rad/s]. The robot is equipped with two ultrasonic sensors for distance measurements and a ZED™ Stereo Camera[2] to capture RGB images and as well as depth images. The ultrasonic sensors can detect objects within a distance of [5cm, 400cm] (or [2in, 156in]). The valid depth estimation of the stereo camera falls into [0.5m, 20m]. These two types of depth/distance sensors, namely ultrasonic and stereo camera, complement with each other to derive our sensor policy. In addition, a NVIDIA® Jetson TX1[3] is used as the robot's brain to make inferences, which has 256 CUDA cores with 2GB memory to support a moderate-sized neural network in real-time computation.

[1]http://www.irobot.com/About-iRobot/STEM/Create-2.aspx

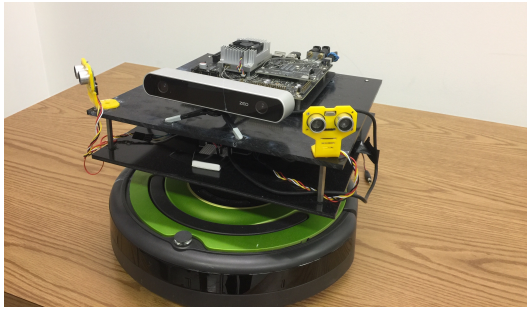


Fig. 4: iRobot Create2 equipped with a stereo camera, two ultrasonic sensors, and a Jetson TX1.

The experiments have been performed in the 3rd floor of the Robert Bell (RB) Hall building at Ball State University. Fig. 5a and 5b are indoor environments where our robot has been trained and Fig.6 shows the floor plan.

(a) Hallway

(b) Classroom

Fig. 5: Training environments for robots.

We have implemented *MS3L* on tensorflow [1]. Adam optimizer [7] is used and configured with a learning rate of 0.0001 for π_{θ_n} and 0.001 for π_{θ_r}. Both networks are trained over 50 epochs with a $L2$ weight decay of 0.0001. We set $\gamma = 0.8$ and $\tau = 0.00025$ while training π_{θ_r}. In addition, we set $\beta = 0.99$ in data collection using the recording policy. Input images are rescaled to 128×128 RGB images. We set $k = 4$ during self-supervised training. For each training iteration, we run the robot for $250s$ and the images are taken at $30fps$. We use the traveled *distance* and *time* to collision as evaluation metrics in the experiments. We set the maximum travel duration to be $250s$ throughout the experiments.

We have trained our robot in two indoor environments: RB 3rd floor hallway as shown as red arrow in Fig. 6 and in RB-356 classroom during break time. The classroom is an extremely difficult environment, where the robot collides into obstacles even with human operator's supervision in some tests.

There are two types of performance evaluations in the experiments. During the training phase, the performance is measured as mean sqaured error (MSE) for navigation policy and cross-entropy loss for recording policy. In addition, the distance traveled is also recorded. After training, the performance is measured as 1) travel time and 2) travel distance. The comparison with DAgger method is also shown at the last of this section.

B. Training Evaluation

1) Validation: We first evaluate the performance of our navigation and recording policies during training process. We

