Introspective Perception through Identifying Blur, Light Direction, and Angle-of-View

Mary Theresa Hatfalvi

CMU-RI-TR-22-33

August, 2022

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Aaron Steinfeld, *Advisor*
David Held
Ming-Fang Chang

Submitted in partial fulfillment of the requirements for the degree of Masters of Science in Robotics.

Copyright © 2022 Mary Theresa Hatfalvi. All rights reserved.

To my mother, my first teacher of life, liberty and love for God and others.

Abstract

Robotic perception tasks have achieved great performance, especially in autonomous vehicles and robot assistance. However, we still often do not understand how and when perception tasks fail. Researchers have achieved some success in creating introspective perception systems that detect when perception tasks will fail, but they usually are tuned to only specific, connected perception tasks and do not identify the reasons for failure such as blur, light direction changes, and angle-of-view changes. To address this shortcoming, we work on the use case of a perception task that determines non-destructive block removal from a scene. We design a combined introspective perception system that detects three common failure types: blur, light direction, and angle-of-view. We split these failure cases into two target areas of research: Blur Detection & Classification and Light Direction & Angle-of-View Failure Prediction.

One of the main failure cases for perception tasks is due to blur in the camera image. Blur can occur from issues with motion, lighting, and focus. Proper classification of the blur type is important for determining corrective robot action (e.g., slow down, turn on a light, etc.), but most blur classification techniques do not classify lighting-related blur properly. Here, we present a new approach by including two new types of blur in blur detection and classification and evaluate its performance using an extension of a standard image dataset with an eye towards informing subsequent robot action.

Another key challenge in robot perception is due to changes in light direction and angle-of-view. To this end, we have identified if certain angles-of-view or light directions are more likely to cause failures than others. Here, we present a failure prediction network that used a new dataset to demonstrate how perception task performance can be predicted and explained through light direction and angle-of-view.

We combined these subsystems into one introspective perception system that allows us to identify failure in the perception task. We show that using an introspective perception system that can identify at least these three failure types have risk mitigation benefits for a perception task.

Acknowledgments

I would like to thank my thesis committee: Aaron Steinfeld, David Held, and Ming-Fang (Allie) Chang. They have given great support and feedback in this research. I could not have done this without them. I would like to especially thank my advisor Aaron Steinfeld, who has allowed me to take Sarthak Ahuja's work to another level that furthered my interest and knowledge in computer vision and machine learning. Without his knowledge and support throughout my graduate program, this research would not have happened. I would especially like to thank Tesca Fitzgerald, who even though she was a post-doc in the program, has acted as a co-advisor and has pushed to produce quality research. Her support and help has been a great resource.

I would especially like to thank the lab for their generous support and encouragement in my research: Allan, Fern, Kirabo, Liz, Oscar, Sam, and Zhi. They have given advice and help when asked and I am extremely grateful.

I would like to thank all of my previous professors and teachers that have given me the tools to grow in my field of research. I also want to thank my homeschooling teachers, especially Mrs Hahne. She started the homeschooling robotics group (FTC Team 4081) that got me into robotics in the first place. She gave me the inspiration through her math classes and her mentoring that I could do great things.

My friends and family have seen my journey through research and classwork. Thank you to my husband's family for their support and encouragement. I would especially like to thank all of my siblings: Anne, Kimberly, Julie, Susan, Gregory, and Clare. They give me inspiration to make them proud and to give them encouragement to follow their dreams. I would like to thank my Dad, Peter, who gave me my first introduction to engineering and problem solving. I want to give a special thanks to my mother, Kathleen, who sacrificed her engineering career for her kids and gave me my hard work ethic and motivation when things got difficult. I want to thank my husband Ben Sarkis who has sacrificed to move to Pittsburgh, got a new job and has worked while studying for his masters so I could pursue mine. I am entirely grateful and humbled to be with him.

And finally I would like to thank God who led me to CMU, who always

answered my prayers even when it wasn't what I initially asked, and who has given me peace and rest during these times.

Funding

This work was supported by the Office of Naval Research (N00014-18-1-2503).

Contents

Chapter 1

Introduction

1.1 Motivation

In recent years, robot perception has greatly advanced for performing very complicated computer vision and machine learning tasks. Some examples of complicated perception tasks include driving autonomous vehicles and robot assistance with human tasks, among other robot applications. However, even when great accuracy has been achieved in execution of these perception tasks, it is still unclear when, how and why these perception models fail. Regardless, failure risk is dependent on the perception task due to "garbage-in, garbage-out" issues. Some tasks have small risk when failure occurs whereas tasks such as rescue missions, failure is extremely detrimental and has to be avoided at all costs. For machine learning networks, failure comprehension is especially difficult. The simple solution that training with more data results in fewer failures is difficult to achieve for reasons of memory and time. Thus, researchers have looked at how humans solve these issues.

Humans have great accuracy and knowledge of their environment. They know if the scene is too dark, if their vision is blurry, or if they cannot see their object of interest because they are not at a good angle-of-view. Having self-awareness in perception, whether from seeing, touching, or tasting, is something humans at an early stage in life learn and eventually master. Humans learn how different objects can have different texture or how different shapes are better than others for stacking on top of each other. Thus leading to a natural self-awareness and adaptation to

perception tasks, such as non-disruptive block removal. Robots, however, do not have this natural awareness. Machine learning networks are given an input and, from their trained parameters, produce a prediction that best represents the training data that they understand. These networks are very robust and machine learning professionals and computer vision experts have used neural networks for many industrial robot applications. As noted previously, networks can be sceptical to light changes, blur, camera angle, testing mismatches, etc., that causes perception failures. Thus, we ask the following research questions: Can we build a self-aware perception system that looks at these failures and can predict them like humans do? Can we have a risk mitigation detection based on identifying the potential risks which allow for better correction?

1.2 Problem Statement

This area of research is called introspective perception, which is the act of self-awareness in the current state [7]. Introspective Perception is the ability for the system itself to predict if and when it will fail given the current input. The idea is that self-awareness leads to better risk mitigation. Existing systems have been tried in many perception tasks. However, most of these systems perform all-in-one failure prediction and do not understand until after the fact why the failures were predicted. Thus, we ask the research question: **Can we predict failure through determining common failure issues and thus developing reasons for the failures?**

In this thesis, we study this question of how to predict failures through identifying failures by first defining the common perception failure types: Blur, Light Direction and Angle-of-View. We then create subsystems that handle these three categories and combine them into one introspective perception system for predicting failure for a perception task. We successfully predicted task failure and provide a risk mitigation metric to measure how failure can be avoided using our system. We will go into each of these problem areas, explain them, and describe the problem space we used to test our introspective perception system.

1.2.1 Perception Failures through Blur

As one example, *consider a mobile robot entering a poorly-lit room looking for hazards. To determine the accuracy of its object classification, it uses blur detection to determine if motion or focus blur are present and have introduced error into its perception network. If it detects either type of blur, it can reduce perception errors by slowing the motion of its camera or refocusing, respectively. However, suppose that the robot enters that room and incorrectly interprets its perceptual errors as resulting from an out-of-focus blur. Its strategy of refocusing the camera would not reduce the blur and resolve the perceptual error.* This scenario demonstrates a major robot perception error that can result in task failure.

Defining different categories of blur is important for enabling robots to choose the correct mitigation strategy. Recent deep learning methods can identify focus and motion blur in an image with high accuracy [19, 48]. After classifying out-of-focus blur in an image, a robot could reduce this blur by changing the camera's aperture size. To address motion blur, the robot could slow its own motion (thus also reducing the camera's motion), increase shutter speed, or wait for a moving object to pass. However, these strategies do not account for blur due to over/underexposure (Figure 3.9), which similarly cause images to lack sharp and clear edges and may cause perceptual errors. Using current blur classification methods, these images would be incorrectly classified as containing focus or motion blur, yet need different repair strategies (e.g., turning on a light to resolve underexposure). We therefore looked to address this by creating and expanding a blur dataset that identifies not just motion and focus, but over-and underexposure. We then evaluated this classification through training and testing conditions which test different dataset creation methods. We then looked at classifying these new blur types and apply blur detection and classification to our introspective perception system.

1.2.2 Perception Failures through Light Directions and Angles of View

As another example, *another mobile robot enters a room with lighting in a different direction than before. This robot's task is to find water in the room. The image is*

mostly well exposed but the scene is slightly different from what was in the training dataset because of light direction change and thus the robot fails to find water in the room. This failure is hard to understand and is usually identified as a network out-of-distribution error. However, for perception tasks such as rescue robots, mistakes like this are not acceptable errors.

The usual solution to this problem is to improve the diversity of the training and validation datasets used. Networks depend on images and objects to appear similarly in real deployment as they do in training and validation. In reality, this is difficult because it requires extensive training datasets and validation runs that exhaust all possible scenarios (lighting changes, angle changes, etc.). Another solution is building in uncertainty to the network and thus reporting when the detection is uncertain for risk mitigation [11, 26]. This solution is more stable, but just measuring uncertainty does not provide a possible reason for failure.

In previous works [7, 51], introspective perception is done before the perception task to predict failure or success. We explored this solution because it allowed for risk mitigation before the task is executed. However, even though these systems are highly successful in mitigating risk and predicting failure, they do not give information about the cause of the failure, such as due to blur in the scene. These introspective perception systems [7, 34] found post-hoc that overexposure, focus, underexposure, and/or glare images were most commonly flagged potential failure. Knowledge of the type of failure during task execution gives direction for possible corrections that can be done in real time. This helps perception systems be more self-aware. We used a failure prediction network to predict failure for a dataset that includes light directions and angles of view. We then performed training and testing splits that give evidence to failure prediction when light directions or angles of view are not present in the training dataset.

1.2.3 Perception Failures in Ahuja et al. [3]'s Physics Intuition Network (PI)

In Ahuja et al. [3]'s Physics Intuition Network (PI), the main perception task determines if a block can be be removed from a scene of multiple blocks without disturbing the other blocks. PI accuracy depends on well-defined object masks and clear, well-lit

images to predict which block is removable in the scene. In real world applications, it is hard to create good object masks because they can depend on accurate salience detection and color thresholding. Color thresholding is a noise-prone computer vision application since its accuracy depends on consistent lighting, color correction, and the absence of shadows. This is shown in Ahuja et al.'s model where simulation testing had high accuracy vs. real world testing. In our research, we looked to further resolve this real-world accuracy disconnect problem through introspective perception by identifying the three failures that specifically caused this PI system to fail.

1.3 Thesis Objective and Contributions

The main objective is to create an introspective perception system (IPS) that checks for uncertain inputs from the environment through three common perception issues: blur, light direction and angle-of-view. This thesis makes the following research contributions:

- Expansion and refinement of current blur definitions to reflect unaddressed blur issues in robot systems;
- Enhancement of an existing motion and focus blur dataset to include over- and underexposure blur for classification evaluations;
- A dataset for identifying failure through light direction and angle-of-view for Ahuja et al. [3]'s Physics Intuition Network (PI);
- A combined Introspective Perception System (IPS) that identifies possible failures from the following vision issues: blur, light direction and angle-of-view for Ahuja et al. [3]'s Physics Intuition Network (PI);

1.4 Thesis Outline

We will first discuss the related work (Section 2) in this area, including current blur solutions, introspective perception networks and uncertainty awareness in computer vision today. We will then discuss the methodology (Section 3) for our IPS where we describe blur, light direction, and angle-of-view issues and explain our approach to failure prediction for Ahuja et al. [3]'s Physics Intuition Network (PI). Next,

we describe our results (Section 4) for our subsystems for blur, light direction and angle-of-view and review how our IPS metrics show risk mitigation and accuracy in predicting failure. We then present limitations and discuss our approach and future research directions that we anticipate (Section 5). Our final section 6 is the conclusion, where we present final thoughts on our combined IPS system and its subparts.

Chapter 2

Related Work

Our related work includes five topics:

- **Uncertainty in Deep Neural Networks**: There are many current methods for measuring uncertainty in neural networks. We summarized the most common techniques and identify the methods that we used in our uncertainty prediction for common failure types.
- **Blur Detection and Classification**: We discussed current blur types and techniques for detection and classification and why overexposure and underexposure should be considered as types of blur.
- **Next-Best-View**: Current next-best view research is explored, and we identified its similarities to and differences from our light direction and angle-of-view failure prediction network & goals.
- **Introspective Perception**: We looked at the current introspective perception networks that have been discovered and examine our network's similarities, differences and contributions.
- **Physics Intuition Network**: We discussed the perception task of determining if blocks in a scene can be removed in a non-disruptive manner and showed that this task gives a good basis for our introspective perception network testing across blur, angle-of-view, and light directions.

2.1 Uncertainty in Deep Neural Networks

Uncertainty in Deep Neural Networks is usually defined as a given measurement that identifies the model's limitation in handling inconsistent issues that cause a prediction to be wrong. Measuring uncertainty in neural networks can impact knowledge of system failure. To date, multiple researchers have succeeded in modeling uncertainty using Bayesian methods, such as using Monte Carlo Dropout [11, 26], Assumed Density Function [26], and probabilistic NCNNs [10]. In the next few sections of this chapter, we detailed some of these methods and explained how they relate to what we do in IPS and how IPS can help close gaps in identifying failure that previous uncertainty models cannot achieve easily.

2.1.1 Types of Uncertainty in Neural Networks

Kendall & Gal [18] divided uncertainty for computer vision into two categories: aleatoric uncertainty and epistemic uncertainty. Aleatoric uncertainty is the uncertainty that comes from the dataset [18]. It is also called dataset uncertainty. Epistemic uncertainty arises from the model parameters, which is also called model uncertainty [18]. Aleatoric uncertainty can be further split into two more categories, homoscedastic uncertainty and heteroscedastic uncertainty [18]. Heteroscedastic uncertainty is caused by unusually noisy input data, leading to inaccurate outputs [18]. Homoscedastic uncertainty is normal input noise that is constant [18]. Kendall & Gal [18] shows that incorporating these two main types of uncertainty in a neural network generates better results. In this research, we addressed aleatoric uncertainty through our IPS solution of a combined failure detection that evaluates common input data issues and gives a failure prediction. Thus, it removed data that was considered to be insufficiently informative for the main perception task to generate an accurate output.

2.1.2 Neural Network Uncertainty

Gal & Ghahramani [11] proposed the use of a dropout layer during training and testing placed at each weight layer in their network. This dropout during testing allows for uncertainty to be measured by performing multiple passes on an input. This

process results in a mean (prediction) and variance (uncertainty) of the output [11]. Applying the dropout during testing and performing multiple test passes on a single input is equivalent to Monte Carlo sampling, which leads to an accurate uncertainty measurement from the variance [11]. This method works well for modeling neural network uncertainty but does not have a identification method for that uncertainty. In contrast, we identified three points of uncertainty through our combined IPS.

Loquercio et al. [26] successfully modeled both aleatoric and epistemic uncertainty for a steering task. They modeled aleatoric uncertainty through Assumed Density Filtering (ADF) and epistemic uncertainty through a Monte Carlo (MC) Dropout. Loquercio and colleagues [26] stated that ADF replaced the network activations with probability distributions that represented the data distribution. They used MC Dropout [11] to find the model uncertainty. The final model uncertainty was then combined through Bayesian belief networks because both uncertainties can be dependent on each other [26]. This method was successful in measuring uncertainty by attaching itself to a perception network without having to retrain the original model. However, it did not have a way of defining where uncertainty came from (e.g., blur, light, etc.).

Eldesokey et al. [10] converted an NCNN into a probabilistic NCNN that was trained with uncertainty in mind for the task of depth completion. They did this by first having an estimator that predicted input confidence that was optimized to assign high confidence to valid inputs, then used a probabilistic NCNN to predict the final uncertainty measurement for the input [10]. They were able to successfully outperform other state-of-the-art methods on this task, but they had to create a model for the perception task that incorporated uncertainty. Our goal was to not have to modify the original perception network and to add a failure prediction method that can be used on other perception tasks that are unrelated to depth completion.

2.1.3 Takeaways from Uncertainty in Deep Neural Networks

We discussed the categories of uncertainty that are present in neural networks as well as the current ways to measure them using Bayesian methods. In this work, we focused on identifying and mitigating aleatoric uncertainty (which arises from the

data) in our created IPS solution.

2.2 Blur Detection and Classification

One of the main reasons that perception systems fail is because of blur. We go into detail in this section about how blur is defined, how over- and underexposure can be defined as blur, and how the current blur detection and classification methods compare to our proposed system. We also discussed how current blur datasets lack the ability to define over- and underexposure as blur.

2.2.1 Defining Blur

Blur reduces the visual quality of images and can cause perception networks to fail because the edges and shapes in the image cannot be clearly defined. Image blur is commonly delineated into two categories: motion and focus [20]. Motion blur is directional and can be caused by objects moving in the scene or as a function of the shutter speed of the camera [32]. Focus blur is a non-directional blur that occurs when parts of the image are integrated by light over the area of the aperture lens [16]. This results in parts of the image lacking clearly defined edges. Well focused images can be successfully used for 3D shape and depth estimation [27] and for salient object detection [15, 43]. However, unfocused images can negatively affect both salient object detection and depth visual features.

Typically, over- and underexposed images are not defined in computer vision as blur although they result in poorly-defined edges. Overexposure is defined as a loss of highlight details in bright regions of an image [12]. Underexposure is the loss of lowlight details in dark regions of an image. Because these exposure errors have similar effects as other forms of blur on the robot's ability to detect and classify objects, we argue that the resulting images comprise new, separate categories of blur than those detected by current blur detection and classification networks. If current blur detection methods misclassify over- and/or underexposure as blur, their corrections cannot fix the perception issues, which can lead to risk-mitigation failure. We demonstrated in our results subsection 4.1.2 that current blur detection can falsely classify over- and underexposure as traditional blur, which can lead to poor correction

results. These existing processes negatively affect computer vision applications, and current IPS systems have detected relevant images as failure cases post-hoc [7, 34]. We identified and correctly classified these failure cases through the blur detection and classification subsystem in our IPS.

2.2.2 Detecting and Correcting Over/Underexposure

Overexposure is defined as uniformly high intensity regions that have to be clipped at a maximum value for the range of the camera, resulting in pixels that appear white. It can be caused by sunlight, spotlights, and High Dynamic Range (HDR) [12]. Cameras have a limited dynamic range that they can capture, but sunlight and outdoor settings can have a very high dynamic range [12]. Underexposure of images occurs when there is not enough light in a scene; thus, the image appears dark.

One current method for detecting overexposure is to convert the image to a CIE L*a*b* (CIELAB) colorspace, calculate the likelihood of overexposure and color confidence, and used the resulting two maps for color correction [12]. The major drawback to this method is that it cannot correct severely overexposed images, such as those requiring an external change to the robot's environment (e.g., the robot turning off a light). Additionally, this method cannot detect or correct underexposure.

An HDR imaging technique that corrects underexposed and overexposed images well takes different exposure images and blends them into one well-exposed image. Major drawbacks for using this technique in robotics include expensive systems, the robot needing to stay still to avoid motion blur, and overexposure (especially outdoors) [35]. Auto-exposure correction methods [23, 39] work well for everyday images taken on a phone or other industry cameras; however, in subsection 5.1.2, we demonstrated that these standard auto-exposure correction methods fail in some situations where even minor exposure changes can affect a robot's task performance.

2.2.3 Blur Detection

At its simplest, blur detection algorithms identify whether an image, as a whole or in certain regions, contains blur [20]. Some current methods are easy to implement, such as measuring the variance of the Laplacian of an image [29] and defining a variance threshold for determining how much blur can be tolerated in an image.

Smaller variance values correspond to less sharp or blurrier images. In this method, the blur threshold is applied to the entire input image rather than to specific regions of interest. This method does not classify the type of blur within the image, so a corresponding mitigation strategy cannot be determined.

Zhuo et al. [52] measured the amount of focus blur in an image by re-blurring an input image edge with a known Gaussian kernel. Then, the ratio between the gradient magnitude of the edge and the re-blurred edge gradient was found [52]. The maximum value of the ratio was used to find the unknown blur amount in the other segments of the edge [52]. This method of blur estimation creates accurate defocus maps; however, the authors point out that this method cannot determine if a blur edge was caused by either focus blur or shadows. This method also was not tested on motion blur images.

Another blur detection method returns the probability (or extent) of blur using Support Vector Machines (SVM) [13]. This detection depends on a threshold parameter ($0 \leq p \leq 1$; p is the blur probability). While many robot systems can allow for some blur, the appropriate threshold is domain-dependent (differing, for example, for rescue missions vs. autonomous driving). Depending on where the blur is in the image, even a small amount of blur can be detrimental.

Deep learning has become a staple for detecting motion and focus blur [14, 19, 48]. However, some of these networks cannot classify blur for the purpose of correction and none of these networks can distinguish over-and underexposure blur, which can thus lead to potential blur mitigation failure.

2.2.4 Blur Classification

Image segmentation is used in many computer vision applications, such as image classification [5, 25, 36], and it can be used to classify local blur regions (as opposed to labeling an entire image as blurry). As a result, segmentation can be used for both blur detection and classification, and different types of blur can be detected within a single image. For example, Su et al. [42] presented a blur classification method that results in high accuracy for distinguishing between motion and focus blur through an alpha-channel constraint where the shapes of the blur are examined. According to Su et al. [42], the alpha channel feature is the variation of an alpha

distance array. They stated that motion blur regions have larger distance values than focus blur regions and thus a threshold can be used to define focus or motion blur [42]. However, this method relies on the blur detection region boundary threshold that determines the size of the blur region. This solution is not always ideal because blur identification should not rely on changeable thresholds. In contrast, Point Spread Function estimation (PSF) classifies both defocus and motion blur within image segments, but its results are highly dependent on the width and height of the grid elements [13]. Other blur detection classification methods [19, 24, 49] accurately used image segmentation to separately identify motion and focus blur. However, these methods cannot classify blur due to over- or underexposure, which is necessary to enable a robot to select the appropriate mitigation strategies to reduce blur.

Recently, deep learning research has successfully classified and detected motion and focus blur [19, 50]. However, it does not classify over and underexposure as blur. Kim et al. [19] used an encoder VGG-19 network and a decoder U-Net network to learn unique identifying features for each blur type. We expanded on this network by adding two additional classification labels to each multi-layer output.

2.2.5 Dataset

Currently there are a few datasets with motion and focus blur segmented and labeled within images; however, none include labels for segmented over- and underexposure blur. The CUHK dataset created by Shi et al. [38] contains images with segmented motion and focus blur regions, but does not contain any segmented under- or overexposed images. Exposure datasets such as [1, 44, 45] contain images that are captured and/or rendered at multiple exposure levels, but do not contain labeled motion and focus blur regions. One major contribution of our work is the creation of a new dataset containing all four blur types. We did this by overlaying real and synthetically created blur combining the CUHK and PHOS datasets, and created a new dataset with labeled regions for motion, focus, over- and underexposed blur. We used this new dataset for training and testing a blur detection and classification network. Our work used the CUHK dataset created by Shi et al. [38] as a baseline for training and testing motion and focus blur within images. We used the PHOS dataset [44, 45] as a baseline and for creating our training and testing over- and underexposure blur in

images. We synthetically created training over- and underexposure images for the purpose of expanding the dataset creation methods for over-and underexposure blur. To validate that our synthetic overlay of under- and overexposure blur is generalized to real-world images, we recorded our own over- and underexposed images with the real-world clutter scenes from Ahuja et al. [3] and random real-world objects. We then expanded this dataset creation method to our final collected IPS blur dataset that was used for our IPS subsystem for detecting and classifying blur.

2.2.6 Takeaways from Blur Detection and Classification

As discussed, we defined blur as including motion blur, focus blur, and also over- and underexposure in images. This is because over- and underexposure can have similar characteristics to blur but can be incorrectly defined and incorrectly classified as blur, which leads to poor perception correction. The current blur detection and classification methods perform well for motion and focus blur, so we expanded a current network that utilized image segmentation classification to classify over-and underexposure. We extend a current blur dataset with motion- and focus-blurred images by adding real and synthetic over-and underexposure images to learn those newly defined labels. We created training and testing scenarios that evaluated our synthetic over-and underexposure creation and gave evidence that this creation method produces accurate classification results for real over-and underexposure objects.

