KU LEUVEN

FACULTY OF ENGINEERING TECHNOLOGY

KU LEUVEN - BRUGES

Safety mechanisms for vision-based object localization for robot grasping

Thijs ALENS

Supervisor(s): Mathias Verbeke
Laurens Devos

Master's Thesis submitted to obtain the degree of Master of Science in discipline name specialisationprogramme Engineering Technology

Academic year 2025 - 2026

Preface

Het voorwoord vul je persoonlijk in met een appreciatie of dankbetuiging aan de mensen die je hebben bijgestaan tijdens het verwezenlijken van je masterproef en je hebben gesteund tijdens je studie.

In the preface, you write a personal account with an acknowledgement or expression of gratitude to the people who assisted you in completing your master's thesis and supported you during your studies.

Samenvatting

De (korte) samenvatting, toegankelijk voor een breed publiek, wordt in het Nederlands geschreven en bevat **maximum 3500 tekens**. Deze samenvatting moet ook verplicht opgeladen worden in KU Loket.

Summary

This (short) summary, accessible to a wide audience, is written in English and contains a **maximum of 3500 characters**. This summary must also be uploaded to KU Loket.

Keywords: Enter (more or less) five keywords.

Contents

List of Figures

List of Tables

Acronyms

AI	artificial intelligence. 2, 3, 5, 6
CV	computer vision. 2
MC	Monte Carlo. 9
NN	neural network. 7, 8, 9

Chapter 1

Introduction

1.1 Positioning

We are living in a world where automation is adopted at a rapidly increasing pace. Luggage gets sorted and autonomously transported across an airport, autonomous vacuum cleaners are being used daily, medicines are being produced with extreme cleanliness, etc. This also applies to robotics, where robots are slowly becoming more apparent in factory's and warehouses, improving efficiency, quality, and safety.

These robots need systems to operate properly, now more then ever, these systems are often based on artificial intelligence (AI). More precisely, computer vision (CV), a subdomain of AI which specialises in enabling computers to interpret and make decisions based on visual data via camera's, can be used to detect and localize objects, making it possible for the robot to process them. But also a more general AI could be used to plan the robot's actions and control its actuators to perform the desired tasks. Furthermore, some systems are trained to monitor certain processes, detect defects to finished products, or even predict maintenance needs. Its use thus does not only bring more efficiency, but also some gains in quality and enables for much more complex tasks for a robot to excecute.

These real world scenarios require the AI systems to be robust and reliable. However, AI systems often struggle when faced with situations that differ from their training data, leading to unreliable predictions. Systems should therefore be able to express their uncertainty about their predictions, enabling safety mechanisms to prevent the robot from taking actions based on uncertain predictions. Uncertainty-aware AI models are able to do this, by not only outputting a prediction, but also an uncertainty score about this prediction.

1.2 Problem statement

While an AI system has many advantages in a factory setting, it also introduces new problems that need taken care of. The good but mostly bad thing, in such a setting, with such systems is that it will always output something, even when it is not sure about its output. Meaning that when it were to be unsure about what it should output, it will not inform the user about this and continue operating as normal. This could be good, because it will sometimes be able to make the right decision even when unsure, increasing efficiency compared to a human worker which would need to inspect and learn the new situation. But it will most probably be bad, since it can do things wrong, which could lead to defects to the products, damage to the robot or its environment, or even worse, harm to humans working nearby.

One example of why a system could be unsure about its output is when it needs to infer something from data that is not seen during training. Most training data is gathered in controlled environments, where all conditions are optimal. However, when used in real-world scenarios with conditions different from the training environment, like different lighting conditions, some dust on the camera lens causing some distortion, someone turning the lights off, etc. the system could become a lot worese at its task. Other unexpected events, like a human operator placing the wrong object for the robot to pick up or a sudden movement of an already placed object, can also throw of the AI system.

1.3 Objectives

This thesis aims to adress these issues by developing and evaluating uncertainty-aware object localization models for a robot arm capable of reliably indicating when their predictions may be unreliable, particularly in challenging real-world conditions. By making the models uncertainty-aware, safety mechanisms that prevent the robot from taking actions based on uncertain predictions can be implemented, thereby reducing the risk of errors and accidents.

The primary research question is: How can we develop uncertainty-aware object localization models for a robot arm that can effectively identify and communicate uncertainty in their predictions, particularly in challenging real-world conditions?

This overarching question is further explored through the following sub-questions:

- What uncertainty aware (object localization) models already exist, and how do they perform in real-world conditions?
- How can we effectively quantify and communicate uncertainty in object localization predictions to ensure safe robot operation?

To address these questions, the following objectives are set:

- Conduct a comprehensive literature review on existing uncertainty-aware models with an emphesis on object localization.
- Create a dataset that simulates real-world conditions, including various challenges such as occlusions, lighting variations, and unexpected object placements.

- Develop, implement and evaluate uncertainty-aware object localization models for our use case.
- Design and implement safety mechanisms that utilize the uncertainty information to prevent the robot from taking actions based on uncertain predictions.