[2]https://www.stereolabs.com/

[3]http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html

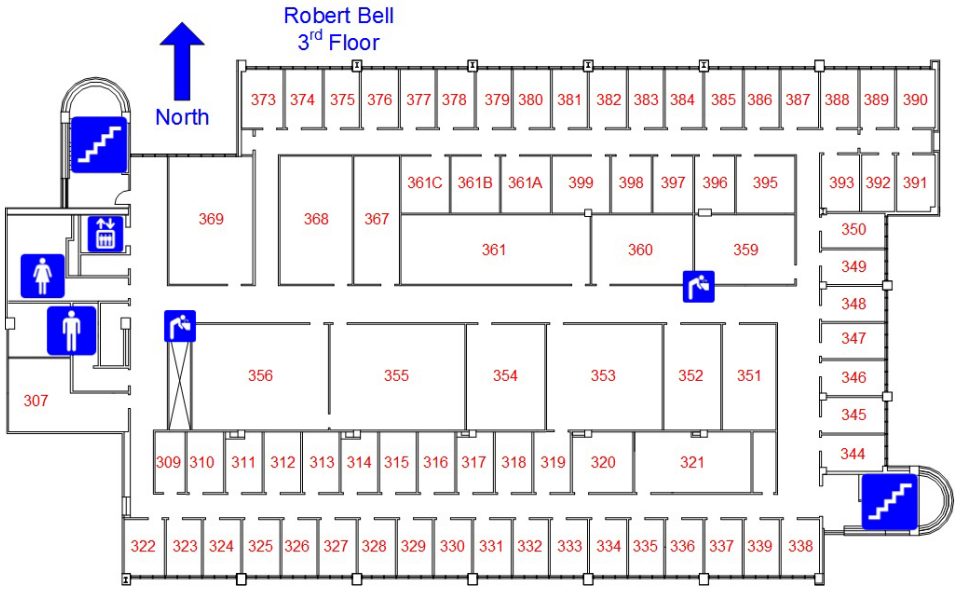
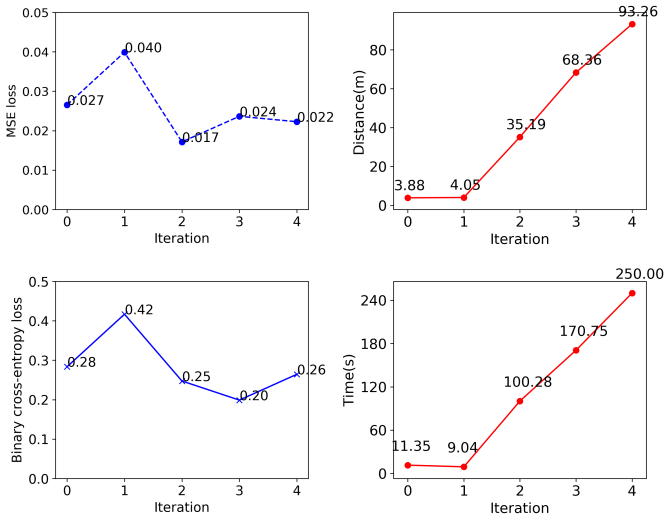


Fig. 6: Data collection trajectories. Blue arrow indicates the trajectory for collecting recording policy dataset and the red one shows the trajectory for collecting navigation policy datasets for all iterations.



(a) **(b)**

Fig. 7: Averaged evaluation losses on navigation **(top)** and recording policies **(bottom)** over five training iterations are shown in **(a)**. Traveled distance (**top**) and time (**bottom**) to collision in each iteration are shown in **(b)**.

have reserved 10% of data samples as the validation set in each iteration. We have measured the losses for navigation and recording policies, which are the MSE between reference and navigation policies and the binary cross-entropy loss respectively. They are averaged over each training iteration as shown in Fig.7a. The averaged losses of navigation and recording policies both reach their largest at the first iteration in the self-supervised stage. We conjecture this is due to the discrepancy between human and sensor policies. After this iteration, the navigation policy rapidly converges to reference policies and then fluctuates within a small range.

2) Performance of Navigation Policy: Fig.7b shows the performance of navigation policy during the five training iterations. It reflects the results of averaged losses in Fig.7a: there is no performance gain in the first two iterations and then the performance linearly increases after the first iteration of the self-supervised learning phase. We have noticed that the navigation policy is able to safely navigate the robot across the hallway without any collision within the time limits at the last iteration, which is also clearly indicated by the rightmost point on Fig.7b. These observations imply that *MS3L* can safely and autonomously navigate robots in complex real environments after sufficient iterations of training.

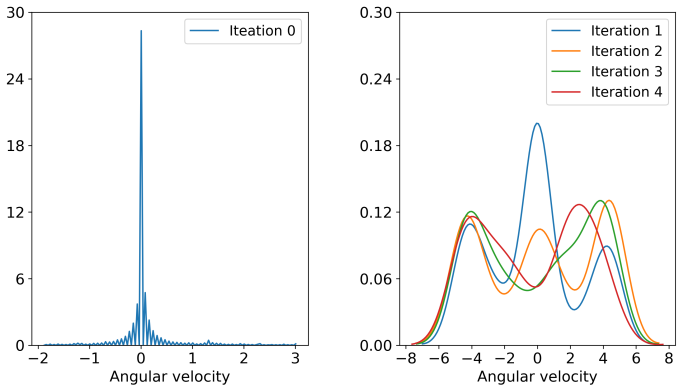
3) Performance of Recording Policy: We also analyze the performance of recording policy during each iteration. As an illustration, Table I shows the number of training samples collected during each iteration. In the pre-training stage (Iteration 0), human operator navigates the robot to collect the initial dataset within $250s$ at $30fps$. In the self-supervised learning stage, recording policy is extremely effective in that it drastically reduces the number of collection samples by ruling out those deficient labeled observations.