2.3 Next Best View

Next Best View (NBV) is a research area where the objective is to determine the next optimal sensor position that gives the most information about a scene. This area is very useful for finding the best position for a camera to minimize the number of necessary images and maximize the information obtained from the scene. Our light direction and angle-of-view failure prediction is similar because it also looked at angles of view in a scene for determining if it is a possible failure case in the perception task. Next best view is used in 6D object pose [8], 3D exploration [4] and 3D object reconstruction [30, 46].

2.3.1 Current Next Best View Algorithms and Tasks

Connolly [6] proposed two algorithms using NBV for scene reconstruction, planetarium and normal. Connolly [6] represented the scene reconstruction as an octree which had nodes with voxels labeled as empty, occupied, or unseen. The planetarium algorithm measured a "sphere" of unseen voxels in the scene and determined the NBV where the most unseen voxels are present [6]. This algorithm is very simple and completes scene reconstruction but does not have information about how light direction or angle-of-view influences its next best view choices and this algorithm can be computationally expensive for time depending on the sampling resolution. The normal algorithm is faster because it used the unseen and empty voxel faces in the octree to determine NBV [6]. Again, this algorithm does not have light direction or angle-of-view failure information on the NBV performance.

Wong et al. [46] defined their objective function for NBV to find the view where the largest number of unknown voxels can be seen. Voxels are representations of an occupancy 3D grid of a cubic model that is used to make a reconstructed object [46]. Surface normals were then summed for each visible unknown face and the greatest unknown view was chosen [46]. This method preforms well for object reconstruction; however, even the authors note how different initial views can affect object reconstruction results. Pito [30] used NBV to plan the least number of views necessary to get accurate 3D reconstruction. This method adapts to angle-of-view changes but does not address the question how different angles of view or blur can affect NBV's performance. We addressed this through our IPS, which we propose will aid in understanding how to get the most information for mitigating risk from identifying failure cases.

Bircher et al. [4] created a NBV algorithm to explore unseen space in 3D map exploration. An octree and voxel representation was used to represent unseen and seen space. They create a planner where the objective function was built to maximize the information gain of the generated unknown paths [4]. According to Bircher et al. [4] (pg. 1463), the "gain is the summation of the unexplored volume that can be explored at the nodes along the [octree] branch." Exploratory networks are good for perception tasks, such as aerial vehicle exploration. However, this approach lacks knowledge of failure scenarios where next best view is affected by possible light

direction differences or angles of view differences in the scene.

Doumanoglou et al. [8] used Hough Forests to store features for leaf nodes that are used for determining the NBV for finding the 6D object pose. Their auto-encoder produced features that were input into their NBV search in which the best view was found that minimized the expected entropy [8]. During training, viewpoints were simulated to determine the expected entropy from those viewpoints for finding the next best view [8]. View with occlusion was accounted for in the entropy calculation [8]. They tested their network with a dataset that included different lighting illumination. They also tested their next best view by moving the camera to 10 different positions where they outperformed three other baselines. On the different lighting illumination dataset, they were able to get an average accuracy of 89.1 of 6D object pose estimation. This algorithm is very robust but is specifically tuned for 6D object pose estimation and does not measure failure of the 6D pose estimation based on blur, light direction or angle-of-view.

2.3.2 Takeaways from Next Best View

Next Best View algorithms are very robust with using voxels and optimization algorithms for finding the best position to be at for the most information gathering. Very few of these works take light directions failure into consideration and usually assume that an object can be viewed at multiple angles. We wanted to explore the possibility of being able to predict failure at other views that are unseen. We created a dataset where we included three light directions and three angles-of-view. We created different dataset splits to evaluate how light directions and/or angles-of-view that are not present in the training dataset preform on the task of failure prediction. We showed that we can include this subsystem with our IPS and give failure predictions that take into account light direction and angle-of-view.

2.4 Introspective Perception

Objective self-awareness is the ability to focus on oneself and to observe their reaction to things within themselves thus "..., he is the object of his own consciousness..." ([9] pg. 2). The idea is that when someone is more self-aware, then their self-

perception accuracy goes up. The hypothesis is that self-awareness can improve self-perception accuracy. The field of uncertainty in deep neural networks (Section 2.1) has shown that we can have neural networks measure their uncertainty which itself is a version of self-awareness. Introspection for robots is the act of self-awareness in the current state [7]. Introspective Perception is the ability for the system itself to predict if it will fail. It is motivated by the idea that self-awareness can lead to better risk management. Introspective perception has been successfully tried in multiple perception-based systems [7, 22, 34]. However, very few of the systems used datasets that have defined failure conditions such as angle-of-view, and they are usually tuned to specific perception tasks. They also do not design their IPS with the failure identification in mind. Post-hoc analysis is usually performed to identify those failures. We go beyond basic IPS failure prediction and created our IPS with identification of more common failure conditions in mind.

2.4.1 Introspective Perception in Computer Vision Tasks

Kuhn et al. [22] created an image segmentation network that took introspective failure prediction info account for improved task performance. They build an introspective failure prediction network by reusing the encoder decoder network from the original task and retraining just the decoder part to predict error maps [22]. Performance for this image segmentation task was improved when the authors added this introspective prediction. The authors used precision-recall curve to evaluate their network against the baseline. We reused the original model that was used for the original perception task to predict failures for light direction and angle-of-view in our IPS. We used precision and recall as additional metrics for evaluating our IPS failure prediction accuracy and PI accuracy with and without our IPS connected.

Daftry et al. [7] built a introspective perception system for an MAV using two modified AlexNets [21] that captured spatial and temporal features in an image input. These features were combined onto one fully connected layer that was fed to a softmax output [7]. This output was a failure prediction score that was used in real time to make flight decisions [7]. Their prediction was if the input is unreliable (1) or reliable (0) [7]. They proved that the Error vs. Failure Rate was improved with the addition of their IPS [7]. However, this IPS was not built for stationary tasks. We used a

single modified AlexNet [21] similar to this to predict our failures for the physics intuition network. In post-hoc observations, Daftry et al. [7] discussed how they found that specifically overexposed and out-of-focus images are the common sources of failure. This gave us motivation that these sources of failure would be key for our introspective perception system to detect. Our blur detection and classification found focus, motion, over-and underexposure images and marked them as failure.

Rabiee & Biswas [34] built an IPS for their task of obstacle avoidance that used the AlexNet [21] architecture. This network output four labels per image patch of True Positive, False Positive, False Negative and True Negative that were filtered with a uncertainty measurement for a final failure prediction for that image segment [34]. The authors did analysis on the images that their obstacle avoidance failed on most, which was dark spots and reflections [34]. This analysis was performed post-hoc. We found these failures in our IPS by identifying them in the process of detection, which can allow for more accurate correction responses to these failures.

For general perception systems, Zhang et al. [51] built a system called ALERT which predicted failure for vision systems. The ground truth labeling for their failure network was produced by their perception task network. We produced our ground truth that is used for our failure prediction from light direction and angle-of-view through finding failures cases in our perception task. They used two evaluation metrics: Accuracy vs. Declaration Rate (ADR) and Risk-Averse Metric (RAM) [51]. The Declaration Rate is the the proportion of test images that the system does not output a decision because it will prove a false detection [51]. The Risk-Averse Metric (RAM) is the trade-off risk of the system making an incorrect decision versus not making any decision [51]. We used RAM to show how our IPS system has risk mitigation capabilities. ALERT is very good at predicting failures for vision systems but does not identify failures. We went a step further in our IPS by adding a subsystem that classifies the failures that come from blur. We showed through our IPS testing scenarios how each failure type effects PI performance (subsection 4.3.1).

2.4.2 Takeaways from Introspective Perception

Introspective perception has been studied in vision systems extensively. Success has been achieved in computer vision systems such as ALERT [51] and the system by

Daftry et al. [7]. As stated before, these systems usually fail to identify till post-hoc the causes of failures. Failure identification is needed for proper correction measures to be taken. We used a modified AlexNet [21] that Ahuja et al. [3] used for their perception task to predict failure from light direction and angle-of-view. We also expanded the current IPS with the addition of a blur detection and classification network. We made one combined IPS that can distinguish blur, light direction and angle-of-view, which can lead to better perception task risk mitigation for high-risk scenarios.

2.5 Physics Intuition Network (PI)

Ahuja et al. [3]'s perception task goal was to be able to predict which blocks can be removed without disturbing other blocks in a scene. The two scenarios they used for evaluating their non-disruptive network were Clutter (blocks in multiple formations) and Jenga (the game where blocks are formed in a deliberate placement). They argued that conditioning the target object using masking across multiple views allows for better results [3]. The input to their network was an RGB image that was masked for the specific object of interest and the output was either 0 for unstable or 1 for stable. This task was solely decided through the perception input; thus, it allowed our IPS to be tested without another input interference. Another contribution that this paper made is the training trough synthetic datasets and the effect on testing through real datasets. This dataset creation method has been a growing area of research since it allows for easier dataset creation than real world image collection [3, 17, 37]. They used V-REP and MuJoCo simulation platforms to generate the datasets of blocks for training. Their real-world accuracy for Clutter scenarios ranged from 80% (single-view images) to 86% (multi-view images) and for Jenga scenarios ranged from 78% (single-view) to 83% (multi-view) [2]. These real world accuracy results were lower than the simulation data accuracy. Multi-view took into account different camera positions in the network input and performed better than single-view.

A possible reason for the drop in accuracy when moving from simulation to the real world theorized by Ahuja et al. [3] was light sensitivity and shadows. This is one of the main reasons that we chose this perception task. This gap of synthetic and real-world images gives evidence for the need for an IPS that can check for the

dataset inconsistencies. Because Jenga scenarios are less common, we focused on only the Clutter scenario trained using MuJoCo noisy images and used the Ahuja et al. [3] PI's pre-trained AlexNet weights to test and evaluate our IPS. We showed that we can increase accuracy and improve risk mitigation for PI through using our IPS.

2.6 Related Work Summary

We summarize our related work and how it relates to our solution in the following:

- **Uncertainty in Deep Neural Networks**: Using Bayesian methods is common for measuring uncertainty in deep neural networks. However, these forms of uncertainty measurement do not have a way of defining and predicting failure through light direction, angle-of-view and blur. Our IPS solution lowered uncertainty through detect these forms of failure for PI.
- **Blur Detection and Classification**: Blur is defined usually into two categories: motion and focus. Previous works do not classify over- and underexposure as blur; however, they can have similar characteristics to blur and be incorrectly classified as blur. This leads to poor perception correction. We expanded a current blur detection and classification network that adds over-and underexposure labels. We extended a current blur dataset with real and synthetic over- and underexposure images and evaluated its performance on real-world images. We added blur detection to our IPS to further identify these failure types in PI.
- **Next Best View**: Next Best View research looks to find the best angles to get the most unknown information for perception tasks. Very few prior methods account for light direction. Likewise, they usually do not have knowledge of failure for light directions or angles of view that are not present in the training dataset. We created a dataset for PI where different light directions and angles of view are observed and we predicted failure across them. We also analyzed whether one light direction and one angle-of-view from our training data can predict the failure of an unseen light direction and angle-of-view.
- **Introspective Perception**: IPSs have achieved better risk mitigation compared to vision tasks without IPS. However, most current IPSs do not classify

failures pre-hoc and do not have blur detection involved in their networks. We added a blur detection and classification network to our IPS and predicted failure from light direction, blur, and angle-of-view before the PI task executes.

- **Physics Intuition Network**: The goal for the PI built by Ahuja et al. [3] was to determine if a block in a scene could be removed without disrupting the other blocks in the scene. Because of the real-world gap in accuracy that they saw when training with synthetic images, we saw room for improvement in the task by incorporating our IPS. We focused on the Clutter scenario trained using MuJoCo noisy images and used the pre-trained AlexNet network weights for gathering ground truth for our IPS evaluation and light direction and angle-of-view failure prediction network.

Chapter 3

Methodology

We split this chapter into three main sections: **Blur Detection & Classification**, in which we described our blur network and our dataset creation methods, **Light Direction & Angle-of-View Failure Prediction**, in which we delved into how we gathered our data and ground truth and developed our failure prediction network for light direction and angle-of-view, and **Introspective Perception Network (IPS)**, in which we discussed the final IPS network design, our testing dataset and our metrics used for evaluation.

3.1 Blur Detection & Classification

In this section, we will talk about the methods for creating our blur detection and classification that is used in our IPS. We used a modified version Kim et al.'s network [19] (discussed in Section 3.1.2) as our baseline detector and classifier for our IPS. We defined blur categories to include motion, focus, and over- and underexposure and created real and synthetic images for training and testing the blur detector and classifier (Section 3.1.1). In Section 3.1.3, we discussed our defined different dataset training and testing scenarios for evaluating our synthetic creation methods, over- and underexposure classification using a current motion and focus blur classifier, and real-world performance.

3.1.1 Dataset Creation

Rather than collect entirely new data containing all four categories of blur, we instead synthetically applied blur to images. Synthetic dataset creation has become a growing area of research to address the problem of insufficient data for training neural networks [17, 37]. Taking this approach allowed us to leverage image datasets that are already available, and it can be applied to domains and image datasets other than those explored in this paper. In this section, we introduce a method for synthetically applying over- and/or underexposed blur to images, resulting in a dataset that contains multiple forms of blur. We discuss our methods for applying real and creating synthetic motion and focus blur (subsection 3.1.1), applying real and creating synthetic over- and underexposure blur (subsection 3.1.1), gathering our own dataset for exposure evaluation (subsection 3.1.1), and finally gathering a dataset for evaluating our IPS (subsection 3.1.1).

Motion and Focus Blur Images

Figure 3.1: Motion Blur Testing Image Example

Motion blur (Figure 3.1) can be synthetically created by applying a convolution on the image with a directional kernel and then alpha blending it [31, 41] into the image. Alpha blending allows for the motion in the image to be integrated in to create the effect of real blur. We used this method as well as real motion blur images to create training image segments overlaid on the out-of-focus images. Many motion blur images in CUHK have motion blur located in the background or include large

motion objects. These are hard to overlay with a training image that has many other types of blur present. Therefore, we decided to overlay those training images with synthetic motion blur objects from the motion images's salient objects found through using Qin et al.'s saliency detection [33]. For our IPS blur dataset, training images were overlayed with real motion blur salient objects found through using Qin et al.'s salient detection [33]. For test images, we did not create motion blur because the CUHK and our recorded test set already contain images with pre-segmented motion blur that does not need to be overlaid.

Figure 3.2: Focus blur testing image example

Focus blur (Figure 3.2) was not created synthetically because the CUHK and our recorded training and testing datasets already contain out-of-focus images. CUHK and our recorded out-of-focus images were used as the base images for the training dataset with overlaying of motion and over- and underexposure blur objects that were saliency-detected using Qin et al.'s salient object detection network [33]. The PHOs dataset was not used to create synthetic motion or focus images.

Over/Underexposure Blur Images

For over- and underexposure dataset creation, we wanted to evaluate over- and underexposure as a blur classification type and our synthetic creation of these blur types. We also wanted to evaluate our real collected images for testing in a robot pick-and-place task scene. Three categories of dataset creation for over- and underexposure were defined as: real over- and underexposure images taken at different ranges from

PHOs (subsection 3.1.1); synthetic over- and underexposure images created at different ranges using well-exposed PHOs (subsection 3.1.1); and finally, our recorded over- and underexposure dataset taken at different exposures for real-world blur classification evaluation on pick-and-place task scenes (subsection 3.1.1).

Figure 3.3: Real overexposed (top) and underexposed (bottom) image ranges used for training and testing (+-2 to +-4)

Real Figure 3.3 (top) shows the range of overexposure represented in the PHOS dataset. We used the dataset that contains 15 unique scenes recorded in uniform illumination, each captured at 9 exposure levels between -4 and +4 stops. Of these, we kept images with a f-stop $\leq$ -2 as underexposed and those with a f-stop $\geq$ 2 as overexposed, resulting in 45 labeled images for each exposure (in total 90). The challenge in defining these ranges is that if the user viewing a single image could not tell that it was over- or underexposed, the network would also find it difficult to classify. Blur characteristics for over- and underexposed images are usually found in

images that a human user would not confuse. In total, 72 real over- and underexposed images were randomly chosen, salient objects were extracted using Qin et al.'s saliency detection [33], and then those objects were alpha-blended onto the base image for training. For testing, 18 random real over- and under-exposed images were used.

Figure 3.4: Synthetic overexposed (top) and underexposed (bottom) image ranges

Synthetic Using the standard image contrast and brightness change equation:

$$output = image * \alpha + \beta \tag{1}$$

where α is the contrast term and β is the brightness term, we can create over- and underexposed images by performing standard image scaling with high and low values corresponding to over- and underexposure. Using $\alpha > 1$ and $\beta > 0$ creates brighter images and $\alpha < 1$ and $\beta < 0$ creates darker images. All bright images do not necessarily contain overexposure blur and vice versa. To account for this, we used high α and β values that correlate to overexposure. From extensive testing, these

ranges were found to be $1.5 \leq \alpha \leq 2.6$ and $0 < \beta < 100$. For underexposure, our extensive testing resulted in the value ranges $0.01 \leq \alpha \leq 0.6$ and $-100 < \beta < 0$. Pixel values higher than 255 or lower than 0 were clipped to [0, 255]. Figure 3.4 shows the range of synthetically created over- and underexposed images that we look to detect as blurry. For creating synthetic over- and underexposure images, 12 randomly chosen, correctly exposed images (no over- or underexposure) from the PHOS dataset were used. Each image chosen was applied random alpha and beta values in the ranges correlating to over- and underexposure and then salient object extracted using Qin et al.'s salient detection [33]. We then alpha-blended the objects into the base image.

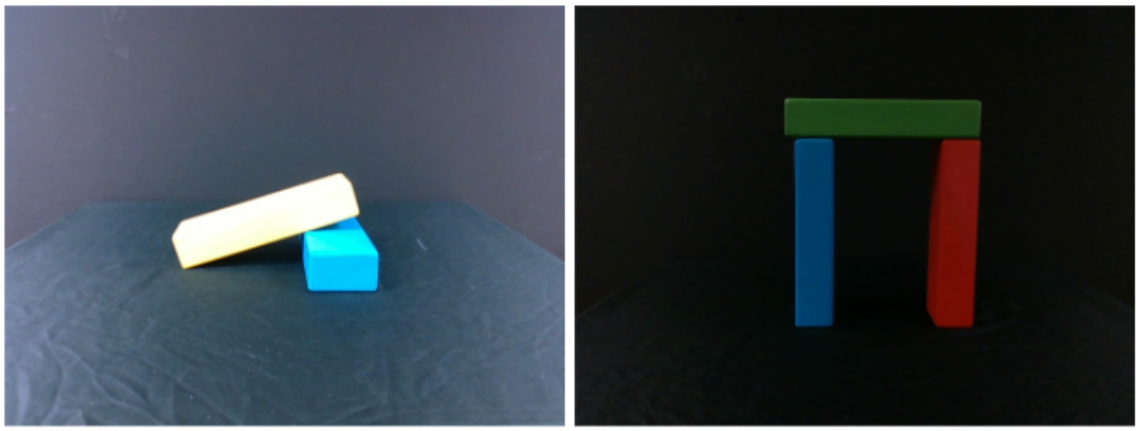


Figure 3.5: Example overexposed (left) and underexposed (right) images from our recorded dataset

Our Recorded Exposure Figure 3.5 displays two example images from our taken image exposure dataset. To experiment with real robot pick-and-place task environmental settings, we collected images using a standard Realsense RGBD D435 camera attached to a Fetch robot arm. We took exposure ranges in 10 different scenes. We used auto-exposure and manual exposure settings to define the well-balanced exposure level (no over- or underexposure present) for each unique scene. Ten overexposed and ten underexposed images were gathered at ranges +- to the original correct exposure image. The min and max exposures were from 1.0 to 95.0 microseconds according to the Realsense ranges. For ground truth labels, the background of the exposed objects in the scene was labeled as out-of-focus blur because the Realsense D435 has a fixed

focal length. 200 images in total were gathered. We used 160 images for training and 40 for testing.

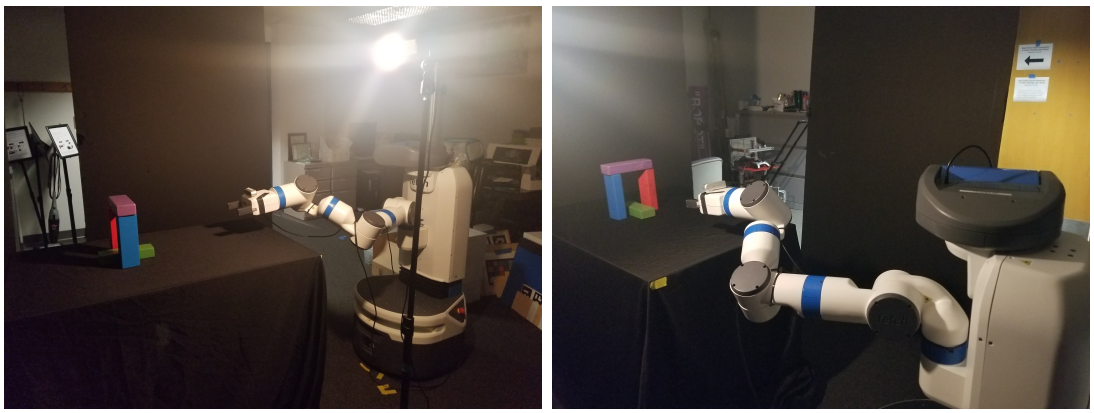


Figure 3.6: Our IPS blur dataset gathering setup

Our IPS-Recorded Blur Dataset

We decided to record our own separate blur dataset and used that for our IPS blur detection because it increased the blur classification accuracy. Figure 3.6 shows some images of our dataset gathering. We collected images of Ahuja et al. [3]'s clutter scenario using a standard Realsense RGBD D435 camera attached to a Fetch robot arm. We took images of 45 different scenes with motion, focus, and over- and underexposure blur, including 10 unique scenes with 1, 2, 3, 4, and 5 blocks present in multiple different clutter arrangements. We used manual exposure settings with an additional light to define the well-balanced exposure level (no over- or underexposure present) for each unique scene. We gathered one well-exposed no-blur image, one motion-blur image (by moving the camera on the fetch), one focus-blur image (moving the fetch arm farther away), one overexposed image (turning the light on all the way), and one underexposed image (turning off all the lights) for each unique scene. For ground truth segmentation, the background of the exposed objects in the scene was labeled as out-of-focus blur because the background was not well-focused. A total of 450 images were gathered. We used 400 images for training (overlaid creation and synthetic creation as described in subsections 3.1.1, 3.1.1) and 50 for testing (all real images split across all blur types equally). Figure 3.7 displays some testing image examples from our IPS blur dataset.