Chapter 2

Literature study

In this chapter, we discuss existing research relevant to uncertainty in AI models and how it can be mapped to real-world scenarios. First, we will elaborate on what uncertainty is and different ways it can sneak in AI models. Next, we will discuss how this uncertainty can be quantified. After that, we will dive into different existing methods to make AI models uncertainty-aware and discuss how their ideas can be applied to object detection. Following that, we will look into ways data is gathered to train and evaluate uncertainty-aware models. To close, we will briefly discuss safety mechanisms that can be implemented to prevent a robot from taking actions based on uncertain predictions.

2.1 Uncertainty in AI

There are lots of complex systems that need to be modeled and understood. These systems can be modelled using AI models, however these are not perfect and can make mistakes. While moddeling such a system and gathering data about it, there are multiple sources of uncertainty that can sneak in [1].

- The variability in real world situations
- The errors inherent to the measurement systems
- The errors in the architecture specification of the DNN
- The errors in the training procedure of the DNN
- The errors caused by unknown data

All these unsertainties can result in a model making wrong predictions. When in an academic or non safety-critical settings, these mistakes do not exist, can be mittigated or accapted. For example, the collected data will be very clean as the data gathering process is done in a controlled environment with no real variability. The systems are mostly developed and used in a controlled environment, for example a lab setting, where the lighting is always the same, there is no unexpected occlusions, etc.

However, when deploying AI models in real-world safety-critical settings, these mistakes can have severe consequences.

2.1.1 Variability in real world situations

The real world changes constantly, resulting in different settings where objects can behave / look differently. All situations should be covered sufficiently by the training set. If not, the network trained on the data can not garentee a good output for every new real world situation. A distribition shift occurs when real world situations differ from the training set. These shifts are very hard for a NN to grasp, causing its performence to change significantly.

Presume that there is a dataset created for detecting a sertain object in a factory setting at a set machine during the day. When a NN is trained on this dataset, it will be very good at detecting this object in that specific setting. However, it might happen that, because of a sudden increase in orders, the machine needs to run overtime and work during the night, moreover more machines at different locations need to help out. Because the dataset does not cover the objects at night, nor the settings of the other machines (lighting is different, shadows are cast differently, etc.), the trained NN fails to addapt to these new conditions. Therefore, its performence drops, which can lead to all kinds of problems.

2.1.2 Measurement errors

This comes down to the acquired data having some mistakes in them. These can come from limitations of the measurement devices or incorrect / inprecise labeling of the data. These mistakes can however be a way to regulize a network, however it needs to be subtle.

To continue the example of the dataset used for object detection on a machine, it could be that the camera used for the capturing of data has a low resolution. This could lead to an object, which needed to be detected, getting lost because of the lack of pixels in the image. Even more, it could lead to incorrect labeling of an object, as it is labeled as object a, when it is in fact object b. Finally it can lead to the bounding box not precisely being placed around the object.

2.1.3 Architecture specification errors

The structure of a network differs from model to model, with each model having its advantages and disadvantages. Each model will be different and thus its output will have a sertain level of unsertainty.

Suppose the factory requires an object detection model to best fit their need. There are a lot of models available, think of any CNN network, U-net, etc. Each model will have its own capabilities and thus unsertainty.

2.1.4 Training procedure errors

During the training of a model, there are much decisions that need to be made. The learning rate, the number of epochs, the way of using the data, the optimizer used, the weight initialization, etc. Each combination of parameters will result in a different model with its own errors and unsertainty.

The factory dataset can be split in different batch sizes, ordered in different ways, etc. The chosen model can be trained with these different ways of handeling data, using different parameters to optimize the chosen model. Again leading to different models who are better in sertain things and worse in other.

2.1.5 Errors by unknown data

When the model is fully trained, it can happen that, during inference, the model encounters a situation it has not seen before. This situation was not expected in the original scope of the problem, but the model will need to deal with it, causing unsertainty.

In the factory example, the model is trained for a sertain enviorment and objective: detecting sertain objects in a factory setting with variable lighting conditions and different machines. If an object that was not expected to be in the enviorment, a bird for example, enters the camera's viewpoint, it can cause the model to be unstertain on what to do, since it has never seen a bird before. This is not because the enviorment was insufficiently specified, but because of the bird being there unexpectedly.

2.2 Quantifying uncertainty

2.3 Uncertainty-aware AI models

2.3.1 Bayesian Neural Networks

2.3.2 Deep Ensemble Methods

A deep ensemble method [2] rests on the idea that a combination of (simpler) models can be trained and perform better then a single (complex) model. Not only that, but it also outputs a great uncertainty score about its predictions. A typical ensemble method uses multiple decision trees as their models, a deep ensemble method uses, as the name suggests, uses deep neural networks (NNs) as its models. Creating a deep ensemble method contains 3 main steps:

- A proper scoring rule, which is used as a way to train the models in the ensemble.