TABLE I: Training data collected during 5 iterations

Iteration	0	1	2	3	4
# of observations	7500	351	853	790	335

In addition, distributions for angular velocity during each iteration are are another metric to measure the performance of the recording policy. They are evaluated and presented in Fig.8. The training samples collected in the pre-training stage primarily consist of data with angular velocity $0rad/s$ as in Fig.8a, which represents going forward. This singular value can't contribute anything useful to data labeling for training.

In contrast, the data collected is more uniformly distributed in self-supervised learning stage as in Fig.8b. This means these data contain more stateful information such as turning in different angular velocities and are thus significantly useful for training. This also indicates that, in addition to reducing the effect of deficient labeling from sensor policy, *MS3L* also ensures only hard samples that have not been seen in the past iterations are recorded and trained by navigation policy.



(a) **(b)**

Fig. 8: Angular velocity distributions over the pre-training stage (**a**) and self-supervised learning stage (**b**).

C. Recording Threshold β

An important factor that affects the framework and recording policy is threshold β that controls what observations should be considered as hard samples. We use three different β values to evaluate how it influences the performance of the system as shown in Fig.9. The performance has been evaluated at the third iteration of training process[4]. As we can observe from the figure, the traveled distance decreases as we reduce β. We conjecture that this is because the system learns too much from the suboptimal sensor policy with a small β. When β is near 0, the recording policy is merely utilized and the data collection is not constrained, which degrades the performance of navigation policy. A large β value is therefore necessary to ensure the recording policy to perform effectively during the self-supervised learning stage.

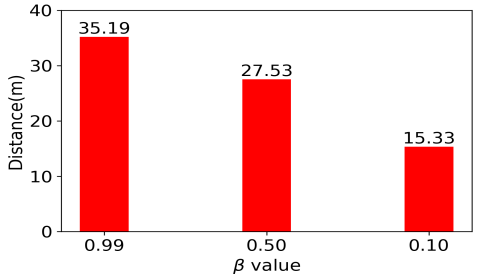


Fig. 9: Traveled distances with β values of 0.99, 0.5, and 0.1.

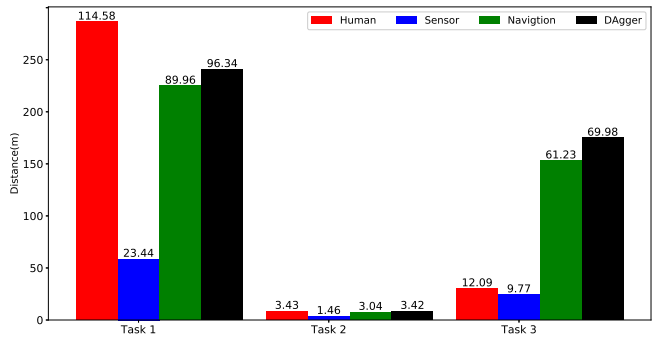
[4]Evaluating the performance after the third iteration is dangerous when β is 0.5 or 0.1 because the performance of navigation policy degrades quickly.