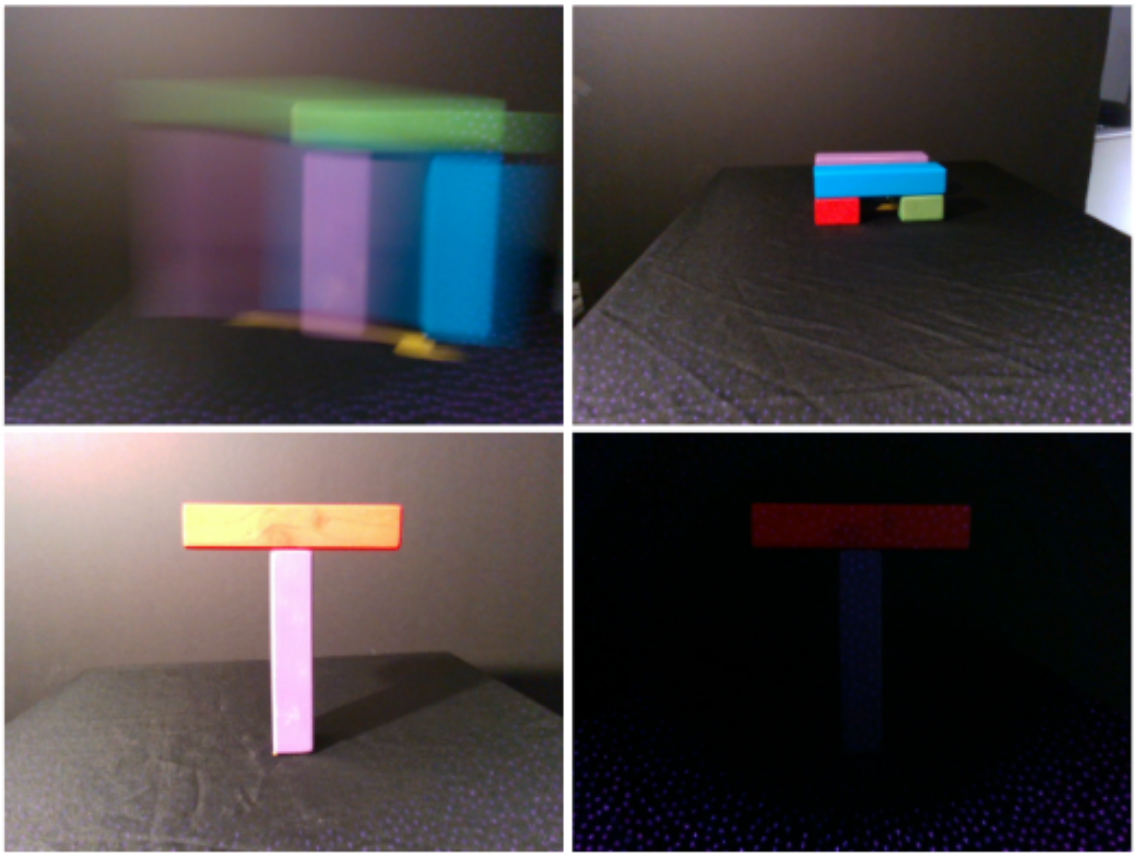


Figure 3.7: Our IPS blur dataset example testing images: motion (top left), out-of-focus (top right), overexposure (bottom left), and underexposure (bottom right)

3.1.2 Network Architecture

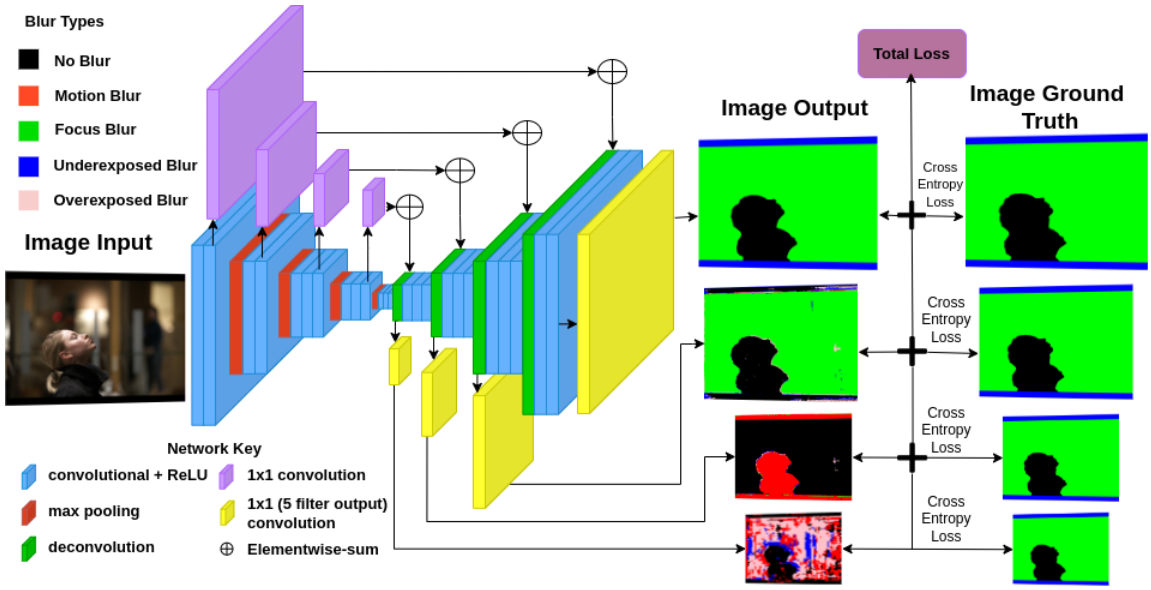


Figure 3.8: Network block diagram: We modified the Kim et al. network [19] and changed the yellow layers to be 5-filter output for 5 classification labels.

We used the neural network architecture of Kim et al. (Figure 3.8). This model consists of an encoder VGG-19 without its last fully connected layers connected to a U-Net decoder with a multilayered output to define the model loss. The input and output are the same image size with 3-channel input and 5-channel output for each blur type (none, motion, focus, over- and underexposure). The VGG-19 [40] network consists of 16 convolutional layers and 4 max-pool layers. The last max-pool and fully connected layers were removed to connect to the decoder, a U-Net architecture with four deconvolution layers and 15 convolution layers. Kim et al. [19] added skip connections with a convolutional layer that added features from the encoder to the decoder at each layer and an element-wise summation at the skip connections instead of a concatenation.

We used the same multichannel, cross-entropy loss function as Kim et al. [19] It combines the softmax output at each of the 4 skip connection-level outputs. Each level performed a cross-entropy loss comparing the softmax output of the network with the ground truth map. That map was scaled to the output size of each layer output. The loss at each layer was then summed to create the final loss value, which

was minimized. Thus, the network learned the high-level features in each layer. We adopted this network because it learned the blur features as well as over- and underexposure features. Multi-convolutional layers can identify distinct differences like the difference between dark parts of an image and underexposure blur.

3.1.3 Experiment

We recorded the following performance metrics:

- Relative performance of a network trained over either (a) synthetically over- and underexposed objects, or (b) real over- and underexposed objects;
- Relative performance of a network trained over either (a) a combination of real and synthetic over- and underexposed objects, or (b) only real over- and underexposed objects;
- Relative performance of a network using the baseline weights ([19]) versus a network trained over a combination of synthetic and real over- and underexposed objects; and
- Relative performance of a network trained over either (a) a combination of synthetic and real over- and underexposed objects from standard dataset scenes, or (b) a combination of synthetic and real over- and underexposed objects from a robotic pick-and-place task scene, where both are evaluated in the context of a robotic pick-and-place task.

Training Conditions

To record these measures, we created 7 training conditions and 3 test conditions. For datasets that used CUHK and PHOS datasets, we used the same training/testing split that was used by Kim et al. [19] and used the uniform illumination images from the PHOS dataset. For our exposure and IPS blur datasets, we used the same randomly picked training/testing split across all condition variation. These training conditions were as follows:

- PHOS + CUHK, Real Blur (PC-R): 564 focus blur base images overlaid with synthetic motion blur objects from 40 motion images from the CUHK dataset, and real over- and underexposed objects from 36 over- and 36 underexposed

images in the PHOS dataset;

- PHOS + CUHK, Synthetic Blur (PC-S): 564 focus blur base images overlaid with synthetic motion blur objects from 40 motion images from the CUHK dataset, and synthetic over- and underexposed blur objects created from 12 well-exposed images in the PHOS dataset;
- PHOS + CUHK, Real and Synthetic Blur (PC-RS): 564 focus blur base images overlaid with synthetic motion blur objects from 40 motion images from the CUHK dataset, and a mix of real and synthetic created over- and underexposed objects that used 36 over- and 36 underexposed images and 12 well-exposed images in the PHOS dataset;
- Our Exposure + PHOS + CUHK (OE-RS): 564 focus blur base images overlaid with synthetic motion blur objects from 40 motion images from the CUHK dataset, and a mix of real and synthetic over- and underexposed objects using 80 over- and 80 underexposed images from our recorded exposure dataset and 12 well-exposed images from the PHOS dataset;
- Our IPS, Real Blur (O-R): 80 focus blur base images overlaid with real motion blur objects from 80 motion images from our IPS blur dataset, 80 well-exposed objects from those type of images from our IPS blur dataset, and real over- and underexposed objects from a combined 160 over- and underexposed images in our IPS blur dataset;
- Our IPS, Synthetic Blur (O-S): Focus blur base images overlaid with real motion blur objects from 80 motion images from our IPS blur dataset, 80 well-exposed objects from those type of images from our IPS blur dataset, and synthetic over- and underexposed blur objects created from 80 well-exposed images in our IPS blur dataset; and
- Our IPS, Real and Synthetic Blur (O-RS): Focus blur base images overlaid with real motion blur objects from 80 motion images from our IPS blur dataset, 80 well-exposed objects from those type of images from our IPS blur dataset, and a mix of real and synthetic created over- and underexposed objects that used a combined 160 real over- and underexposed images and 80 well-exposed images from our IPS blur dataset.

We used the same blur object overlaying method for creating our training images as Kim et al. [19]. We synthetically overlaid and created training images by converting the images from non-linear to linear space using a standard gamma correction value of 2.2. This value is used by most camera manufacturers [47], so we used it for the PC-R, PC-S & PC-RS dataset creations. We used a gamma correction value of 1.5 for our other training dataset creation because our recorded images used a gamma encoding of 1.5. They were gamma decoded back after the blur creation was completed. All four types of blur were usually present in the images. Because of placement errors where images would be placed on top of each other randomly, not all blur types would be present in one image for each training image. This overlaying and creation method was repeated for each image five times, resulting in 2,820 images for the PC-R, PC-S, PC-RS, & OE-RS training datasets. For the training datasets O-R, O-S & O-RS, we used the same overlaying method but randomly pick base focus images to overlay from our training dataset and resulted in 400 training images.

Testing Conditions

We evaluated our network over three testing conditions:

- PHOS + CUHK, Real Blur (TPC-R): 100 focus and 100 motion blur images from the CUHK dataset, combined with 18 real over- and underexposed images from the PHOS dataset;
- PHOS + Our Exposure, Real Blur (TPOE-R): Real over- and underexposed images from PHOS (18 images) and our recorded (40 images) datasets;
- Our IPS, Real Blur (TO-R): 10 normal, 10 focus, 10 motion, 10 overexposure and 10 underexposure blur images from our IPS dataset.

While training images contained multiple overlaid types of blur, testing images were created to contain real-world blur segments (with no overlay) in each image. Figure 3.9 displays some example testing images from the TPC-R dataset.

Network Setup

We implemented the network in Tensorflow and trained it on 1 RTX 2080 Ti GPU. To prevent over-fitting, we used the previous weights from Kim et al. [19] for all of the layers except for the output layers. We retrained the last layer to output 5

Figure 3.9: Testing image examples. Clockwise from top left: Overexposed, underexposed, focus, and motion blur.

channels (as opposed to the original 3 channel output from Kim et al.) to account for the addition of over- and underexposure blur. We used the same data augmentation method as Kim et al. [19]. For the training conditions PC-R, PC-S, PC-RS, & OE-RS, we trained the decoder for 2000 epochs training rate of 1e-6 and a batch size of 10. For training conditions O-R, O-S & O-RS, we trained the decoder for 3000 epochs training rate of 1e-6, and a batch size of 10. We used the previous training weights for the encoder, as it resulted in good performance and less training time relative to retraining both the encoder and decoder.

3.2 Light Direction & Angle-of-View Failure Prediction

We wanted to combine two points of perception failure into one network: Light Direction and Angle-of-View. We used a Modified AlexNet that Ahuja et al. [3] used for their PI system that we discussed in subsection 3.2.3. We created a training, validation and testing dataset where our images are taken at three different light directions and angles of view. The dataset was taken of variations of clutter scenes that Ahuja et al. [3]'s network used for their predictions. Subsection 3.2.1 further details this dataset's creation and image processing. We collected this dataset to predict and analyze failure prediction accuracy across different light directions and angles of view. We did this by splitting this dataset into different sets of training, validation and testing images in which we varied the light directions and angles of view that are included (discussed in subsection 3.2.2). Thus, we were able to evaluate validation and testing accuracy for light directions or angles of view that were not available in the training dataset. This process was for exploring the question of whether we can predict failure for validation and testing light directions angles of view that are not present in the training dataset.

3.2.1 Dataset Generation

We detailed how we collected our dataset of different light directions and angles of view in subsection 3.2.1. We then discussed the image processing and formatting

(subsection 3.2.1) that made the data ready for running through the PI network to compute the ground truth for our failure prediction network (subsection 3.2.1).

Our Recorded Dataset Generation

The goal for our dataset generation was to represent scenarios in which the PI network can fail with the ability to analyze the failures across different angles of view and light directions. Using Ahuja et al. [3]'s real-world data collection libraries, we modified the code to gather a single scene at 3 different light directions and at 3 different angles of view. In total 9 images are created for one scene. We took images of 20 unique clutter scenes. We used the Fetch robot's arm to move the RGBD D435 Realsense camera to each of the 3 different angles of view. The 3 angles of view that were recorded were center (0 degrees), 30 degrees to the right and 30 degrees to the left. The light directions that were recorded were 90 degrees right, 90 degrees left and $\sim$10 degrees forward to the left. These angles of view and light directions that were gathered are not a representation of all possible angles of view and light directions that are available. In the results sections 4.2.3 & 4.2.4, we explored the possibility that we did not need to see all angles to predict the performance of the network at those unseen angles even if we only have a few seen angles. We gathered the following information in our dataset collection process:

- RGB Image,
- Depth Image,
- Angle of View, and
- Light Direction.

Image Processing

The PI network relied on masking when identifying the object of interest when predicting no-disruptive removal from the scene. We performed color correction using a color checker on standalone images that corresponded to the different light directions and angles of view. We then performed object masking with a combination of custom color thresholding and salient object detection through Qin et al. [33]. This color threshold object-masking implementation was not perfect, as shown in Figure

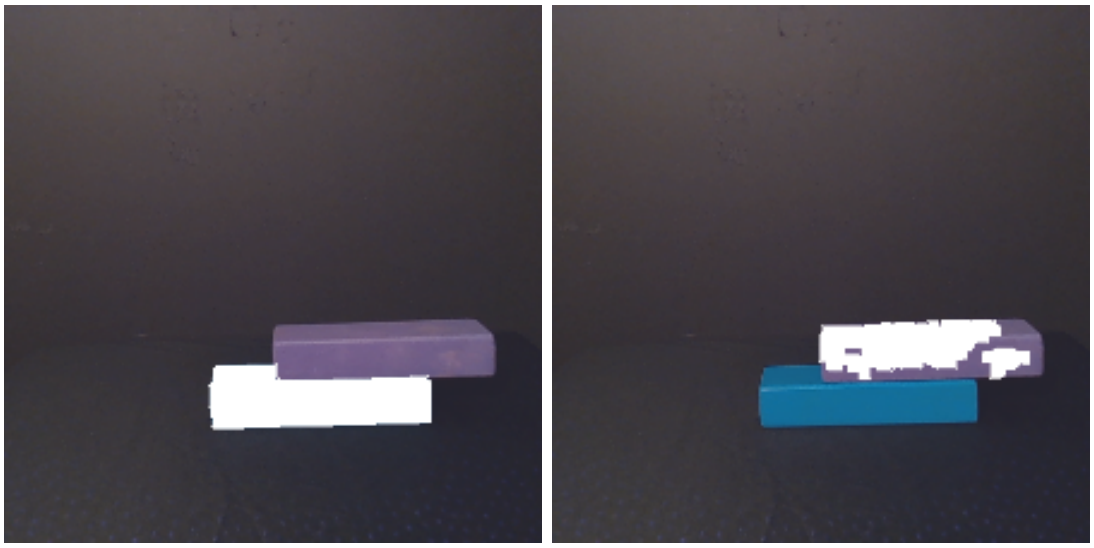


Figure 3.10: An example of a clutter scene from our dataset generation with masking the object of interest

3.10. In general, color thresholding performs very badly in real-world scenarios due to changes in lighting and environment. Color checkers cannot always be used to normalize the color changes that occur in real-world scenarios. This form of failure was something that we wanted to include in our dataset and is one of the possible reasons that this PI network can fail. After masking was computed, all of the images were cropped to size 224 by 224. The depth image was smoothed using bilateral filtering implementation from the NYU Depth MATLAB toolbox [28].

Ground Truth Data Gathering

To predict failure we need to know which data points fail for the PI network. To this end, we formatted our images and ground truth labels to be processed through the PI network. We used the pre-trained Clutter MuJoCo noisy weights from the PI AlexNet network to gather the predictions. This gave us the perception task accuracy of the PI network. We then compared the output predictions to the ground truth labels and determined if they were correct or incorrect (1 or 0). This correct or incorrect measurement becomes the ground truth data for our failure prediction network.

3.2.2 Experiment

We recorded the following performance metrics for our failure prediction network:

- Relative performance of a network trained, validated, and tested over scenes where all three light directions and angles of view are present; and
- Relative performance of a network trained such that (a) only two light directions are present in training and one light direction not in training is present in validation and testing (b) only two angles of view are present in training and one angle-of-view not in training is present in validation and testing.

To evaluate our performance metrics, we split the dataset gathered for our prediction network into three different training and testing condition pairs:

- Scenario-based Split - (SS): Training contains images of unique scenarios where all light directions and angles of view were included in the dataset. Testing and evaluation contain images of unique unseen scenarios where all light directions and angles of view were included in the dataset. Training, testing, and validation are split by the unique scenes.
- Light Direction-based Split - (LS): Training contains images of all unique scenarios where only *two* light directions were included in the dataset. Testing and evaluation contain images of scenarios where only *one* light direction was included in the dataset that was not included in the training dataset. Training, testing, and validation are split by the different light directions. Each training, testing, and validation split included all three angles of view.
- Angle of View-based Split - (AS): Training contains images of all unique scenarios where only *two* angles of view were included in the dataset. Testing and evaluation contain images of scenarios where only *one* angle-of-view was present in the dataset that was not in the training dataset. Training, testing, and validation are split by the different angles of view. Each training, testing, and validation split included all three light directions.

All dataset splits were strategic in terms of the number of blocks that are present in the scene. For example, if there were two blocks in a scene, then a full scenario consists of 2*3*3 (2 blocks, 3 light directions, and 3 angles of view), which is equal to 18 separate input images with corresponding ground truth labels. This number

changes for the number of blocks that were present in the scene.

For our dataset collection, we created 20 scenes: 4 - two blocks, 7 - three blocks, 7 - four blocks, and 2 - five blocks. The SS split has training 80%, validation 10% and testing 10%. For 20 scenes, this is 16 for training, 2 for testing, and 2 for validation. The splits were found by determining the number of blocks in each scenario and calculating the total number of data points for the main dataset that maximized the training dataset split. For the SS split, this was found to be 603 = (4*3*3*2) + (7*3*3*3) + (7*3*3*4) + (2*3*3*5). 603*.1 ≈ 60.3. The numbers for validation and testing data points using one three-block scenario and one four-block scenario are each 63. This is ≈ 10.4% of the full dataset. The number of training images was 477, which is ≈ 79.1% of the full dataset. This maximized the training dataset without changing the percentage of images in the training, testing and validation split. The unique training variations for the SS split are shown in table B.1 in Appendix B.

For the LS & AS split, the number of data points in training and testing were dependent not only on the number of blocks in each scene in the dataset, but on the number of light directions and angles of view. For this split, we got 510 total data points, 402 for training and 54 each for testing and validation. This number differs from the SS split because we wanted to maximize the number of training data points while keeping the splits at 10%. For testing, we used all four of the two-block scenes and all two of the five-block scenes, which equaled 54 data points. For validation, we used six out of seven of the three-block scenarios to equal 54 data points. The unique training variations for the LS split are shown in table B.2. Likewise, the unique runs for the AS split are shown in table B.3. Both in Appendix B. Figures 3.11 & 3.12 show examples of the images taken at different angles of view and light directions.

3.2.3 Network

A visualization of the network that we used from Ahuja et al. [3] is shown in Figure 3.13. They used the baseline AlexNet feature extractor [21] with a 4-channel input, not including depth. The classifier has a dropout and linear layer followed by a sigmoid activation function. This network was easy to implement and was easily edited to accommodate our needs for implementing our prediction network. Other IPS networks used similar network architectures [7]. An ResNet18 was also tested for

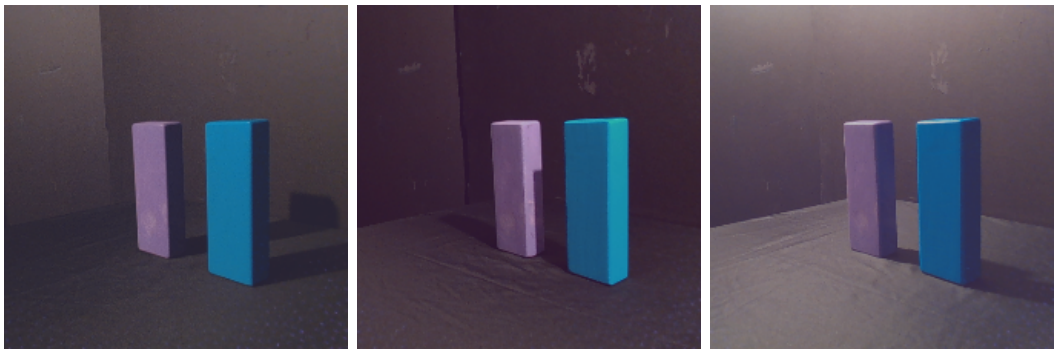


Figure 3.11: Example from our Dataset from Different Light Directions

Figure 3.12: Example from Our Dataset at Different Angles of View

the possibility of raising the accuracy of our failure prediction network. ResNet18 was modified to have a 4-channel input with a 1-channel sigmoid output.

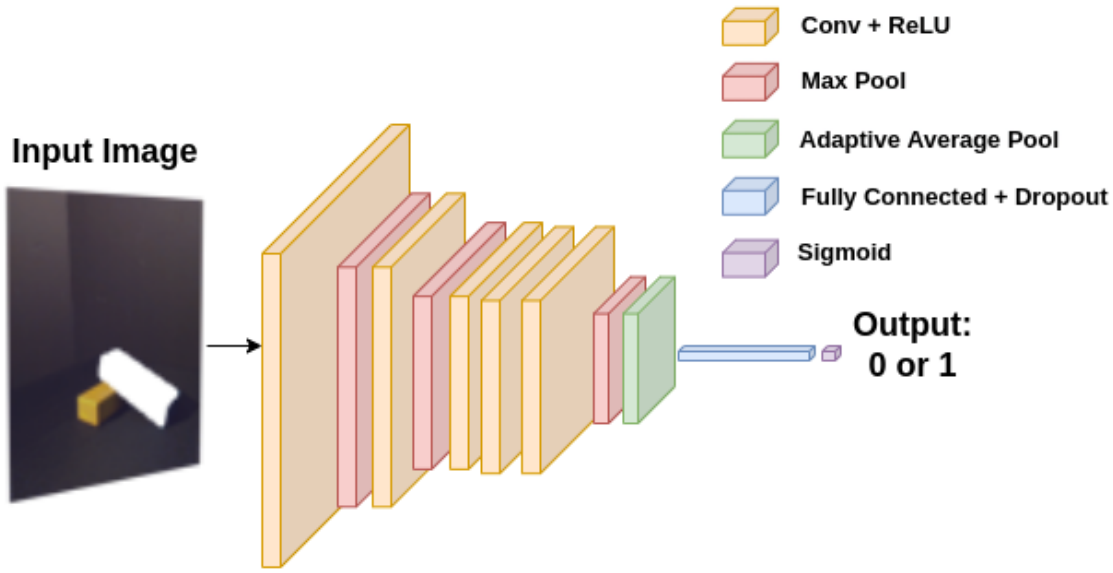


Figure 3.13: Modified AlexNet from Ahuja et al. [3]

We wanted to test two different input variations: an (a) RGB image with a mask around the object of interest and (b) RGBD image with a mask around the object of interest. Our output was a single probability value where we used either cross entropy loss or mean squared error loss to minimize. We trained this network using different hyper parameters (as defined in Appendix B.1) and found the best combination for all three dataset splits.