D. Performance of Navigation After Training: Comparison with Baselines

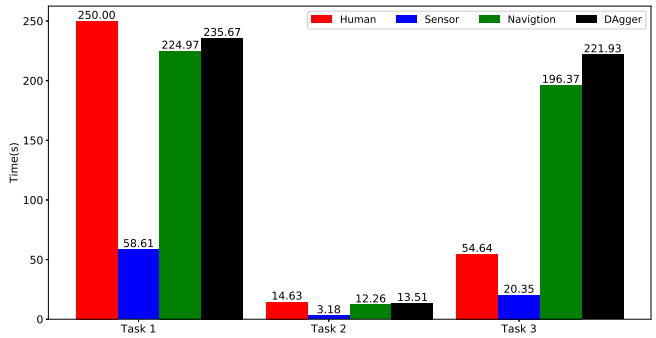
After training, it is of utmost interest to assess the actual performance of the trained *MS3L* robot in real environments. Since the robot has been trained and learned from human and sensor policies, we define these two policies as baselines. In addition, DAgger algorithm is also used to compare with our framework. The training process of DAgger is the same as our framework except that it requires all 37500 images to be recorded during the five training iterations and it needs human to correct the actions. It is valuable to compare the robot navigation performance with these baselines. We have tested the comparison over three tasks. For fair comparison, during the tests, the human operator can only perceive the environments at the first-person view from the cameras of the robot as it does at its own. The results show that *MS3L* is robust to learn from noisy and suboptimal policy. We have also noticed that *MS3L* surpasses the sensor policy by a large margin in most of the tasks, it is also able to achieve near-human performance in two tasks, and even outperforms the human operator in one task. Two human operators participate in each task. Every task is performed 5 times and the results are shown in average distance in meters and time in seconds. The three tasks are presented as follows:

- The first task is to navigate the robot through the hallway during normal business time. While our robot is trained in this environment, the training data collected is during break time when there are few people walking in the environment. We set this task during business hours when the walking traffic is heavy inside the hallways to examine how robust *MS3L* is to deal with unseen cases after training.
- The second task is navigate the robot through a classroom where chairs and tables are main obstacles. This environment is difficult even for human operator.
- The third task is navigate the robot through the 3rd floor with Gaussian noise with zero mean and unit standard deviation added to the controller that emulates hardware malfunction. We would like to use this test to show the robustness of *MS3L* in case of hardware failures or human manipulation mistakes.

Fig.10 shows the results of comparisons in these three tasks. We can observe that the performance of *MS3L* exceeds sensor policy in every environment tested. Human policy as the baseline has a slightly higher performance in the first tasks. However, in case of hardware failures or operation mistakes as in the third task, human operator is not able to deal with such unexpected events, while *MS3L* is robust to these noises and greatly surpasses human performance. This is because human can only perceive a first-person view, it is likely that he/she only knows partial environment. When the controller is noisy, the human operator is not able to make meaningful adjustments. DAgger achieves slightly better performances in all three tasks compared to *MS3L*. However, DAgger requires $2.8\times$ more examples to be recorded compared to our framework. In addition, it needs heavy human labeling which does not exist in our work.



(a)



(b)

Fig. 10: Traveled distance (**a**) and time to collision (**b**) in the test stage after training in three tasks.

From the experiments, the sensor policy often fails when the robot faces a plain wall or narrow corridor where depth information can not be reliably estimated from stereo images correctly. The navigation policy fails in avoiding multiple objects. This is because the environment is not completely observable and the policy only considers the current observation, not any historical information. For example, it forgets the position of the previous object while trying to avoid the current one.

V. Conclusion

In this work, we proposed a solution, Multi-Sensory Self-Supervised Learning(*MS3L*), for autonomous robotic navigation. *MS3L* is an imitation deep learning with sensor fusion. It is able to perform robotic navigation tasks in real environments. *MS3L* designs a suboptimal sensor policy that replaces human operators after the initial training. A recording policy is then proposed to restrict learning from the suboptimal policy that likely lead to serious robotic damage. Extensive experiments in real indoor environments have demonstrated that *MS3L* is able to successfully and reliably surpass the suboptimal policy that it learns from and even outperforms the performance of human operator in unexpected events such as hardware failures.

Acknowledgment

The authors would thank all anonymous reviewers for their precious time. This material is based upon work supported by the National Science Foundation under Grant No. #1408165 and #1726017, as well as by NVIDIA through an academic GPU grant.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.

[4] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive Mapping and Planning for Visual Navigation. 2017.

[5] W. G. Hatcher and W. Yu. A survey of deep learning: Platforms, applications and emerging research trends. *IEEE Access*, 6:24411–24432, 2018.

[6] D. Isele, A. Cosgun, K. Subramanian, and K. Fujimura. Navigating Intersections with Autonomous Vehicles using Deep Reinforcement Learning. *arXiv preprint arXiv:1705.01196*, 2017.