3.3 Introspective Perception Network (IPS)

We combined the two previous discussed subsystems, Blur Detection & Classification (section 3.1) and Light Direction & Angle-of-View Failure Prediction (section 3.2), into one IPS network. We discussed the combined model in subsection 3.3.1. We took each of the PI's object input and output pairs and predicted failure based on the blur output and the angle-of-view and light direction failure prediction network. We created a set of tests for performance evaluation of our introspective perception network (discussed in subsection 3.3.2). We also discussed the evaluation metrics

that we used in subsection 3.3.3.

3.3.1 Model Overview

From the input image, we performed Qin et al.'s saliency detection [33] to localize on the area of interest for our blur detection and classification. We did not care about the possible out-of-focus or non-blur background of the image for our blur detection and classification. We only were concerned about the objects of interest. Salient detection allowed us to focus only on the object of interest and not on the whole image background which could cause failure in our network. We then performed the blur detection and classification using the network (section 3.1.2) with the O-R training dataset weights (subsection 3.1.3). This was followed by color thresholding and object split (subsection 3.2.1) for running though our light direction and angle-of-view failure prediction network (section 3.2.3). We used weights found from our results section that are chosen based on our testing conditions (subsection 3.3.2). We outputted a prediction of either failure (0) or success (1) per each PI masked object input based on the output of these two networks. For blur, we defaulted to 0 failure prediction for all masked object inputs of one image. We used the PI AlexNet network with the pre-trained Clutter MuJoCo noisy weights for our performance evaluation for our IPS. Figure 3.14 shows our IPS model flow.

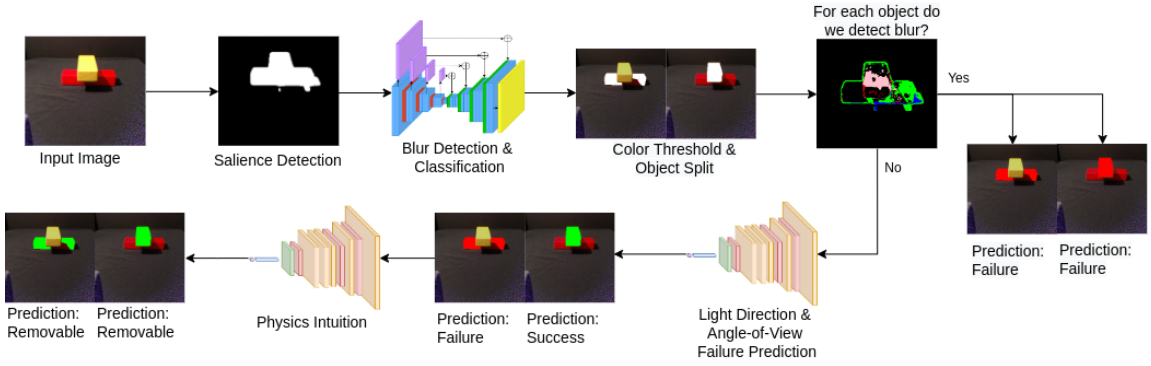


Figure 3.14: Introspective perception failure prediction network workflow diagram

3.3.2 Experiments

We evaluated our IPS using the following metrics:

- Relative performance of our IPS with PI on blur types, motion, over- and underexposure where angle-of-view and light direction is stationary;
- Relative performance of our IPS with PI on angle-of-view and light direction changes where blur is not present; and
- Relative performance of our IPS with PI on angle-of-view and light direction changes where certain light directions and angles of view were not present in the training dataset and where blur is not present.

For the following test scenarios for our IPS, we collected multiple clutter scenes with different numbers of blocks in each scene. We collected images using a standard Realsense RGBD D435 camera attached to a Fetch robot arm. We did not test focus blur failure because the RealSense RGBD camera has a large focal range and focus is not an issue in our environment. We classified focus blur detection as no blur. The following unique test conditions were collected and used for evaluation of each failure condition:

- Motion Blur (MB): 9 motion scenes; (four 2-block, four 3-block, and one 4-block scene, resulting in 21 data points (one 2-block, one 3-block and one 4-block scenes failed to mask all colors that were present)): Training Condition: O-R Blur Dataset & SS Dataset Split using run11 weights;
- Overexposure Blur (OB): 9 overexposure scenes; (four 2-block and five 3-block scenes resulting in 21 data points (one 2-block scene and one 3-block scene failed to mask all colors that were present)): Training Conditions: O-R Blur Dataset & SS Dataset Split using run11 weights;
- Underexposure Blur (UB): 22 underexposure scenes; (three 1-block, ten 2-block, six 3-block, and three 4-block scenes resulting in 22 data points (All block scenes failed to mask all colors that were present. Either 1 block would be masked or no blocks would be masked)): Training Condition: O-R Blur Dataset & SS Dataset Split using run11 weights;
- Angles of View with SS weights (SSA): 11 angles of view scenes; (five 2-block, five 3-block, and one 4-block scenes totaling 27 data points (two 3-block scenes

failed to mask all colors that were present)): Training Condition: O-R Blur Dataset & SS Dataset Split using run11 weights;

- Angles of View with AS weights (ASA): 11 angle-of-view scenes; (five two-block, four 3-block, and two 4-block scenes, resulting in a total of 29 data points (one 4-block scene failed to mask all colors that were present)): Training Condition: O-R Blur Dataset & AS Dataset Split using run11 weights; Testing Condition: Angle-of-view not present in training dataset
- Light Directions with SS weights (SSL): 9 light direction scenes; (four 2-block, four 3-block, and one 4-block scenes, resulting in 22 data points (one 2-block scene and one 3-block scene failed to mask all colors that were present)): Training Condition: O-R Blur Dataset & SS Dataset Split using run11 weights;
- Light Directions with LS weights (LSL): 19 light direction scenes; (two 1-block, thirteen 2-block, and four 3-block scenes, resulting in 37 data points (two 2-block scenes and one 3-block scene failed to mask all colors that were present)): Training Condition: O-R Blur Dataset & LS Dataset Split using run11 weights; Testing Condition: Light Direction not present in training dataset

As described, the collected data had poor masking and data points that had to be removed because of faulty data (e.g., being underexposed images that were not correctly captured as underexposed), leading to unequal datasets for each scenario. To account for this, test scenarios with more than 21 data points were shuffled 50 times. For each random shuffle, the metrics were evaluated at the 21 picked data points and added to an array. This metrics array was then averaged and the final reported metrics were the mean of this array. For the combined scenario evaluation, 21*7 = 147 data points were used where scenarios with more than 21 data points where randomly shuffled 50 times. Using this method, all data points were considered without having unbalanced metric evaluation for each test scenario. For data point filtering and removal, some objects failed because (a) lights failed to turn off before data was gathered for underexposure images and (b) some objects that were masked (specifically purple and red) were combined in which each had a separate ground truth PI label that would have indicated failure and success for either prediction. We removed these data points because they caused confusion and left this issue as a limitation to be considered in future work, as discussed in subsections 5.3.2 and 5.3.2.

All light direction and angle-of-view test scenarios had no blur images attached. Failure detection due to blur was considered failure for all objects in the image.

3.3.3 Evaluation Metrics

We used four metrics to evaluate our failure prediction network: (a) **Accuracy, Precision & Recall of Physics Intuition (APR)**, (b) **Accuracy, Precision & Recall of Prediction of Failure (APRF)**, (c) **Accuracy of Blur Detection and Classification (ABDC)**, and (d) **Risk-Averse Metric (RAM)**.

For the Accuracy, Precision & Recall evaluation, we computed these values for the PI network with (a) all failure conditions combined and (b) each separate failure condition. We tested with and without our IPS connected to the PI network and with and without blur detection connected to our IPS. This allowed us to see the change in accuracy across the focused failures and examine how the addition of blur detection affects the metrics for the PI network. When our IPS was connected to the PI network, a balanced accuracy calculation was computed only on the test points that were marked as successful by our IPS or else not counted. For the PI network not connected to our IPS, all data points were used for accuracy calculation as if there was no IPS. The percentage of data points discarded by our system was also evaluated. If an image was classified as blurred, all of the objects in the image were defaulted to failures by our IPS. A weighted precision and recall was calculated where imbalance was taken into consideration. A balanced accuracy score was also used for label imbalance consideration.

For Prediction of Failure (APRF), we assessed the accuracy, precision and recall for our prediction network's determinations of failure and success. A correct prediction (whether the prediction was fail or succeed) that matched the network's results was computed as a 1. An incorrect prediction (whether the prediction was fail or succeed) that did not match the network's results was computed as a 0. This metric was recorded for the purpose of our accuracy in predicting failure or success for a perception task. We assessed this metric (a) across all the testing conditions and (b) the individual testing conditions for performance comparison. Weighted precision and recall scores was calculated where imbalance was taken into consideration. The balanced accuracy score for our IPS was also used for label imbalance consideration.

For our Blur Detection and Classification metrics (ABDC), we used the same evaluation as used in the Blur results section 4.1. The confusion matrix evaluated our accuracy for final blur label classification that was in the input test image. Because focus blur was not considered as a test condition, it was labeled as no blur in our evaluation. We looked to compare if our blur network (separate from the IPS network) was accurate at predicting blur in our PI real-world scene.

For Risk-Averse Metric (RAM), we based our evaluation off of Zhang et al. [51]'s method in which they evaluated their system's risk trade-off. We tested our IPS trade-off value for the cost of making an incorrect prediction to the cost of letting a incorrect prediction through. Meaning, if we made a prediction of success and it was successful, then 1 was assigned. If we made a prediction of failure and we did not allow it go through the PI network, then we assigned this a 0. Finally if we predicted success and it was a failure, then we assigned a failure risk value between [0,-2]. We assessed this by plotting possible failure risk values between [0,-2] by -0.1 increments. This metric allowed us to see how this can be used to prevent risk where risk varies in its detrimental effect to a perception task.

Chapter 4

Results

This chapter is split into three main sections: **Blur Detection & Classification**, where we document the segmentation results for synthetic & real blur and the results from our recorded datasets; **Light Direction & Angle-of-View Failure Prediction**, where we discuss the dataset split results and the analysis results per angles of view and light directions; and **Introspective Perception System (IPS)**, where we discuss the results from running PI with and without the IPS network connected.

4.1 Blur Detection & Classification

Appendix A.1 gives brief descriptions and key names for each of our training and test conditions. In this section, we describe the results for real vs. synthetic over- and underexposure blur data (subsection 4.1.1). We see that real and synthetic results did have similar accuracy. We also discuss the binary blur & no-blur results in subsection 4.1.2, where we showed that the baseline network confuses over- and underexposure as blur and thus our network resulted in fewer false positives. Finally, we discuss the real-world results from both the TPOE-R (subsection 4.1.3) and the TO-R (subsection 4.1.4) testing conditions, examining how our real-world image results correspond to the online blur datasets and how the change in accuracy is observed.

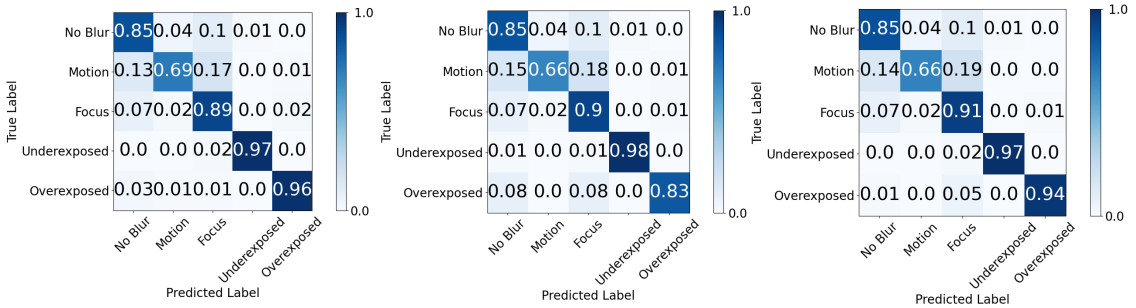


Figure 4.1: Real vs synthetic over- and underexposure blur confusion matrix results using TPC-R: (a) PC-R (b) PC-S (c) PC-RS.

	PC-R	PC-S	PC-RS
Mean Accuracy	86.01	86.08	86.33

Table 4.1: Mean accuracy across TPC-R test condition

4.1.1 Real vs. Synthetic Over/Underexposure Blur Results

We first compared the performance of training the network over either (a) real, (b) synthetic, or (c) a combination of real and synthetically over- and underexposed images. We recorded these results by applying the PC-R, PC-S, or PC-RS training conditions, respectively, and evaluating them on the TPC-R test condition. We present these results as confusion matrices in Figure 4.1. The PC-R training condition resulted in the best per-class performance for over- and underexposure blur (true positives of 97% & 96%, respectively). However, as shown in table 4.1, the PC-RS training condition resulted in the best mean overall accuracy (86.33%) and comparable performance in classifying over- and underexposure blur, as shown in Figure 4.1c (true positives of 97% and 94%, respectively). Figure 4.2 displays examples of our results from the PC-RS training condition.

4.1.2 Blur-No Blur Results Comparison

Figure 4.3 shows confusion matrices comparing the baseline performance of Kim et al. [19] to that of our PC-RS trained model using the TPC-R test condition. For over- and underexposed pixels, we expected the baseline to classify them as no blur

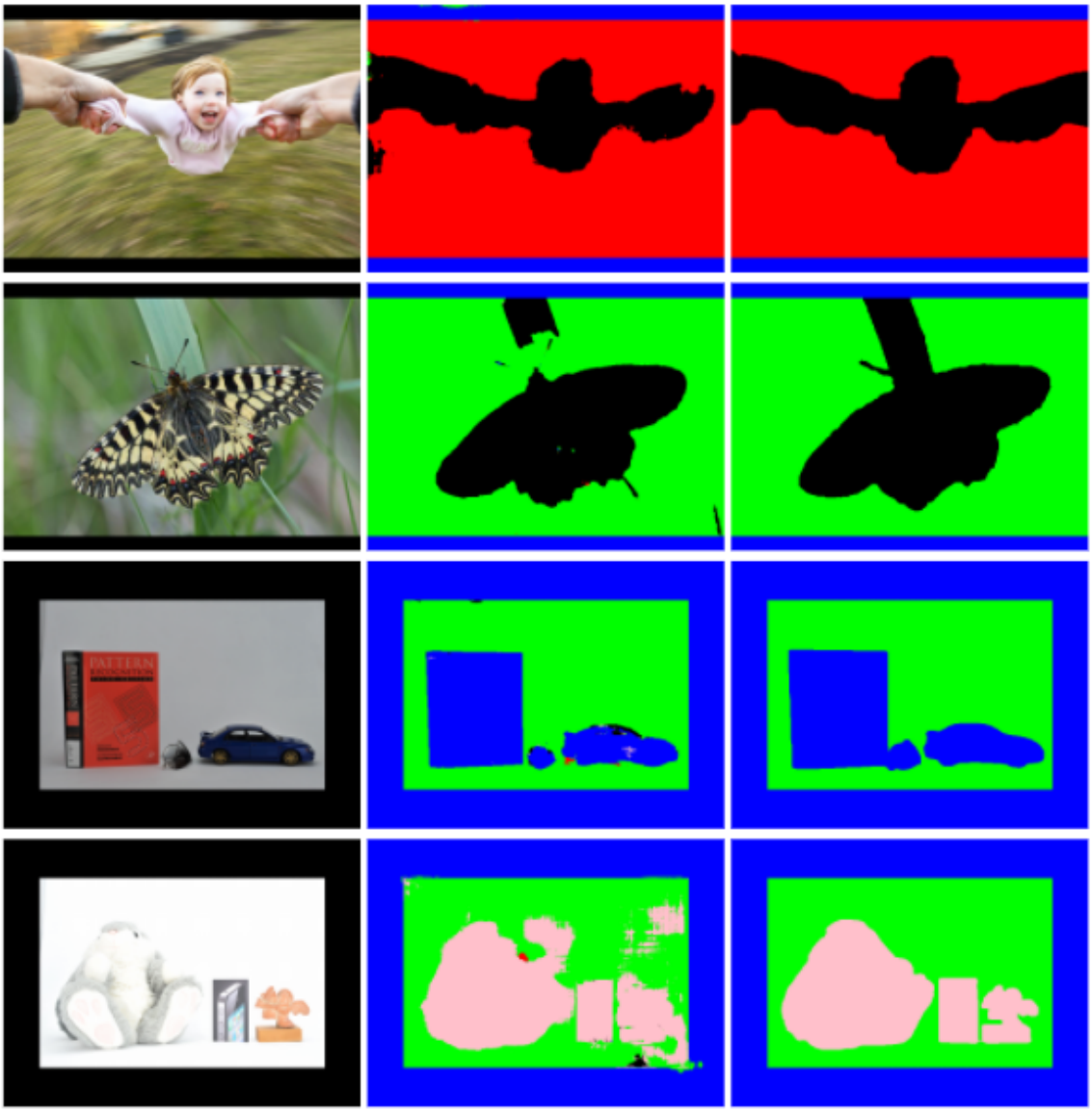
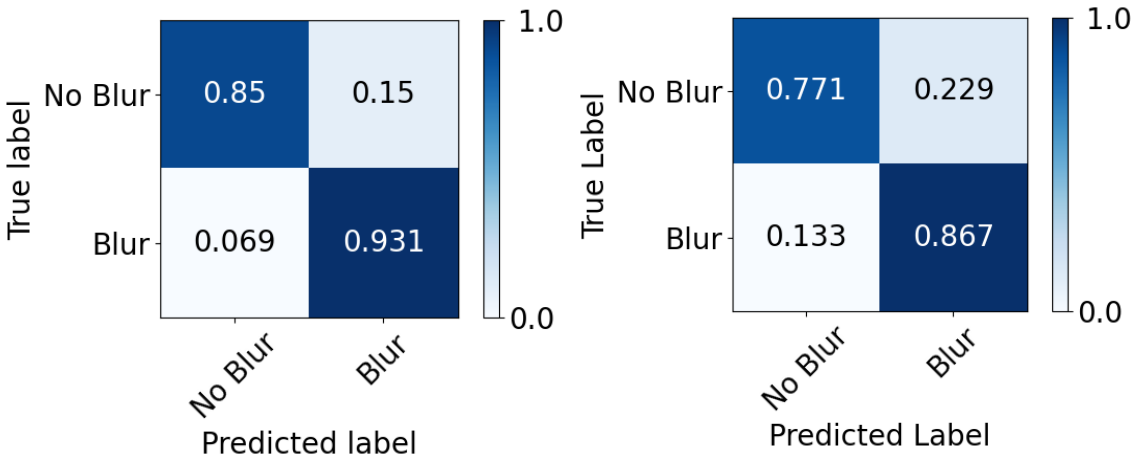
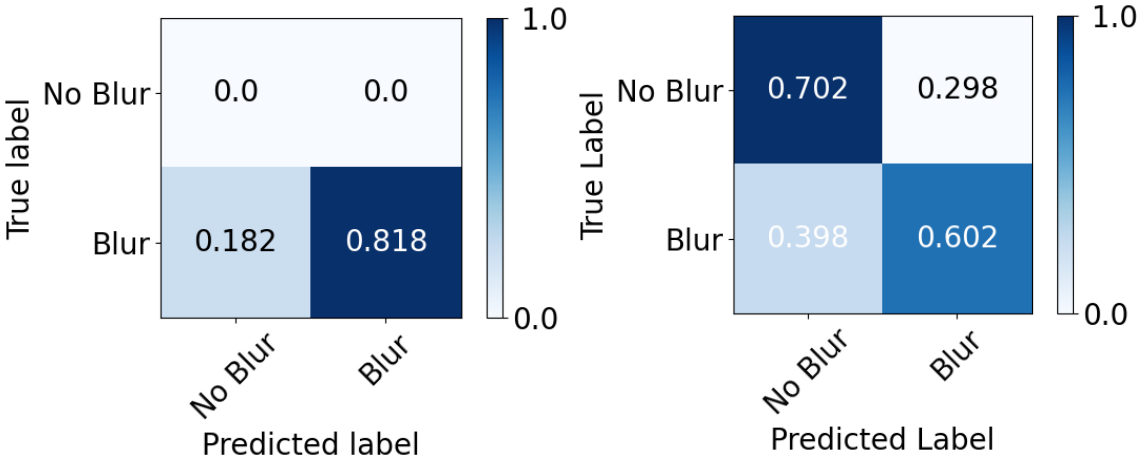


Figure 4.2: Blur Result Examples. Blur types: Black - No Blur, Red - Motion, Green - Focus, Blue - Underexposure, Pink - Overexposure. Left: Input images. Middle: Our results. Right: ground truth. Top to bottom: Examples of motion blur, focus blur, underexposure blur, and overexposure blur.



(a) (b)

Figure 4.3: Binary blur-no blur confusion matrix results using TPC-R: (a) PC-RS, (b) Kim et al. [19].



(a) (b)

Figure 4.4: Binary blur-no blur confusion matrix results using TPOE-R: (a) PC-RS. (d) Kim et al. [19].

because it is not defined by their classification. As we see in Figure 4.3b, the baseline method produced a large false positive classification. Images that were supposed to be classified as containing no blur were actually classified as having blur. Figure 4.4 displays the confusion matrices using the TPOE-R test condition. Again, the baseline method produced a large false positive classification for this over- and underexposure test condition. The baseline method resulted in 29.8% more false positives for this over- and underexposure test condition compared to our model.

4.1.3 TPOE-R Results Comparison

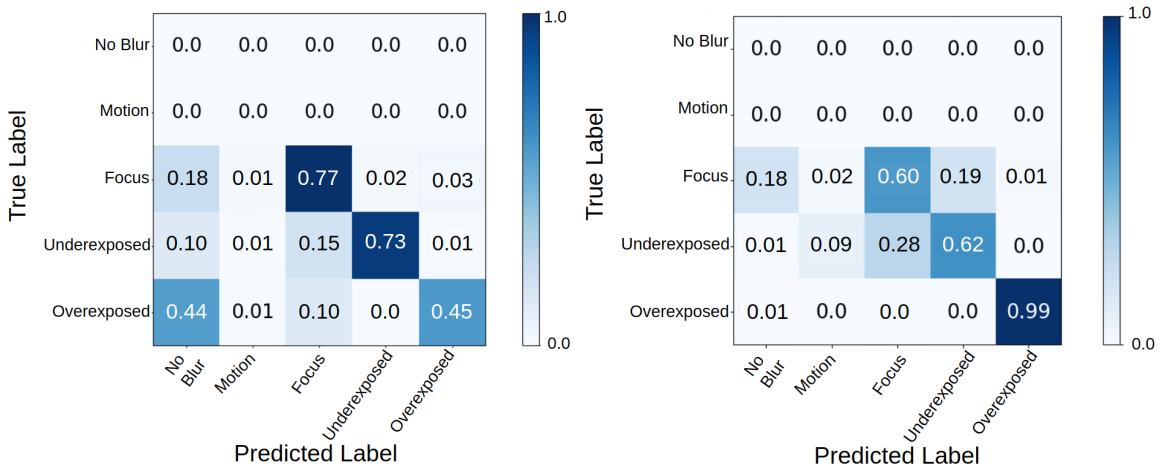


Figure 4.5: Confusion matrix results using TPOE-R: (a) PC-RS, (b) OE-RS.

The baseline method falsely classified over- and underexposed pixels as motion or focus blur 29.8% of the time (shown in Figure 4.4). Applying the PC-RS training condition against the TPOE-R test condition resulted in a 73.4% true positive rate for underexposure, as shown in the confusion matrix in Figure 4.5a. For overexposure, this resulted in a 45.3% true positive rate and a false negative rate of 43.8%. Under the OE-RS training condition (shown in Figure 4.5b), the true positive percentage for overexposure became 99.0%. Our true positive percentage for underexposure decreased to 61.8%, with a false negative percentage on focus at 28.2%. Figure 4.6 exemplifies our blur classification segmentation results using the OE-RS training condition with the TPOE-R test condition.