[7] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *Computer Science*, 2014.

[8] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg. Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations. In *IEEE International Conference on Automation Science and Engineering*, pages 827–834, 2016.

[9] M. Laskey, S. Staszak, W. Y.-S. Hsieh, J. Mahler, F. T. Pokorny, A. D. Dragan, and K. Goldberg. Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 462–469. IEEE, 2016.

[10] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[11] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *Computer Science*, 8(6):A187, 2015.

[13] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[14] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[16] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena. From Perception to Decision: A Data-driven Approach to End-to-end Motion Planning for Autonomous Ground Robots. *arXiv preprint arXiv:1609.07910*, 2016.

[17] R. Rahmatizadeh, P. Abolghasemi, A. Behal, and L. Bölöni. Learning real manipulation tasks from virtual demonstrations using LSTM. *arXiv preprint arXiv:1603.03833*, 2016.

[18] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on robotics and automation*, 8(5):501–518, 1992.

[19] S. Ross, G. J. Gordon, and D. Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *AISTATS*, volume 1, page 6, 2011.

[20] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. Learning monocular reactive uav control in cluttered natural environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1765–1772. IEEE, 2013.

[21] F. Sadeghi and S. Levine. $(CAD)^2$ RL: Real Single-Image Flight without a Single Real Image. *arXiv preprint arXiv:1611.04201*, 2016.

[22] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[24] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

[25] L. Tai, G. Paolo, and M. Liu. Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation. *arXiv preprint arXiv:1703.00420*, 2017.

[26] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[27] S. Thrun and J. J. Leonard. Simultaneous localization and mapping. In *Springer handbook of robotics*, pages 871–889. Springer, 2008.

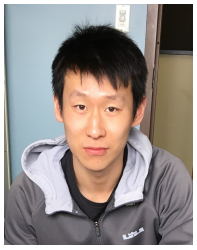
[28] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. 2017.

[29] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2015.

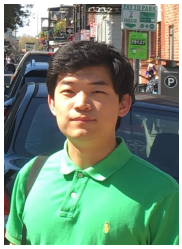
[30] H. Xu, W. Yu, D. Griffith, and N. Golmie. A survey on industrial internet of things: A cyber-physical systems perspective. *IEEE Access*, 2018.

[31] J. Zhang and K. Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

[32] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. 2016.

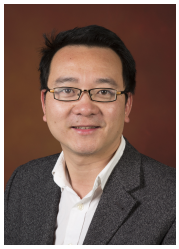


Junhong Xu received the B.S. degree in Computer Science from the collaborative program of Ball State University, Muncie, IN, USA and Kunming University of Science and Technology, Kuming, China, in 2016. He is currently pursuing the M.S. degree in Computer Science at Ball State University, Muncie, IN, USA.



Shangyue Zhu received the B.S. degree in Computer Science from the collaborative program of Ball State University, Muncie, IN, USA and Xi'an University of Posts and Telecommunications, Xi'an, China, in 2015. He received the M.S. degree in Computer Science at Ball State University, Muncie, IN, USA, in 2017.

Hanqing Guo received the B.S. degree in Telecommunication Engineering from Chongqing University of Posts and Telecommunications, Chongqing, China, in 2015. He is currently pursuing the M.S. degree in Computer Science at Ball State University, Muncie, IN, USA.



Shaoen Wu Shaoen Wu (M'14- SM'16) received the Ph.D. degree in computer science from Auburn University, Auburn, AL, USA, in 2008.

He is currently an Associate Professor of computer science with Ball State University, Muncie, IN, USA. He was an Assistant Professor with the School of Computing, University of Southern Mississippi, Hattiesburg, MS, USA, a Research Scientist with ADTRAN Inc., Huntsville, AL, USA, and a Senior Software Engineer with Bell Laboratories, Qingdao, China. His current research interests include wireless and mobile networking, cyber security, cyber-physical systems, and cloud computing.

Prof. Wu has served on the chairs and the committees of various conferences, such as the IEEE ICNC, IEEE INFOCOM, ICC, and Globecom, and an Editor for several journals. He was a recipient of the Best Paper Award of the IEEE ISCC 2008 and the ANSS 2011.