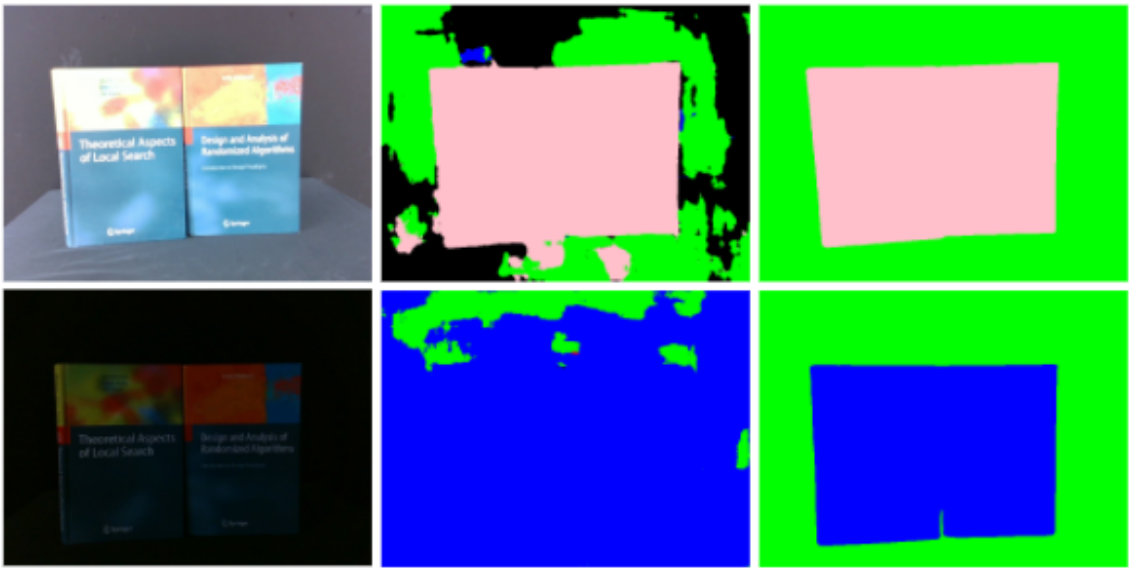
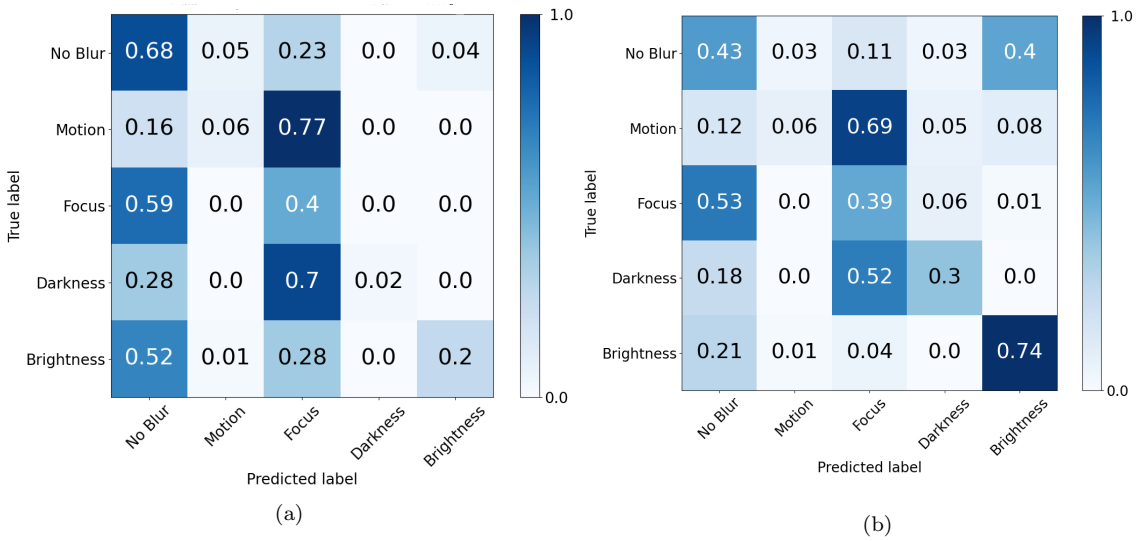


Figure 4.6: Blur result examples from our exposure dataset. Blur types: Black - No Blur, Red - Motion, Green - Focus, Blue - Underexposure, Pink - Overexposure. Left: Input images. Middle: Our results. Right: Ground truth. Top to bottom: Examples of overexposure and underexposure blur.

4.1.4 TO-R Results Comparison



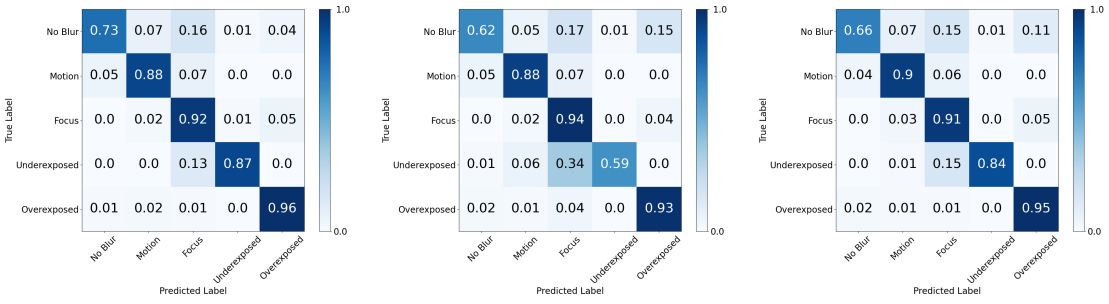
(a) (b)

Figure 4.7: Confusion matrix results using TO-R: (a) PC-RS, (b) OE-RS.

Figure 4.7 displays the confusion matrix results for PC-RS and OE-RS on the testing condition TO-R. We see high classification error for over- and underexposure pixels using PC-RS (2% underexposure true positive and 20% overexposure true positive). OE-RS did better for overexposure true positive results (74%), but underexposure pixels had a high misclassification rate in which 52% of underexposure pixels were classified as focus blur. Because of these poor results, we created the O-R, O-S, and O-RS training conditions. Figure 4.8 shows the confusion matrix results for O-R, O-S, and O-RS on the TO-R testing condition. Table 4.2 shows the mean overall blur accuracy for each training condition. We can see that the best mean accuracy and confusion matrix results come from the O-R training condition. Because of this better accuracy, we used the O-R training condition weights. Figure 4.9 shows some examples from the O-R training condition on the TO-R testing condition. As shown, the background is labeled as focus blur. Because of this, we did a salience masking for the object of interest and only considered blur pixels in that region when we connected this to our IPS network. The final blur pixel calculated for our IPS comes from the maximum number of pixels classified in this region of interest.

	O-R	O-S	O-RS
Mean Accuracy	90.34	84.96	89.23

Table 4.2: Mean accuracy across TO-R test condition



(a) (b) (c)

Figure 4.8: Confusion matrix results using TPOE-R: (a) O-R, (b) O-S, (c) O-RS.

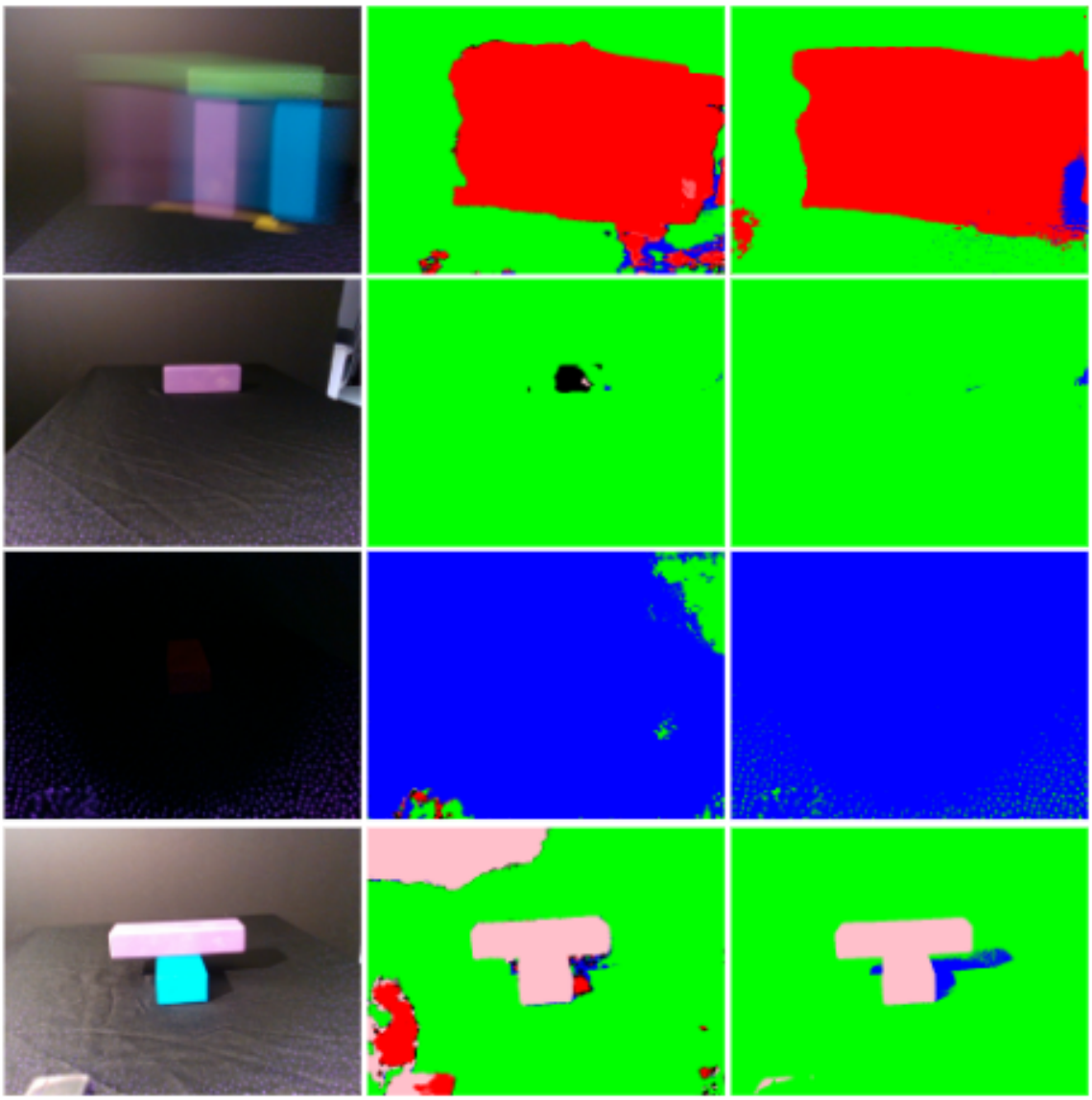


Figure 4.9: Blur Result Examples From TO-R Test Dataset. Blur types: Black - No Blur, Red - Motion, Green - Focus, Blue - Underexposure, Pink - Overexposure. Left: Input images. Middle: Our results. Right: Ground truth. Top to bottom: Examples of motion, focus, underexposure and overexposure blur.

4.2 Light Direction & Angle-of-View Failure Prediction

We split these results into four sections: ground truth results (subsection 4.2.1) from the Physics Intuition (PI) Network of our dataset, SS results with light direction and angle-of-view accuracy comparisons (subsection 4.2.2), LS results (subsection 4.2.3), and AS results (subsection 4.2.4). Appendix A.2 lists the training and testing names and descriptions that are referenced in this section.

4.2.1 PI Ground Truth and Failure Rate Across Light Directions and Angles of View

Table 4.3 provides an example of the output of our RGB images run through PI. The mean accuracy of PI across our whole dataset was 63.3%, and its precision was 68.1%. This was expected because the masks are not perfect and there was a noise difference in accuracy when a new dataset is introduced. The masks represent real-world masking difficulties with colors and salience detection. Another reason for the low accuracy is that any trained weights are expected to perform worse for new, unseen scenarios. Figure 4.10 shows an example of our main dataset masked images.

filename	gt_label	prediction	our_ gt_labels
env-h=3-s=79-n=3-r=0/rgbvseg_c=0-r=0.png	1	1	1
env-h=3-s=48-n=3-r=0/rgbvseg_c=0-r=0.png	1	1	1
env-h=5-s=169-n=5-r=1/rgbvseg_c=0-r=1.png	0	1	0
env-h=4-s=106-n=4-r=1/rgbvseg_c=0-r=1.png	0	0	1
env-h=4-s=149-n=4-r=1/rgbvseg_c=0-r=1.png	0	1	0
env-h=2-s=18-n=2-r=1/rgbvseg_c=0-r=1.png	0	0	1
env-h=3-s=65-n=3-r=2/rgbvseg_c=0-r=2.png	1	1	1
...	...	...	
env-h=3-s=60-n=3-r=0/rgbvseg_c=0-r=0.png	0	1	0
env-h=2-s=30-n=2-r=1/rgbvseg_c=0-r=1.png	0	1	0
env-h=3-s=74-n=3-r=0/rgbvseg_c=0-r=0.png	1	1	1
env-h=4-s=156-n=4-r=0/rgbvseg_c=0-r=0.png	1	1	1
env-h=3-s=64-n=3-r=0/rgbvseg_c=0-r=0.png	1	1	1

Table 4.3: Example GT Data format from Physics Intuition Network (PI)

These masks as shown are not perfect and represent the real-time issues that arise from prediction problems.

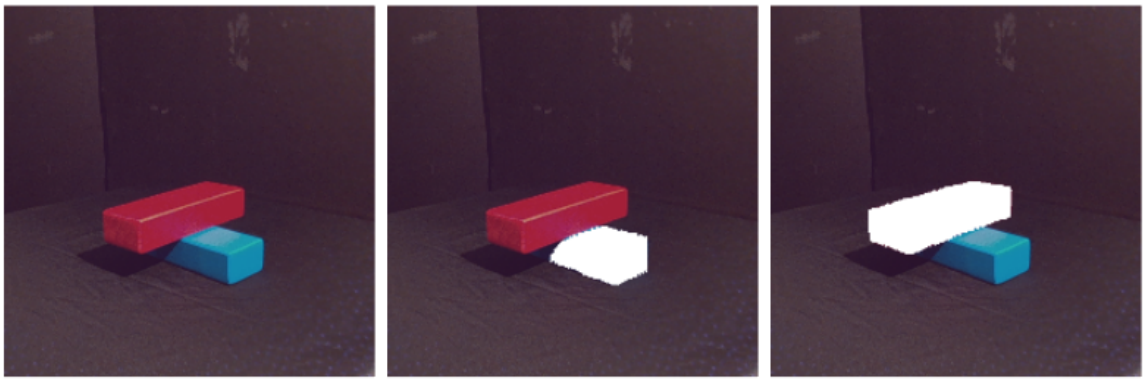


Figure 4.10: An example of our dataset with masking

Table 4.4 displays the number of success and failure predictions from the PI across the different light directions and angles of view in our dataset. For light directions, we see a slight increase in failure rate (2.3% from front and 3.6% from right) compared to the left light direction. The right light direction has the least amount of failure. For angle-of-view, front and right views have the same rate of failure and the left has an increase of 1.2% in failure relative to the other views. This provides evidence that failure can increase or decrease based on light direction and angle-of-view.

Light Direction	Success	Failure
Front	57.7	42.3
Right	58.2	41.8
Left	56.7	43.3

Angle-of-View	Success	Failure
Front	57.7	42.3
Right	57.7	42.3
Left	57.2	42.8

Table 4.4: PI success and failure rates by light direction (left) and angle of view (right)

4.2.2 SS Results

Table 4.5 shows our testing results for each unique run. The best performance on the testing set was run11 using AlexNet and including depth. ResNet18 seemed to overfit to the training set. Figure 4.11 shows some example results from run11. Green

means that there was a success prediction from our network and red means there was a failure prediction from our network.

Run Unique ID	Accuracy	Precision
run1	81.3	87.2
run2	75.4	78.4
run3	79.2	81.2
run4	82.5	84.6
run5	75.4	78.4
run6	73.8	73.8
run7	71.5	71.1
run8	74.7	74.3
run9	69.0	68.4
run10	64.5	67.2
run11	83.6	83.6

Table 4.5: SS Testing Results (percentage)

Table 4.6 displays the accuracy per angle-of-view and light direction. What we can observe is that there was higher accuracy for the left light direction, but lower accuracy for the left angle-of-view. These results show similar accuracy across the different light directions and angles of views for scenarios that were not present in the training dataset. For this SS split, we see that each scene can change in accuracy across the different angles of view and light directions.

Light Direction	Accuracy	Precision
Front	83.7	85.7
Right	80.9	80.9
Left	86.1	84.7

Angle-of-View	Accuracy	Precision
Front	89.3	89.3
Right	85.5	86.1
Left	76.4	75.9

Table 4.6: Accuracy Based on Light Direction (left) and Angle-of-View (right)

4.2.3 LS Results

Table 4.7 shows our testing results in percentages for each unique run. The best performance on the testing set was run11, which used AlexNet and depth. ResNet18 seemed to overfit to the training set again but was better than SS performance for

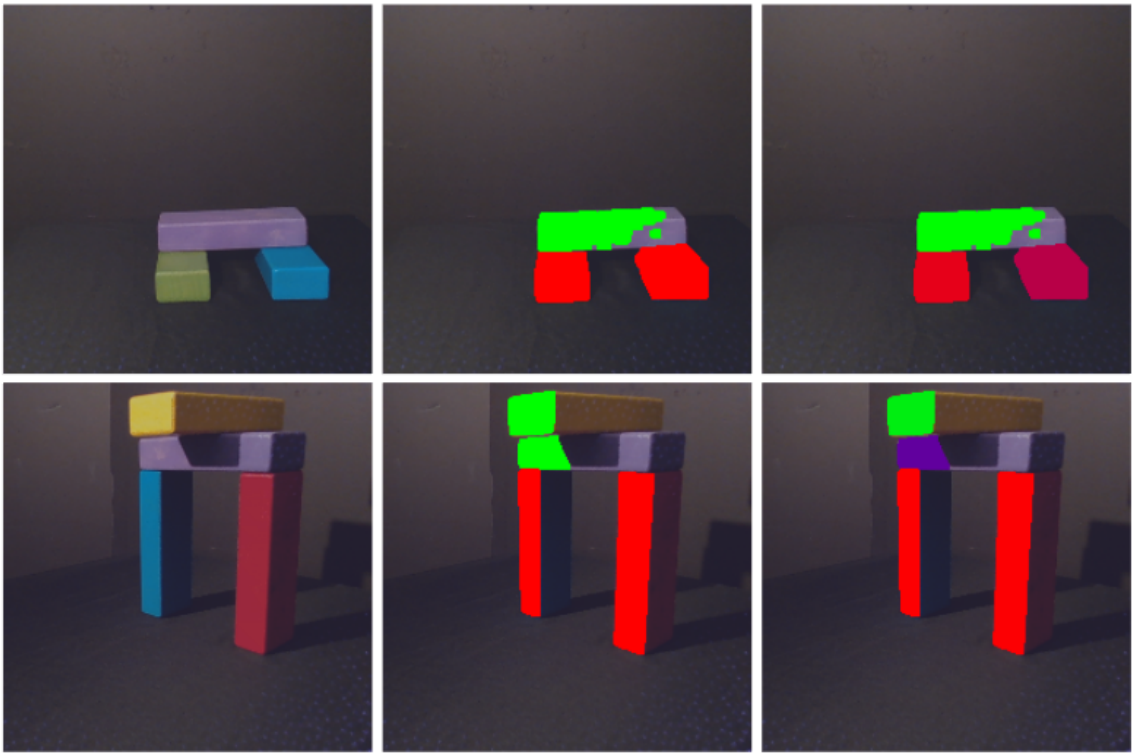


Figure 4.11: Test examples of input (left), GT (middle) and our (right) prediction of system failure results from run11 weights for each block for SS split

ResNet18. Figure 4.12 shows some example results from run11. Again, green identifies a successful prediction from the network, while red shows a failure prediction from our network. For the LS split, the test set includes all scenarios where one light direction is present that was not present in the training dataset. Because the dataset was split such that all scenarios were present in the training dataset, an evaluation of accuracy across all light directions would be skewed to the training dataset light directions. It would not give accurate estimation of light direction accuracy across all light directions taken of the scene. Table 4.8 shows the angle-of-view-based accuracy for this LS split. We saw a small decrease in accuracy for the left angle-of-view (79.2%) compared to the other two (front 88.3% and right 96.4%).

Run Unique ID	Accuracy	Precision
run1	73.4	70.4
run2	74.2	74.2
run3	72.5	69.8
run4	70.0	69.4
run5	75.2	71.9
run6	81.5	80.5
run7	65.0	63.0
run8	51.4	66.0
run9	72.0	69.1
run10	72.3	69.2
run11	87.1	87.1

Table 4.7: LS Testing Results (percentage)

Angle-of-View	Accuracy	Precision
Front	88.3	88.3
Right	96.4	90.0
Left	79.2	82.3

Table 4.8: Angle-of-View Based Accuracy using LS Split

4.2.4 AS Results

Table 4.9 shows our testing results for each unique run by percentage. Tied for best performance on the testing set were run5 and run11, both using AlexNet. Figure

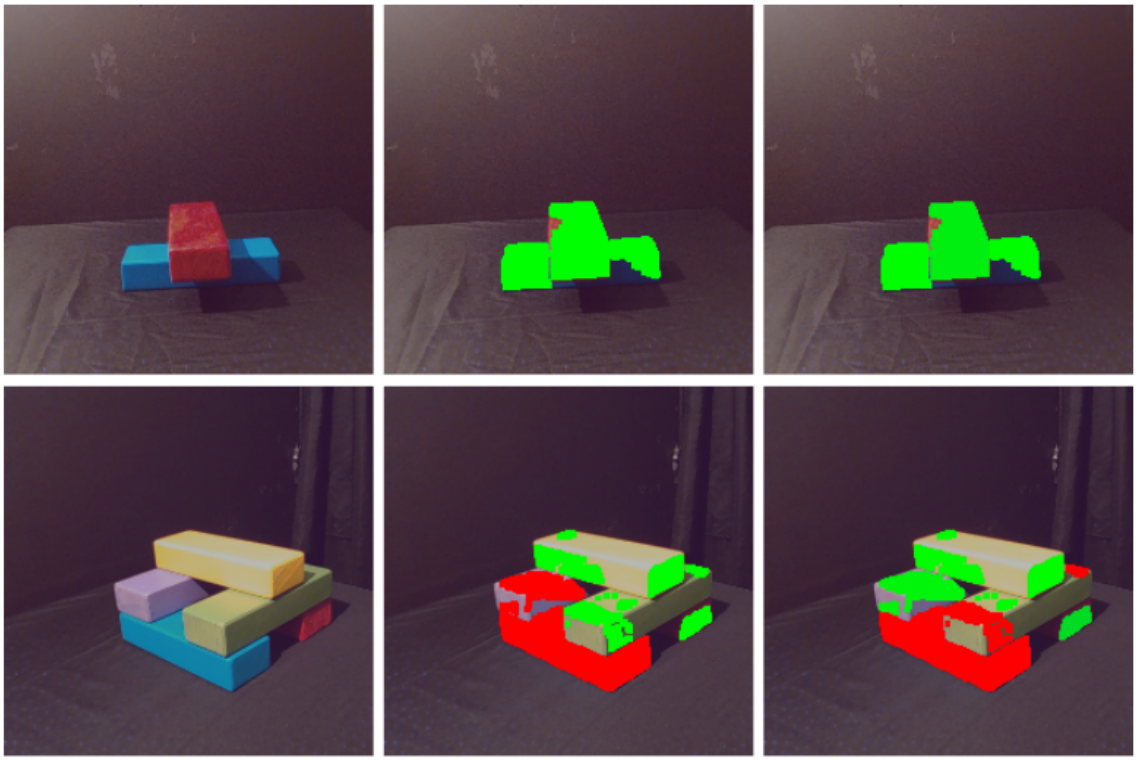


Figure 4.12: Test examples of input (left), GT (middle) and our (right) prediction of system failure results for each block for LS split using run11 weights

Run Unique ID	Accuracy	Precision
run1	76.8	73.6
run2	75.0	71.6
run3	73.0	72.2
run4	74.5	72.6
run5	82.2	77.8
run6	79.3	75.3
run7	58.5	58.0
run8	75.4	72.7
run9	62.1	60.4
run10	57.9	62.2
run11	82.2	77.8

Table 4.9: AS Training Scenarios (percentage)

4.13 show some example results from run11. We can see for angle-based predictions that this network had good accuracy for the angle-of-view that was not present in the training dataset. We did a per light direction-based accuracy analysis in which we examined each separate light direction in the AS test split (table 4.10). The right light direction had the highest accuracy (86.4%), similar to LS's angle-of-view analysis.

Light Direction	Accuracy	Precision
Front	82.1	72.2
Right	86.4	85.0
Left	79.2	76.2

Table 4.10: Light Direction-Based Accuracy using AS Split

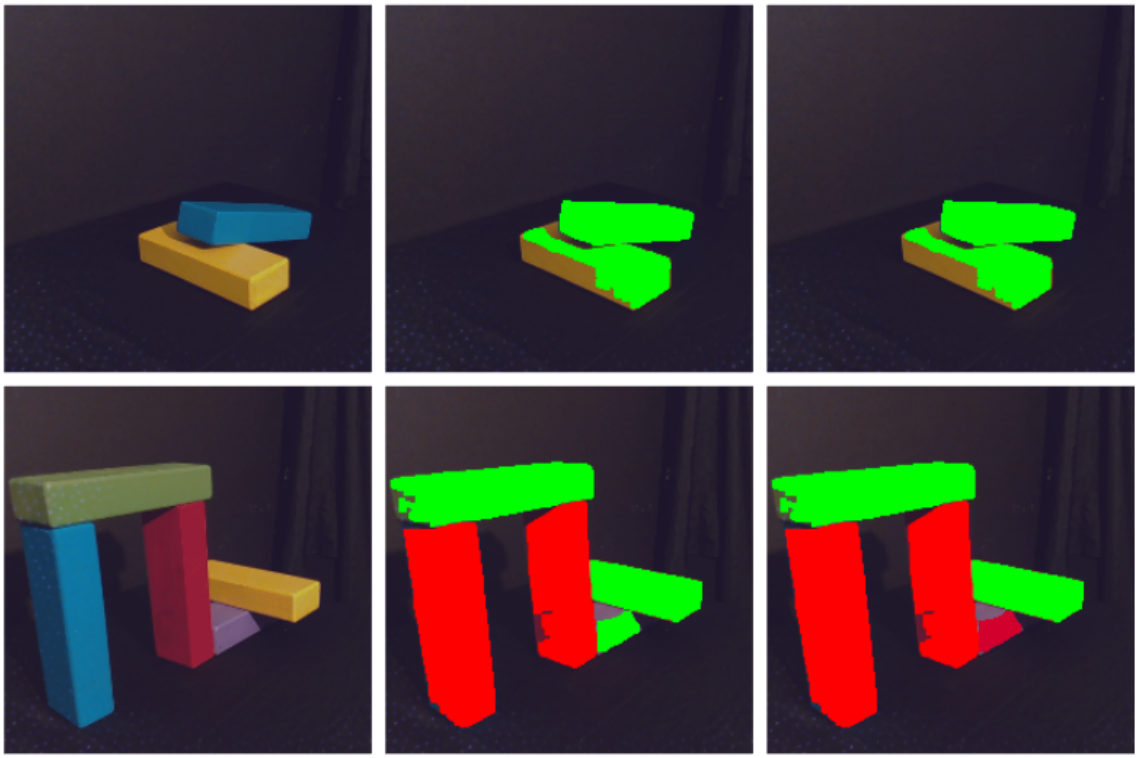


Figure 4.13: Test example of input (left), GT (middle) and our (right) prediction of system failure results for each block of AS split using run11

4.3 Introspective Perception Network (IPS)

Lastly, we evaluated our final Introspective Perception System (IPS), which combined the blur detection and classification with the light direction and angle-of-view failure prediction network. This examined the final IPS network for these common failure issues. This section is split into each of the evaluation metrics that we used for our system: Performance measurement of Physics Intuition (PI) with IPS (APR in subsection 4.3.1), performance measurement of our IPS with predicting failure for PI (APRF in subsection 4.3.2), blur performance evaluation (ABDC in subsection 4.3.3, and the preventive risk analysis (RAM in subsection 4.3.4). We have an overview of our IPS testing scenario names and short descriptions in appendix section A.3.

4.3.1 Accuracy, Precision, & Recall of Physics Intuition (APR)

Accuracy	MB	OB	UB	SSA	ASA	SSL	LSL
PI Standalone	81.0	51.0	26.0	86.8	74.5	76.6	74.5
PI w IPS	_	_	0	82.9	63.6	100	100
PI w IPS wo Blur	81.7	45.8	26.0	82.9	65.6	91.3	86.7

Table 4.11: PI Accuracy per Test Scenarios

Precision	MB	OB	UB	SSA	ASA	SSL	LSL
PI Standalone	81.0	56.1	0.4	88.8	75.4	77.3	77.0
PI w IPS	_	_	0	88.0	67.6	100	100
PI w IPS wo Blur	81.8	45.7	0.4	88.0	69.9	91.7	92.4

Table 4.12: PI Precision per Test Scenarios

Tables 4.11, 4.12, & 4.13 display the accuracy, precision, and recall for the Physics Intuition (PI) System across each scenario tested. We gathered the precision, accuracy and recall of PI standalone without IPS, PI with IPS connected, and PI with IPS connected but without the blur detection and classification component. For IPS data that was marked as a failure, we did not count the data in our metrics because it

Recall	MB	OB	UB	SSA	ASA	SSL	LSL
PI Standalone	81.0	42.9	4.6	84.2	75.0	76.6	76.2
PI w IPS	_	_	0	83.9	63.7	100	100
PI w IPS wo Blur	81.8	42.9	4.6	83.9	67.3	89.5	91.1

Table 4.13: PI Recall per Test Scenarios

would have been thrown out of the overall system as a failed image. We saw an increase in accuracy, precision and recall when IPS was added for Motion (MB), Light Direction with SS (SSL), and Light Direction with LS (LSL). For Overexposure (OB), Motion Blur (MB), and Underexposure (UB), our blur detection and classification did not have metrics for the IPS with blur detection because our blur detection usually detected blur that labeled all of the objects as failure. Because MB & OB scenarios had no predictions that were marked as success (no 1—non-failure—predictions), we could not accurately calculate accuracy, precision, and recall for PI using IPS that was connected with blur detection. This was an apparent disconnection in the IPS failure prediction and blur detection that is discussed more in subsection 5.3.2. For Underexposure (UB), color thresholding failed for most of the underexposure dataset images. Therefore, only a few positive detections reached the PI network, which resulted in overall poor accuracy, precision, and recall for PI. The Angle-of-View test scenarios (ASA and SSA) have lower metrics when using IPS than when not using IPS. This we believe to have resulted from a combination of the amount of data that was used for testing and the way that the dataset was set up for ASA to include all scenes, resulting in poor performance of scenes that were not included in the training dataset. We discussed this more in subsection 5.3.2.

	Accuracy	Precision	Recall
PI Standalone	66.9	70.3	63.6
PI w IPS	83.3	83.6	83.4
PI w IPS wo Blur	77.8	78.7	78.2

Table 4.14: PI Accuracy, Precision, & Recall - All Scenarios (points that did not have a PI prediction were marked as a failed PI prediction)

Tables 4.14 and 4.15 show the PI accuracy, precision and recall across all testing scenarios. We reported the total amount of data points even if we did not obtain a PI

	Accuracy	Precision	Recall
PI Standalone	73.9	74.9	73.1
PI w IPS	83.3	83.6	83.4
PI w IPS wo Blur	77.8	78.7	78.2

Table 4.15: PI Accuracy, Precision, & Recall - All Scenarios with Actual IPS Predictions (points that did not have a PI prediction were removed from the dataset for evaluation)

prediction (table 4.14) and the actual amount of data points with PI predictions (table 4.15). We set the points we did not get predictions (ex: underexposure data points) to incorrect (0) for PI prediction since the PI did fail to make a prediction, and thus failed in its perception task. If PI failed to detect a block to make a prediction, it was marked as a failure in the overall perception task since it needed to make a prediction to either fail or succeed. We observed that the metrics increased when using IPS across all data points, and specifically for when blur detection was used. Figure 4.14 shows an example of a failure prediction for a motion image. We successfully predicted that the PI system would fail in its perception task.

Table 4.16 are the percentages of the number of images that our IPS kept and did not mark as failure. This gave us good evidence of how many dataset points were removed to how the PI accuracy values changed. Our IPS for the scenarios where accuracy was increased (SSL and LSL) removed the necessary amount of images that were failure prone. For the blur testing scenarios (MB, OB, and UB), the percentage of kept images was 0%. This created the low accuracy in our IPS system for those test scenarios since we have little to no positive data points to evaluate on for calculating a balanced accuracy measurement.

	MB	OB	UB	SSA	ASA	SSL	LSL
PI w IPS	0	0	4.4	47.1	29.1	22.7	27.4
PI w IPS wo Blur	52.4	33.3	100	47.1	32.6	45.4	59.0

Table 4.16: IPS % Kept Predictions per Test Scenarios

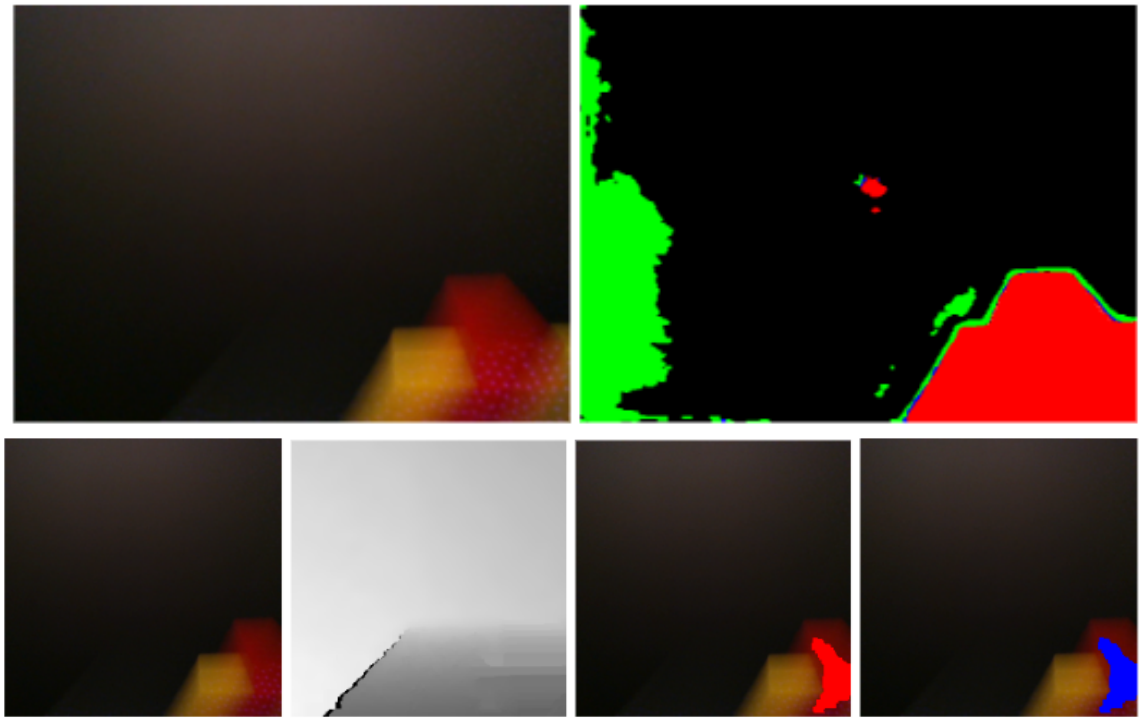


Figure 4.14: Motion Image Example Input (Top left), Blur Output: No Blur - Black, Motion - Red, Focus - Green, Underexposure - Blue, Overexposure - Pink (Top Right), Cropped Input Image (Bottom most left), Cropped Depth Image (Bottom second left), Cropped IPS masked prediction: Red - failure, Blue - success (Bottom second right), Cropped Ground Truth PI Red - non-removable, Blue - removable (Bottom most right).

4.3.2 Accuracy, Precision, & Recall of Prediction of Failure (APRF)

Accuracy	MB	OB	UB	SSA	ASA	SSL	LSL
IPS	50.0	50.0	52.5	48.1	39.7	64.7	68.0
IPS wo Blur	51.5	50.0	_	48.1	42.2	66.4	76.5

Table 4.17: IPS Accuracy per Test Scenarios

Precision	MB	OB	UB	SSA	ASA	SSL	LSL
IPS	3.6	32.7	91.5	73.8	52.1	84.0	84.4
IPS wo Blur	70.0	51.0	_	73.8	56.0	77.1	82.3

Table 4.18: IPS Precision per Test Scenarios

Recall	MB	OB	UB	SSA	ASA	SSL	LSL
IPS	19.0	57.1	91.4	47.9	30.3	45.3	51.1
IPS wo Blur	52.4	52.4	_	47.9	34.1	59.4	73.3

Table 4.19: IPS Recall per Test Scenarios

Tables 4.17, 4.18, and 4.19 display the accuracy, precision, and recall of our IPS failure prediction across the testing scenarios. Each scenario had either an increase or no change in IPS accuracy when blur detection was removed. Since the number of our testing data points was limited per scenario, these numbers could change drastically with a larger dataset. This could also be caused by the disconnect with blur detection and the failure predictions. For example, motion blur (MB) examples would always predict a failure for all objects that were present in the image. Our light direction scenarios (SSL and LSL), had the highest metrics for failure prediction. Motion Blur (MB) and Overexposure Blur (OB) had an increase in accuracy without blur detection since as explained in table 4.16, the percentage of retained images was higher when blur detection was not used in our IPS. We believe SSA and ASA had low IPS failure prediction accuracy and recall because the known limitation of our dataset, discussed in subsection 5.3.2. Our testing dataset for ASA had scenarios that were not present in the dataset thus we believe caused a decrease in the accuracy.

	Accuracy	Precision	Recall
IPS	58.0	68.0	49.0
IPS wo Blur	62.5	66.5	58.7

Table 4.20: IPS Accuracy, Precision, & Recall - All Scenarios (points that did not have a PI prediction were marked as a failed PI prediction)

	Accuracy	Precision	Recall
IPS	55.6	69.0	41.4
IPS wo Blur	55.9	65.6	52.5

Table 4.21: IPS Accuracy, Precision, & Recall - All Scenarios with Actual IPS Predictions (points that did not have a PI prediction were removed from the dataset for evaluation)

The IPS failure metrics across all test scenarios are shown in Tables 4.20 & 4.21. These tables show approximately 50% accuracy across all of the scenarios. This measurement is not great, but this could increase if the amount of data points increased.

4.3.3 Accuracy of Blur Detection & Classification (ABDC)

Figure 4.15 displays the blur confusion matrix of the IPS testing dataset. We combined the no blur and focus blur labels because focus blur was not considered as blur in our testing dataset. The images as a whole were blur-label segmented and then salience detection was used to define an area of interest. This process was used to finding the max about of blur that was present in that area. This is because the image as a whole would always be defined as no blur because the majority of the pixels are present in the background. We only care about the object of interest for blur definition and classification. As shown in the confusion matrix, we obtained perfect accuracy for motion blur and overexposed images. However, focus and motion blur would get confused with each other, with a 32% false prediction of motion for focus/no blur images. Overall, our blur detection and classification sometimes worked so well that no positive detection (prediction success) would be present in the testing scenario (discussed in subsection 5.3.2). Figure 4.16 displays some examples from our IPS testing dataset, with their corresponding blur output.

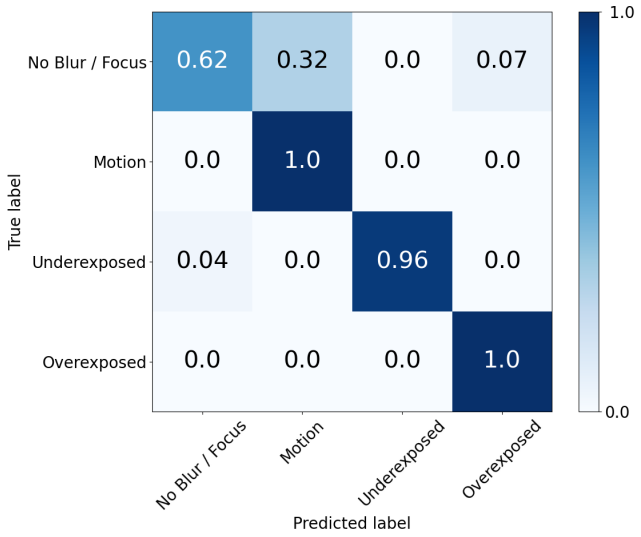


Figure 4.15: IPS Test Dataset Blur Confusion Matrix

4.3.4 Risk-Averse Metric (RAM)

Figure 4.17 displays the per scenario Risk-Averse Plot when not using IPS and when using IPS with or without the Blur Detection. Risk values ranged from 0 to -2.0. The higher the sum value, the better the perception task performance is for risk mitigation. For overexposure, IPS with and without blur had better performance than PI standalone after a risk factor of -0.75 was introduced. This means that if the risk of making an incorrect prediction increased by more than 0.75, compared to 1.0 for a correct prediction, IPS gave the best risk-averse performance. Light Direction scenarios (SSL subfigure (f) and LSL subfigure (g)) had better performance than PI standalone after a risk factor of $\approx$ -1.0 (LSL) & -2.0 (SSL) was introduced. Motion (MB) and Angle-of-View scenarios (SSA and ASA) trended towards better performance with higher risk values but did not lead to better performance if risk was less important. We think this was due to the amount of data points that was present in the test dataset. Underexposure (subfigure (c)) had a high risk factor that IPS mitigated throughout all the possible risk values.

Figures 4.18 and 4.19 display the RAM across all scenarios with all datapoints included (Figure 4.18) or just actual IPS predictions (Figure 4.19). We see that risk-averse performance was better for PI with IPS without blur detection when the

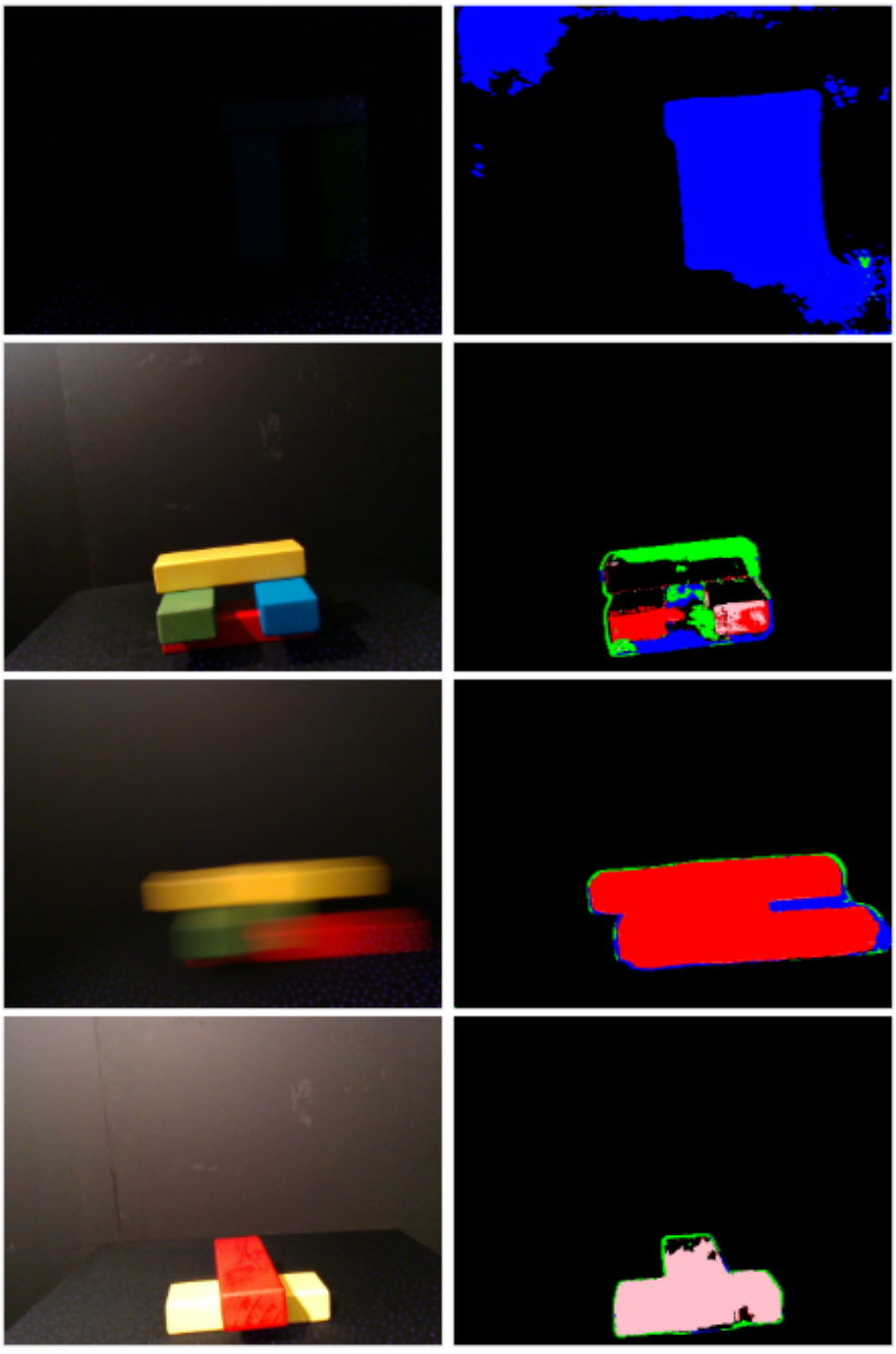


Figure 4.16: IPS test dataset blur inputs (Left) and blur segmented output (Right); Underexposure (Top), No Blur/Focus (Second to Top), Motion (Second to Bottom), and Overexposure (Bottom). Blur Labels: No Blur - Black, Motion - Red, Focus - Green, Underexposure - Blue, Overexposure - Pink

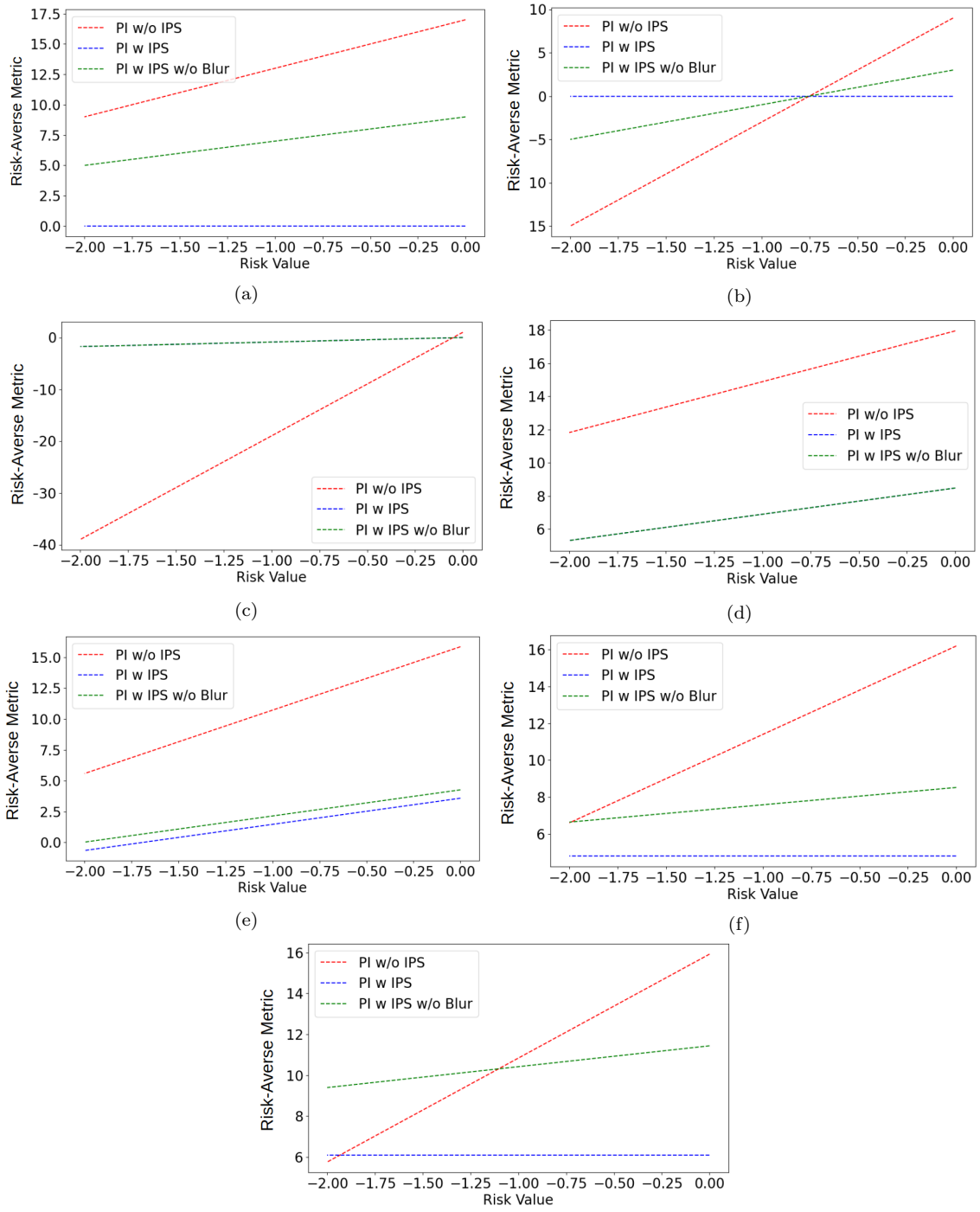


Figure 4.17: RAM results across IPS testing scenarios: (a) Motion, (b) Overexposure, (c) Underexposure, (d) Angle of View w/ SS, (e) Angle of View w/ AS, (f) Light Direction w/ SS, and (g) Light Direction w/ LS

risk value was ≈ -1.1, compared to 1.0 for a correct prediction (Figure 4.18). This means that when the risk for making an incorrect prediction was the same as for making a correct prediction, then using IPS resulted in better risk-averse performance. Overall, using our IPS system did make a difference where risk of failure was very high and incorrect predictions were detrimental.

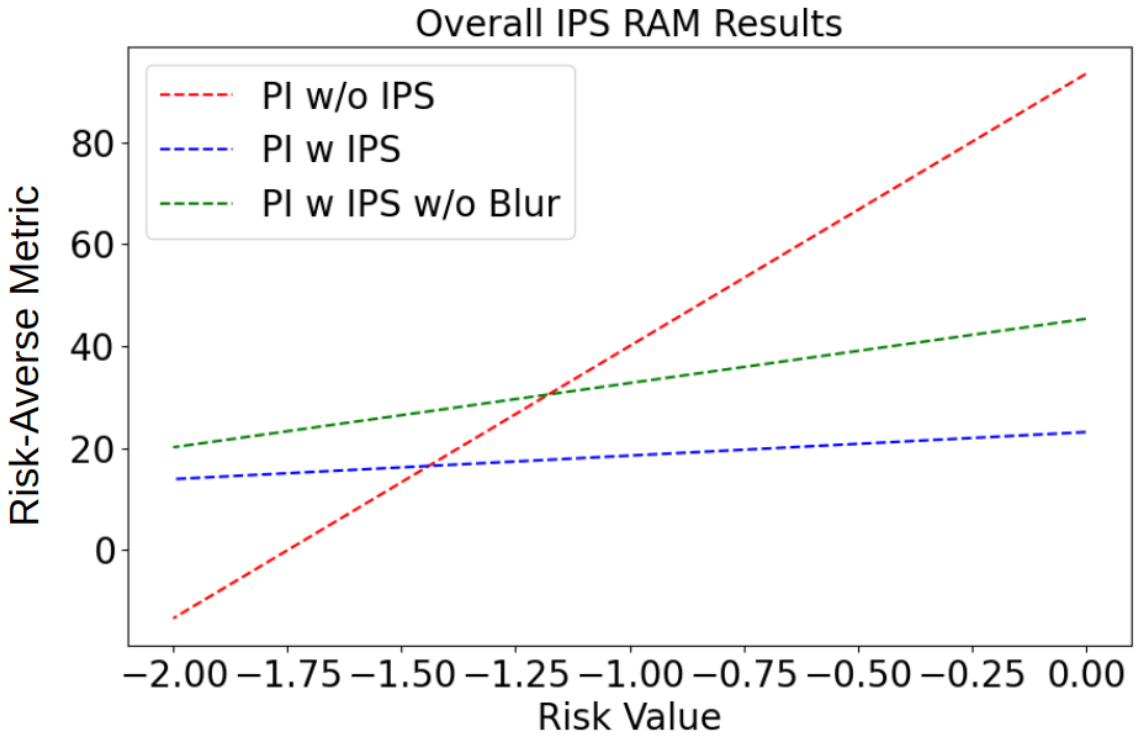


Figure 4.18: RAM Plot - All scenarios (points that did not have a PI prediction were marked as a failed PI prediction)

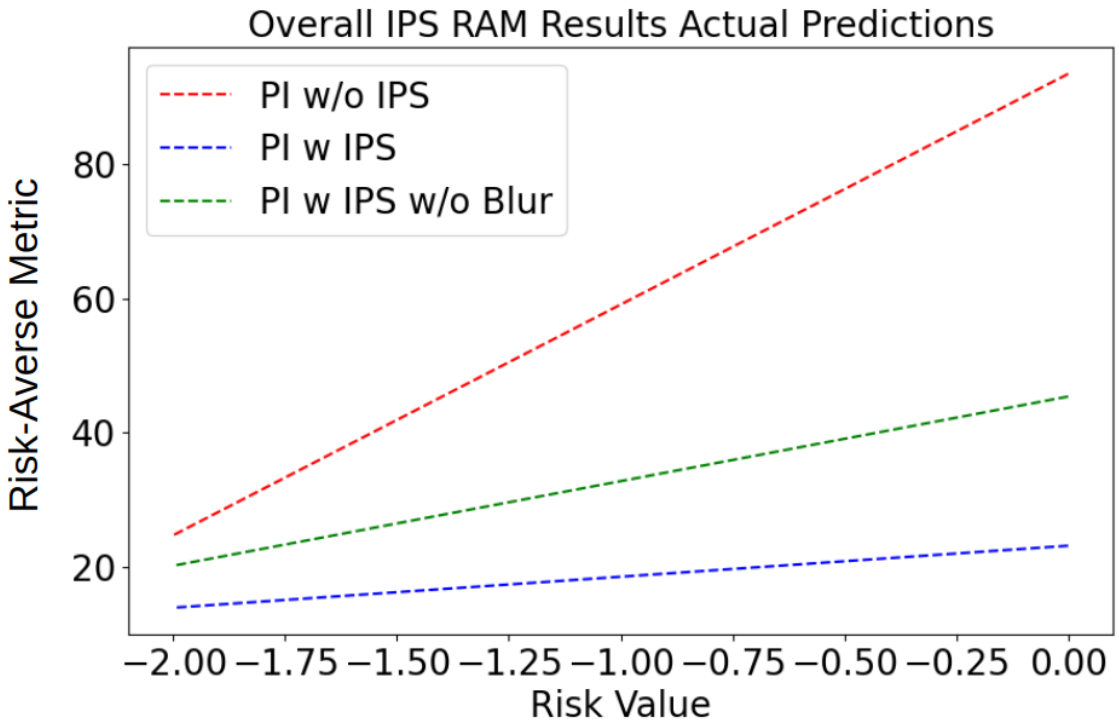


Figure 4.19: RAM Plot - All scenarios with actual IPS predictions (points that did not have a PI prediction were removed from the dataset for evaluation)

Chapter 5

Discussion and Future Work

In this chapter, we discuss the limitations and other observations we found while doing this research. This chapter has three sections: Subsection 5.1 discusses how overexposure and underexposure should be considered as blur, the extra experiments where we look at possible issues that arise from blur detection and classification, and future research in this area. Subsection 5.2 discusses the limitations in the current angle-of-view and light direction failure prediction dataset and ways our method can be extended to future work. Lastly, in subsection 5.3, we talked about limitations of our IPS network and how future work can improve upon these limitations.

5.1 Blur Detection and Classification

We have shown that our model accurately labels over- and underexposure blur in addition to motion and focus blur. In this section, we discussed our detection and classification results (subsection 5.1.1), demonstrate why over- and underexposure should be considered as blur (subsection 5.1.1), and argue why current over- and underexposure correction methods cannot completely solve this problem in real time (subsections 5.1.2 and 5.1.3), especially in a real-world robotics application. We suggest future improvements to our blur detection and classification that would further this area of research and help future roboticists mitigate blur more accurately (subsection 5.1.4). Appendix A.1 contains a list of the training and testing condition names that are referenced in this section.

5.1.1 Detecting Over/Underexposure as Blur

The baseline blur detection and classification method was able to classify over- and underexposure as blur and not distinguish it from focus and motion blur (section 4.1.2). This can lead to poor correction performance and results in perception errors. Our network was able to distinguish over- and underexposed pixels, which can allow for better perception results. In our pick-and-place recorded image results from section 4.1.3, over- and underexposed pixel classification was confused with the background of the image. We expect this was because the background can be both under- or overexposed and out-of-focus, and the network prioritizes focus blur in its predictions. However, we could obtain accurate blur classification results when fine-tuning our model for the specific perception task (subsection 4.1.4). This suggests blur detection and classification can detect perception errors during real robot use.

5.1.2 Auto-exposure Failures

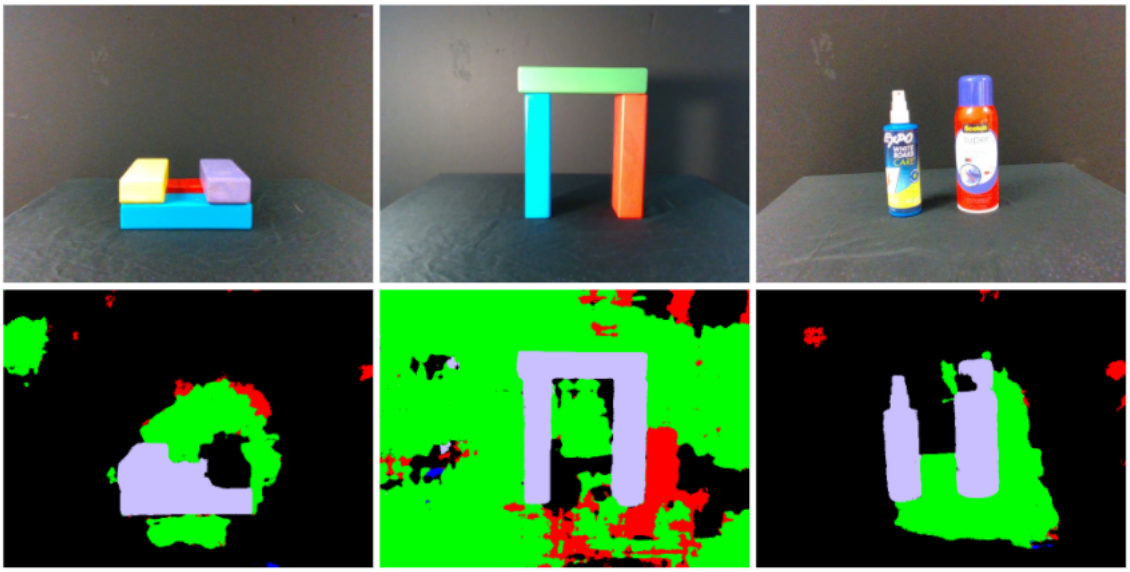


Figure 5.1: Examples of auto-exposure failures with OE-RS results. Top: Input images. Bottom: OE-RS results.

Figure 5.1 displays a few auto-exposed images that were taken using a RealSense D435 camera with the default RealSense2 ROS camera package auto-exposure setting.

As shown, auto-exposure failed to evenly expose the objects of interest in these images. When running these images through our network using the OE-RS training condition, we saw that our results classify each of these objects of interest as overexposed. This showed that our network can correctly identify overexposure that could not be resolved through auto-exposure methods.

5.1.3 Correcting Blur Requires Accurate Correction Methods

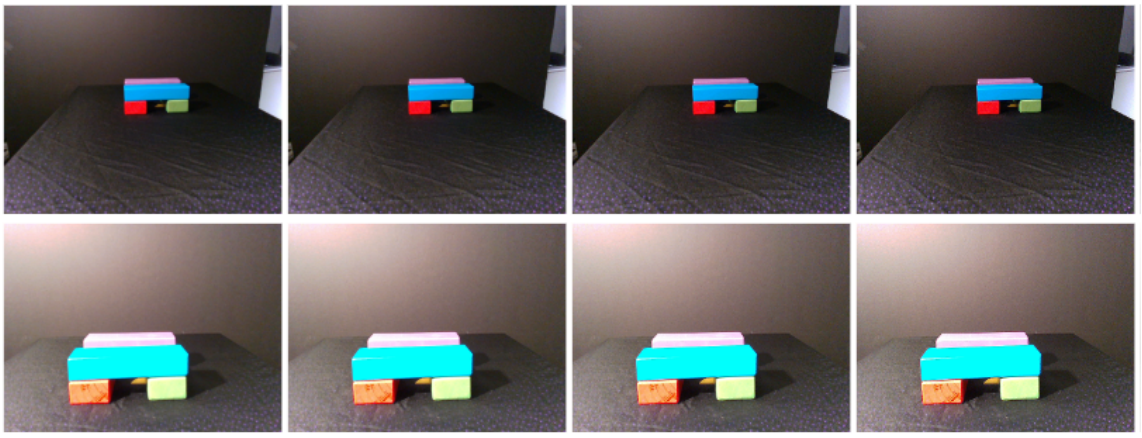


Figure 5.2: Top, left to right: Out-of-focus original, sharpened 1/2/3x. Bottom: Overexposed original, sharpened 1/2/3x.

Blur detection and classification networks are typically used to see if an image needs to be corrected internally (e.g., using focal lens change, refocusing, sharpening and de-convolution). However, these methods fail if the blur detection process misclassifies the blur or if the image cannot be fixed internally (e.g., it requires a lighting change in the robot's environment). Figure 5.2 (bottom left) is an example of an overexposed image taken in the context of a robot's pick-and-place task. Figure 5.2 (top left) displays an image that is is out of focus. Both images were taken by a RealSense D435 camera and were from the TO-R Testing Condition. Figure 5.3 reports the percentage of pixels that were classified as blurred by Kim et al. [19] after each correction attempt for both the out-of-focus image and the overexposed image. If this method were classifying all overexposed images as focus blur, then it would

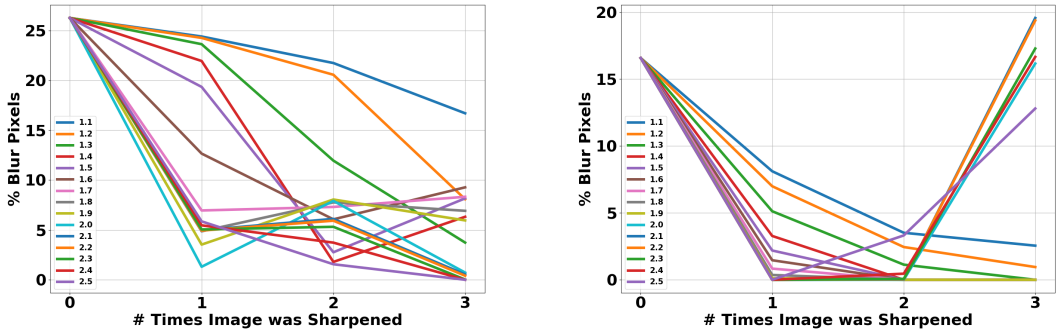


Figure 5.3: Left: Percentage of pixels classified as blurry in an out-of-focus image vs. number of times corrected using sharpness values between 1.1 to 2.5. Right: Percentage of pixels classified as blurry in an overexposed image vs. number of times corrected as out-of-focus using sharpness values between 1.1 to 2.5.

try to correct that blur by sharpening the image. The baseline method classified the out-of-focus example image (Figure 5.2, top left) and the overexposed image (Figure 5.2, bottom left) as no-blur. We corrected both images for focus by resharpening them repeatedly using Pillow's Image Enhance Sharpness tool in Python with a varying sharpness factor from 1.1 to 2.5 to identify correction patterns. The chart for out-of-focus image sharpening (Figure 5.3) indicates that our attempt to correct this image did trend towards refocusing because the percentage of blur in the image decreased as we continually sharpen the image. This indicated that this artificial perception correction method is specific to the out-of-focus blur type. Sharpening an overexposed image (Figure 5.3) performed similarly to an out-of-focus image being sharpened. However, when the sharpness factor grew in magnitude (over 2.0), we actually increased the blur pixel percentage; thus, the correction method failed. Another effect is that after 2 or 3 iterations of sharpening, the image had sharpness artifacts that could cause major errors in object detection or other perception tasks that require realistic imaging. As seen in the sharpened, overexposed images in Figure 5.2, the sharpening improved focus blur up to a certain point, but it resulted in sharpness artifacts while also not reducing overexposure blur.

5.1.4 Future Work

There are several areas for future work that can improve our current blur detection and classification method. First, future work could expand the range of underexposure images included in our dataset. Real underexposed images that auto-exposure failed to correct were hard for our network to detect. One of the hardest issues that we had to overcome was the dataset creation for over- and underexposure images. The range of overexposure changes with each image. This effect was not unnoticed and that was why the ranges for over- and underexposure α & β were very large. Many of our created dataset images for underexposure blur were very dark, so the blur detection network learned more about pixels being of a collection of zero pixels instead of a collection of pixels with dark edges & spots that are underexposed. This might be one of the reasons why the accuracy of the darkness images was high. In future work, creating a better range of values for our dataset would improve training and testing accuracy. Maybe using another exposure dataset such as [1] could expand this dataset and add to the possible exposure rendering possibilities that could be used in detecting over- and underexposure.

Another area of future work could be to improve image ground-truth pixel labels. Over- and underexposure blur is present in some of the CUHK original motion and focus images. Segmenting these out of the ground-truth images would increase accuracy. Other future work could improve the ground-truth image segmentation. Some images in the original CUHK dataset have subtle issues with labels that should be improved (e.g., the wheels and spokes of the motorcycle in Figure 3.9). These types of image examples and ground-truth labels could improve overall performance.

Finally, we would like to demonstrate this new blur detection capability in more real robot perception tasks besides the PI task, with the robot correcting itself by both slowing down and changing light levels to obtain better perception. Our IPS system was able to show PI perception task improvement using our blur detection and classification (see subsection 4.3) but a larger testing dataset might improve the performance of blur detection and classification. Sometimes, IPS performance was better when blur detection and classification were not used (see subsections 4.3.1, 4.3.2 and 4.3.4). We believe that if blur thresholds were tuned for perception task performance, it would improve IPS with blur detection and classification. The

thresholds that would be needed would, for example, check when the amount of focus pixels in an image changed task performance. Another failure detection network that would take the output of the blur detection and classification segmentation output as an input and output a failure prediction based on the relationship of the blur to failure observation could also improve the IPS performance using blur.

5.2 Light Direction & Angle-of-View Failure Prediction

We have shown that changes in angles of view and light directions can cause failure in our perception task. We were able to predict failure relating to light directions and angles of view in the scene. We were even able to predict failure for light directions and angles of view that were not present in the training dataset. Even though we were able to get good accuracy during training and testing, we were not able to replicate some of the good accuracy in the combined IPS testing dataset. Appendix B.1 has a list of the training and testing splits that are referenced in this section.

5.2.1 Limitations and Future Work

The biggest issue with this angle-of-view and light direction prediction system is our testing and validation dataset split. Angle-of-view weights did not perform well during IPS testing when predicting the angle-of-view that was not present in the training dataset. We did not anticipate how predicting the angle-of-view that was not present in the training dataset would affect predicting full scenarios that were not present in the training dataset. Then, we got good results from weights where all light directions and angles of view were present in the dataset (SS). Our light direction test scenarios also performed well in our IPS network. The previous statements give evidence for a few observations:

- We can predict angles of view and light direction failures that were not present in the training dataset if we have training weights where scenes with at least two angles of view and light directions are present in the training dataset; and
- We can predict scenes that were not present in the training dataset if we have

training weights that have scenes where at least three different light directions and angles of view were present in the training dataset.

One point for future work that we identify in this area is to add scenes to testing and validation that were not present in the training dataset. Specifically, this is necessary for the training, validation and testing conditions LS and AS in which we were trying to predict one angle-of-view and/or light direction that was not present in the training dataset. More network tuning (e.g., using another optimizer or network modification) could possibly get the network with a testing accuracy of over 90% across all dataset splits. Another point of future work could be to randomly mix different angles of view and light directions in the training, validation and testing dataset. The testing and validation dataset would also be a random mix of different angles of view and light directions that were not present in the training dataset. This might improve accuracy randomness that can happen for varying failure rates across different light directions and angles of view.

5.3 Introspective Perception Network (IPS)

Our IPS network achieved good results for most of the testing scenarios. We saw how the risk-averse metric gave promise for how our IPS would improve safety for high-risk perception tasks. In this section, we give an overview of our IPS system and its generality. We then discussed our limitations and give possible solutions. Finally, we give future work suggestions that would improve our IPS performance. Appendix A.3 has a key for all of the testing condition labels that we used in this section.

5.3.1 Generality and Assumptions of Our IPS

We saw how including blur detection in our IPS, which is connected to the task of removing a block in a non-disruptive way, increased the main perception task accuracy and risk mitigation. Connecting blur detection with light direction & angle-of-view failure prediction gave better results than not using blur detection in multiple test scenarios (LSL & SSL for example). Some of the assumptions that we took in this approach of introspective perception were the following: All four types of blur affected the perception task performance, light direction & angle-of-view failure were similar

enough to be detected together, any blur in scene is a failure indicator without relation to perception task, and finally, only the final task success rate should be considered as the indicator for training our IPS. While some of these assumptions proved successful in our testing, we saw that some assumptions, like no blur relation to failure, were proven to be the exact opposite. Additional discussion on these assumptions and suggestions for improvement are described in the upcoming future work subsections 5.3.2 & 5.3.3.

We saw with some types of blur, specifically motion and focus, blur classification identifying those as failure types did not have much gain in increasing accuracy of the perception task. This gave us proof though of the generality of our system. For our perception task, motion blur did not impact on failure. However, other perception tasks, such as object identification and 3D object pose, could be greatly impacted by motion blur. Object identification is used for many applications, such as autonomous driving, medical robots, and human helper robots. For example, identifying the Eiffel Tower in a set of images would fail if focus or motion blur was present. Tasks that have high risk, such as human helper robots, need to have accurate object identification. Our blur detection and classification could increase this accuracy since it can identify these types of blur issues and enable appropriate corrective actions to be taken.

The generality of our approach was through the identification of blur. If there is a way of identifying the type of blur that causes failure, then correction methods can be tuned to the perception task needs. If motion blur affected the perception task performance and overexposure had little to no affect, then overexposure classification could be ignored and considered no blur and motion could be a classification for automatic failure prediction for the new perception task. Our blur classifications method allows for this type of variation to be implemented easily for use in most robot perception tasks.

5.3.2 Limitations

The three major limitations that we discussed are the following: Dataset quantity (subsection 5.3.2), color thresholding imperfections (subsection 5.3.2) and masking failures (subsection 5.3.2). We also discussed how blur detection could be improved as

well when a failure prediction analysis was done (subsection 5.3.2). We additionally talk about how some of our test scenarios failed to increase in accuracy using IPS in subsection 5.3.2.

Dataset Quantity

The amount of images that we were able to gather depended on time and the quality of the images that were taken. Our test scenarios therefore contained unequal amounts of data. To help account for this inequality, for scenarios where more than 21 data points were present, 50 random shuffles were done and the mean of the multiple random shuffle scenarios were used (explained more in subsection 3.3.2). Increasing the number of data points in each test scenario would further substantiate our results.

The Angles of View Testing Scenarios (ASA & SSA)

We saw that we failed for two scenarios, especially for the Angles of View with AS weights (ASA) scenario. ASA tested an angle-of-view that was not present in the training dataset. However, our weights did not account for scenes that were not present in the training dataset. We suspect that this led to lower accuracy across this testing scenario for each metric, especially for its risk-averse chart, as shown in Figure 4.17(e). This gap is recognized and thus understood as something that could be improved in the future with an improved validation dataset that included scenarios (with all angles and light directions present) that were not present in the training dataset.

Color Thresholding

As stated before, our color thresholding failed sometimes during testing because of lighting changes and the inability to use a color checker for real-time testing. This threshold resulted in the most common issue, where purple and red could not be in the same scene or else they would combine into one label. This error resulted in some scenes being removed from the IPS testing dataset. However, this issue is a great example of how lighting changes can effect performance. The drawback in this case is that color thresholding was separate from the PI system. This means that failure from color thresholding was not considered in this research because it was not

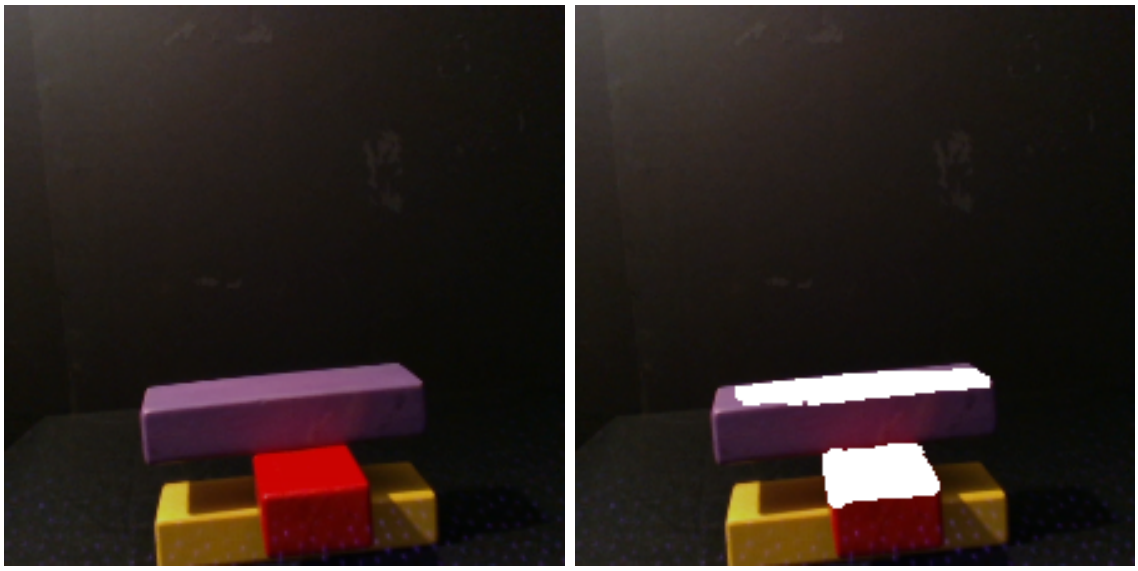


Figure 5.4: Example of IPS test image where purple and red masking failed (removed from the final IPS testing dataset): Input (left), Output masking (right).

the focused perception task. The assumption of our IPS model implementation that failure was only dependent on the final perception task and not on the subparts was wrong. Figure 5.4 displays an example of a red and purple masking image that was confused. This image was removed from our IPS dataset.

Object Masking Failures

Because of problems with color thresholding (described in subsection 5.3.2), masking failures happened. Even if the PI system was able to give a prediction, the initial masking failed such that the PI system would also fail. One major component of this failure that we observed was when blur was present in the scene. For example, one motion-blurred image had two objects in the scene (blue and green), but our color thresholding was only able to threshold one object (blue). We did not add objects that were not masked (unless it was by underexposure blur) to our dataset evaluation. Underexposure images usually failed to be masked because of the color thresholds. This failure was expected and thus each scene that failed to mask any blocks was considered as a single datapoint that PI failed on.

Blur Relations to Failure Prediction

For blur, we were able to get great results for detection and classification. However, one major drawback was that we created a blur detection and classification that was removed from the failure analysis. We have shown in the results section 4.3 that blur results did relate to overall failure prediction, but treating both tasks separately lead to worse results in some cases. We see this because blur in relation to IPS failure prediction was not analyzed. Our results show promise when looking at the underexposure (UB) scenario. If we detected underexposure, the result would usually lead to failure in creating masks. Our perception task would not execute without mask creation. Without blur detection for underexposed scenes, masking would fail and we would not have this identification of underexposure to understand the reason.

5.3.3 Future Work

The quantity of the IPS dataset could be expanded and gathered to be more equal across the scenarios. Adding focus blur as a testing scenario would also give more accurate failure measurements if focus is a point of failure. For our perception task, focus was not an issue because our camera had a large depth of focus. Blur detection and classification needs to have its own failure prediction network that could use the blur-segmented image and give a failure or success prediction. For improving angles-of-view failure prediction, adding scenarios (with all angles of view and light directions present) to the testing dataset that are not present in the training dataset would increase validation accuracy and thus lead to better testing results. Color thresholding and object masking could be better tuned to the scene, or a failure prediction network connected to specifically color thresholding and masking performance would predict if masking or color thresholding failed. In that case, we could predict failure even if color thresholding failed. Failures that were not caused by one of the three categories that we were trying to detect were not accounted for in this system, including occlusion or color thresholding errors. Accounting for these errors would increase IPS accuracy. Adding a correction of failure network after a detection of failure that was tuned to the failure prediction would also be another logical step for this research.

Chapter 6

Conclusion

We succeeded in creating a IPS network that could identify three types of perception failures: blur, light direction, and angle-of-view. We did this by combining two subsystems into one. The subsystems included (a) a blur detection and classification system that identified over- and underexposure as blur and (b) a failure prediction network that could predict failure based on light direction and angle-of-view.

6.1 Blur Detection & Classification

We presented several contributions towards equipping robots with the ability to correctly adapt to blur-induced perception failures. We (a) created definitions for over- and underexposure blur that allowed for identification in a neural network; (b) created a dataset and training conditions that enabled a network to detect overexposure, underexposure, focus, and motion blur; (c) evaluated its performance on real and synthetic data; and (d) adapted this network to identify failures for a real robot perception task. Our work will enable robots to take more appropriate actions to address poor perception than are allowed by current blur detection and classification methods.

6.2 Light Direction & Angle-of-View Failure Prediction

We presented additional contributions towards equipping robots with the ability to correctly identify failures based on light direction and angle-of-view. We (a) created a dataset for a perception task that allowed for training and testing failure prediction across light directions and different angles of view, (b) created another dataset for a perception task that allowed for training and testing failure prediction across light directions and different angles of view that are not present in training, and (c) evaluated our methods across multiple network variations, which found that using a modified AlexNet network with depth gave the best test performance across all of our testing scenarios. This failure prediction network is easily implemented and can be expanded to be used for other perception tasks such as salience detection or 3D reconstruction.

6.3 Introspective Perception

We successfully combined the two subsystems that predicted blur, light direction, and angle-of-view failures into one failure prediction network for the physics intuition network. Throughout computer vision's history, researchers have been continually looking to better understand their systems, and one aspect of that is determining when and why failures happen. This research has contributed to this field by giving the solution of looking forward for the failures instead of classifying them after execution. In conclusion, our system can be easily expanded and adapted to other perception tasks where high risk is involved and lead to the use of more accurate correction and prevention techniques.

Appendix A

Terminology

A.1 Blur Detection and Classification Training & Testing Name Key

A.1.1 Training Conditions

- PHOS + CUHK, Real Blur (PC-R): Focus blur base images overlayed with synthetic motion blur objects from the CUHK dataset, and real over- and underexposed objects from over- and underexposed images in the PHOS dataset;
- PHOS + CUHK, Synthetic Blur (PC-S): Focus blur base images overlayed with synthetic motion blur objects from the CUHK dataset, and synthetic over- and underexposed blur objects created from well-exposed images in the PHOS dataset;
- PHOS + CUHK, Real and Synthetic Blur (PC-RS): Focus blur base images overlayed with synthetic motion blur objects from the CUHK dataset, and a mix of real and synthetic created over- and underexposed objects that use over- and underexposed images and well-exposed images in the PHOS dataset;
- Our Exposure + PHOS + CUHK (OE-RS): Focus blur base images overlayed with synthetic motion blur objects from the CUHK dataset, and a mix of real and synthetic over- and underexposed objects using over- and underexposed images from our recorded exposure dataset and well-exposed images from the

PHOS dataset;

- Our IPS, Real Blur (O-R): Focus blur base images overlayed with real motion blur objects from our IPS blur dataset, well-exposed objects from those type of images from our IPS blur dataset, and real over- and underexposed objects from over- and underexposed images in our IPS blur dataset;
- Our IPS, Synthetic Blur (O-S): Focus blur base images overlayed with real motion blur objects from our IPS blur dataset, well-exposed objects from those type of images from our IPS blur dataset, and synthetic over- and underexposed blur objects created from well-exposed images in our IPS blur dataset;
- Our IPS, Real and Synthetic Blur (O-RS): Focus blur base images overlayed with real motion blur objects from our IPS blur dataset, well-exposed objects from those type of images from our IPS blur dataset, and a mix of real and synthetic created over- and underexposed objects that use over- and underexposed images and well-exposed images from our IPS blur dataset.

A.1.2 Testing Conditions

- PHOS + CUHK, Real Blur (TPC-R): Focus and motion blur images from the CUHK dataset, combined with real over- and underexposed images from the PHOS dataset;
- PHOS + Our Exposure, Real Blur (TPOE-R): Real over- and underexposed images from PHOS and our recorded exposure dataset.
- Our IPS, Real Blur (TO-R): Normal, Focus, Motion, Overexposure and Underexposure blur images from our IPS dataset.

A.2 Light Direction & Angle-of-View Failure Prediction Training & Testing Name Key

A.2.1 Training & Testing Dataset Split Key

- Scenario based Split - (SS): Training contains images of unique scenarios where all light directions and angles of view were included in the dataset. Testing

and evaluation contain images of scenarios where all light directions and angles of view were included in the dataset that were not included in the training dataset.

- Light Direction based Split - (LS): Training contains images of all unique scenarios where only *two* light directions were included in the dataset. Testing and evaluation contain images of scenarios where only *one* light direction was included in the dataset that was not included in the training dataset. Each training, testing and validation split included full scenes with all three angles of view.
- Angle of View based Split - (AS): Training contains images of all unique scenarios where only *two* angles of view were included in the dataset. Testing and evaluation contain images of scenarios where only *one* angle-of-view was present in the dataset that was not in the training dataset. Each training, testing and validation split included full scenes with all three light directions.

A.3 Introspective Perception (IPS) Testing Name Key

The following Test Scenarios used for IPS evaluation Key to Description:

- Motion Blur (MB): Motion Blur Images: Training Condition: O-R Blur Dataset & SS Dataset Split using run11 weights;
- Overexposure Blur (OB): Overexposure Blur Images: Training Conditions: O-R Blur Dataset & SS Dataset Split using run11 weights;
- Underexposure Blur (UB): Underexposure Blur Images: Training Condition: O-R Blur Dataset & SS Dataset Split using run11 weights;
- Angles of View with SS weights (SSA): Training Condition: O-R Blur Dataset & SS Dataset Split using run11 weights;
- Angles of View with AS weights (ASA): Training Condition: O-R Blur Dataset & AS Dataset Split using run11 weights; Testing Condition: Angle-of-view not present in training dataset

- Light Directions with SS weights (SSL): Training Condition: O-R Blur Dataset & SS Dataset Split using run11 weights;
- Light Directions with LS weights (LSL): Training Condition: O-R Blur Dataset & LS Dataset Split using run11 weights; Testing Condition: Light Direction not present in training dataset.

Appendix B

Training Hyperparameters

B.1 Light Direction & Angle-of-View Failure Prediction Training Hyperparameters Tables

The following tables are the individual variations for testing the failure prediction network for each dataset split. For some runs we saw that MSE worked better for this problem over BCE because the the labels of either 0 or 1 are equally important.

Run Unique ID	Network	Learning Rate	Optimizer	Loss Function	# of Epochs	Batch Size	Data Normalization	Weight Decay	Learning Rate Scheduler	Depth
run1	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	None	0	None	No
run2	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	0	None	No
run3	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	1e-4	None	No
run4	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	1e-4	Reduce LR On Plateau	No
run5	AlexNet	1e-5	Adam	Binary Cross Entropy	50	10	Range -1 to 1	1e-3	None	No
run6	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	None	1e-4	None	No
run7	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	None	0	None	No
run8	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	0	None	No
run9	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	1e-4	None	No
run10	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	1e-4	Reduce LR On Plateau	No
run11	AlexNet	1e-4	Adam	Mean Squared Error	100	10	None	1e-4	None	Yes

Table B.1: SS Training Scenarios

Run Unique ID	Network	Learning Rate	Optimizer	Loss Function	# of Epochs	Batch Size	Data Normalization	Weight Decay	Learning Rate Scheduler	Depth
run1	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	None	0	None	No
run2	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	0	None	No
run3	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	1e-4	None	No
run4	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	1e-4	Reduce LR On Plateau	No
run5	AlexNet	1e-5	RMSProp	Binary Cross Entropy	50	10	Range -1 to 1	1e-3	None	No
run6	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	None	1e-4	None	No
run7	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	None	0	None	No
run8	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	0	None	No
run9	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	None	1e-4	None	No
run10	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	None	1e-4	Reduce LR On Plateau	No
run11	AlexNet	1e-4	Adam	Mean Squared Error	100	10	None	1e-4	None	Yes

Table B.2: LS Training Scenarios

Run Unique ID	Network	Learning Rate	Optimizer	Loss Function	# of Epochs	Batch Size	Data Normalization	Weight Decay	Learning Rate Scheduler	Depth
run1	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	None	0	None	No
run2	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	0	None	No
run3	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	1e-4	None	No
run4	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	1e-4	Reduce LR On Plateau	No
run5	AlexNet	1e-4	Adam	Binary Cross Entropy	50	10	None	1e-4	None	No
run6	AlexNet	1e-5	RMSProp	Binary Cross Entropy	50	10	None	1e-4	None	No
run7	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	None	0	None	No
run8	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	Range -1 to 1	0	None	No
run9	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	None	1e-4	None	No
run10	ResNet	1e-4	Adam	Binary Cross Entropy	50	10	None	1e-4	Reduce LR On Plateau	No
run11	AlexNet	1e-5	RMSProp	Mean Squared Error	100	10	None	1e-4	None	Yes

Table B.3: AS Training Scenarios

Bibliography

[1] Mahmoud Afifi, Konstantinos G Derpanis, Bjorn Ommer, and Michael S Brown. Learning multi-scale photo exposure correction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9157–9167, 2021. 2.2.5, 5.1.4

[2] Sarthak Ahuja. Visual assessment for non-disruptive object extraction. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, August 2020. 2.5

[3] Sarthak Ahuja, Henny Admoni, and Aaron Steinfeld. Learning vision-based physics intuition models for non-disruptive object extraction. In *IEEE International Conference on Intelligent Robots and Systems*, pages 8161–8168. IEEE, 2020. (document), 1.2.3, 1.3, 1.4, 2.2.5, 2.4.2, 2.5, 2.6, 3.1.1, 3.2, 3.2.1, 3.2.3, 3.13

[4] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon" next-best-view" planner for 3d exploration. In *IEEE International Conference on Robotics and Automation*, pages 1462–1468. IEEE, 2016. 2.3, 2.3.1

[5] Thomas Blaschke. Object-based contextual image classification built on image segmentation. In *IEEE Workshop on Advances in Techniques for Analysis of Remotely Sensed Data*, pages 113–119. IEEE, 2003. 2.2.4

[6] Cl Connolly. The determination of next best views. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 432–435. IEEE, 1985. 2.3.1

[7] Shreyansh Daftry, Sam Zeng, J Andrew Bagnell, and Martial Hebert. Introspective perception: Learning to predict failures in vision systems. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1743–1750. IEEE, 2016. 1.2, 1.2.2, 2.2.1, 2.4, 2.4.1, 2.4.2, 3.2.3

[8] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3583–3592, 2016. 2.3, 2.3.1

[9] Shelley Duval and Robert A Wicklund. A theory of objective self awareness. 1972. 2.4

[10] Abdelrahman Eldesokey, Michael Felsberg, Karl Holmquist, and Michael Persson. Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12014–12023, 2020. 2.1, 2.1.2

[11] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR, 2016. 1.2.2, 2.1, 2.1.2

[12] Dong Guo, Yuan Cheng, Shaojie Zhuo, and Terence Sim. Correcting over-exposure in photographs. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 515–521. IEEE, 2010. 2.2.1, 2.2.2

[13] Ping Hsu and Bing-Yu Chen. Blurred image detection and classification. In *International Conference on Multimedia Modeling*, pages 277–286. Springer, 2008. 2.2.3, 2.2.4

[14] Rui Huang, Wei Feng, Mingyuan Fan, Liang Wan, and Jizhou Sun. Multiscale blur detection by learning discriminative deep features. *Neurocomputing*, 285: 154–166, 2018. 2.2.3

[15] Peng Jiang, Haibin Ling, Jingyi Yu, and Jingliang Peng. Salient region detection by ufo: Uniqueness, focusness and objectness. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1976–1983, 2013. 2.2.1

[16] Neel Joshi. *Defocus Blur*, pages 171–173. Springer US, Boston, MA, 2014. 2.2.1

[17] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4551–4560, 2019. 2.5, 3.1.1

[18] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, 30, 2017. 2.1.1

[19] Beomseok Kim, Hyeongseok Son, Seong-Jin Park, Sunghyun Cho, and Seungyong Lee. Defocus and motion blur detection with deep contextual features. In *Computer Graphics Forum*, volume 37, pages 277–288, 2018. 1.2.1, 2.2.3, 2.2.4, 3.1, 3.8, 3.1.2, 3.1.3, 3.1.3, 3.1.3, 4.1.2, 4.3, 4.4, 5.1.3

[20] Boon Tatt Koik and Haidi Ibrahim. A literature survey on blur detection algorithms for digital imaging. In *1st International Conference on Artificial Intelligence, Modelling and Simulation*, pages 272–277. IEEE, 2013. 2.2.1, 2.2.3

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification

with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012. 2.4.1, 2.4.2, 3.2.3

[22] Christopher B Kuhn, Markus Hofbauer, Sungkyu Lee, Goran Petrovic, and Eckehard Steinbach. Introspective failure prediction for semantic image segmentation. In *IEEE 23rd International Conference on Intelligent Transportation Systems*, pages 1–6. IEEE, 2020. 2.4, 2.4.1

[23] JiaYi Liang, YaJie Qin, and ZhiLiang Hong. An auto-exposure algorithm for detecting high contrast lighting conditions. In *7th International Conference on ASIC*, pages 725–728. IEEE, 2007. 2.2.2

[24] Renting Liu, Zhaorong Li, and Jiaya Jia. Image partial blur detection and classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 2.2.4

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 2.2.4

[26] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020. 1.2.2, 2.1, 2.1.2

[27] Rashid Minhas, Abdul A Mohammed, QM Jonathan Wu, and Maher A Sid-Ahmed. 3d shape from focus and depth map computation using steerable filters. In *International Conference Image Analysis and Recognition*, pages 573–583. Springer, 2009. 2.2.1

[28] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, 2012. 3.2.1

[29] José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 3, pages 314–317. IEEE, 2000. 2.2.3

[30] Richard Pito. A sensor-based solution to the" next best view" problem. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 1, pages 941–945. IEEE, 1996. 2.3, 2.3.1

[31] Thomas Porter and Tom Duff. Compositing digital images. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 253–259, 1984. 3.1.1

[32] Michael Potmesil and Indranil Chakravarty. Modeling motion blur in computer-generated images. *The ACM Special Interest Group on Computer Graphics*, 17

(3):389–399, 1983. 2.2.1

[33] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020. 3.1.1, 3.1.1, 3.1.1, 3.1.1, 3.2.1, 3.3.1

[34] Sadegh Rabiee and Joydeep Biswas. Ivoa: Introspective vision for obstacle avoidance. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1230–1235. IEEE, 2019. 1.2.2, 2.2.1, 2.4, 2.4.1

[35] Erik Reinhard, Wolfgang Heidrich, Paul Debevec, Sumanta Pattanaik, Greg Ward, and Karol Myszkowski. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010. 2.2.2

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 2.2.4

[37] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016. 2.5, 3.1.1

[38] Jianping Shi, Li Xu, and Jiaya Jia. Discriminative blur detection features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2965–2972, 2014. 2.2.5

[39] Inwook Shim, Tae-Hyun Oh, Joon-Young Lee, Jinwook Choi, Dong-Geol Choi, and In So Kweon. Gradient-based camera exposure control for outdoor mobile platforms. *IEEE Transactions on Circuits and Systems for Video Technology*, 29 (6):1569–1583, 2018. 2.2.2

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3.1.2

[41] Alvy Ray Smith. Painting tutorial notes. *Computer Graphics Lab, New York Institute of Technology*, 79:6–10, 1982. 3.1.1

[42] Bolan Su, Shijian Lu, and Chew Lim Tan. Blurred image region detection and classification. In *Proceedings of the 19th ACM International Conference on Multimedia*, pages 1397–1400, 2011. 2.2.4

[43] Xiaoli Sun, Xiujun Zhang, Wenbin Zou, and Chen Xu. Focus prior estimation for salient object detection. In *IEEE International Conference on Image Processing*, pages 1532–1536. IEEE, 2017. 2.2.1

[44] Vasillios Vonikakis, Dimitrios Chrysostomou, Rigas Kouskouridas, and Antonios

Gasteratos. A biologically inspired scale-space for illumination invariant feature detection. *Measurement Science and Technology*, 24(7):074024, 2013. 2.2.5

[45] Vassilios Vonikakis, Dimitris Chrysostomou, Rigas Kouskouridas, and Antonios Gasteratos. Improving the robustness in feature detection by local contrast enhancement. In *IEEE International Conference on Imaging Systems and Techniques Proceedings*, pages 158–163. IEEE, 2012. 2.2.5

[46] Laurana M Wong, Christophe Dumont, and Mongi A Abidi. Next-best-view algorithm for object reconstruction. In *Sensor Fusion and Decentralized Control in Robotic Systems*, volume 3523, pages 191–200. International Society for Optics and Photonics, 1998. 2.3, 2.3.1

[47] John W. Woods. Chapter 6 - image perception and sensing. In John W. Woods, editor, *Multidimensional Signal, Image, and Video Processing and Coding (Second Edition)*. Academic Press. 3.1.3

[48] Xiao Xiao, Fan Yang, and Amir Sadovnik. Msdu-net: A multi-scale dilated u-net for blur detection. *Sensors*, 21(5), 2021. 1.2.1, 2.2.3

[49] Wei Xu, Jane Mulligan, Di Xu, and Xiaoping Chen. Detecting and classifying blurred image regions. In *IEEE International Conference on Multimedia and Expo*, pages 1–6. IEEE, 2013. 2.2.4

[50] Ruomei Yan and Ling Shao. Blind image blur estimation via deep learning. *IEEE Transactions on Image Processing*, 25(4):1910–1921, 2016. 2.2.4

[51] Peng Zhang, Jiuling Wang, Ali Farhadi, Martial Hebert, and Devi Parikh. Predicting failures of vision systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3573, 2014. 1.2.2, 2.4.1, 2.4.2, 3.3.3

[52] Shaojie Zhuo and Terence Sim. Defocus map estimation from a single image. *Pattern Recognition*, 44(9):1852–1858, 2011. 2.2.3