- Adversarial training, which is used to make the models in the ensemble more robust.
- The actual ensembling, which combines the outputs of the different models to create a final output and uncertainty score.

The scoring rule A scoring rule is a way to measure the quality of a model's probabilistic predictions. It should encourage the model to generate outputs which are accurate and are capable of telling when it is uncertain about its output. To make this a bit more concrete, in a setting where a model needs to predict if there is a disk in frame or not, a great scoring rule should punish the model hard if it predicts a disk to be in frame with a high probability, when in fact the groud truth states there is no disk in frame. The model should be encouraged to predict a low probability for the disk being in frame, if it is not sure if there is a disk in frame. It should result in a model which is rightfully confident about its predictions when it is sertain about its predictions, while also being able to express its unsertainty when it is not confident in its predictions.

Fortunately, for classification tasks, the loss functions, which are commonly used to train NNs, such as the softmax cross entropy loss, are proper scoring rules.

Adversarial training Adversarial training is a technique which is used to make NNs more robust against small changes who can drastically change the output of the model. Usually these changes are so small a human can not observe the difference, but the model is fooled to think it is something completely different. For example, when classifying images of disks and cubes, when an image is classified as a disk with a high probability, it could be that with a very small change in pixel value, the network all of a sudden thinks it is a cube with high confidence. This is something we want to avoid at all cost, since this big swing eliminates any notion of uncertainty the model could have had.

Adverserial training tries to fix these big swings by generating new training samples which are very close to some original data (for example a few pixels tweaked) and classified as some other output-class. Using the example, this means it gets an image of a disk, tweaks a few pixels so the model thinks it is a cube, and adds this new image to the training set with the correct label (disk). By doing this, the model learns to be more robust against these small changes, and the big swings in output are reduced.

Ensembling The final step in creating a deep ensemble method is the actual ensembling. The paper [2] suggests to create an ensemble of deep NNs to predict an output with an uncertainty score. Each model in the ensemble should be trained independently, however this does not mean they should be different model architectures. In fact, when randomly initializing the NN parameters, along with shuffeling the training data, the networks where different enough to obtain good results. This makes that no methods like bagging or boosting are needed to create diversity in the ensemble. Which means all the training data can be used for the training of the models in the ensemble, which is benefitial for the deep NNs which need more data to train properly.

To get the final output and uncertainty score, the outputs of the different models are combined. Each model in the ensemble outputs a probability distribution over the different classes. These probability distributions are then averaged to get the final output distribution. The label with the highest probability in this final distribution is taken as the final output with the uncertainty belonging to it. Lets continue the example of classifying images of disks and cubes to make this more concrete. Assuming we have an ensemble of 3 models, and they output the following probability distributions for a given input image:

- Model 1: [0.7 (disk), 0.3 (cube)]
- Model 2: [0.4 (disk), 0.6 (cube)]
- Model 3: [0.9 (disk), 0.1 (cube)]

Averaging these distributions results in the final output distribution: [0.67 (disk), 0.33 (cube)]. The final output is thus 'disk' with an uncertainty score of 0.67.

2.3.3 Monte Carlo dropout

Dropout is a regularization technique commonly used in the training of NNs to prevent overfitting. The idea is to randomly drop some neurons in the network during the training phase. This forces the model to use all the available neurons and not rely too much on a small subset of them. When the model is fully trained and ready for inference, dropout is usually turned off, using all the neurons to make predictions.

Monte Carlo (MC) dropout [3] expands on this idea by keeping dropout turned on during inference. Every time a set of neurons is dropped, the network changes a bit, which can result in different outputs for the same input. Each alteration of the network can be seen as a different model in an ensemble. By letting the model make multiple forward passes with dropout turned on, we can gather a set of different outputs for the same input with their own predicted probability distributions. These outputs can then be combined in a similar way as the deep ensemble method in Section 2.3.2 to get a final output and uncertainty score.

Reusing the example of classifying images of disks and cubes, assuming we make 3 forward passes with MC dropout, we could get the following outputs:

- Forward pass 1: [0.8 (disk), 0.2 (cube)]
- Forward pass 2: [0.5 (disk), 0.5 (cube)]
- Forward pass 3: [0.6 (disk), 0.4 (cube)]

Averaging these distributions results in the final output distribution: [0.63 (disk), 0.37 (cube)]. The final output is thus 'disk' with an uncertainty score of 0.63.

2.4 Datasets for uncertainty-aware models

2.5 Safety mechanisms for uncertain predictions

Chapter 3

Dataset

In this chapter, we will discuss the dataset created for training and evaluating the uncertainty-aware object localization models. We will outline the requirements for the dataset based on the problem statement and objectives defined in Chapter 1. Furthermore, we will describe the data collection process, including the setup and annotation used.

Chapter 4

Models for uncertainty aware object detection

In this chapter, we will discuss the different uncertainty-aware models discussed in Chapter 2 and compare their adaptability to a real-world scenario.

Chapter 5

Safety mechanisms for uncertain predictions

In this chapter, we will explore various safety mechanisms that can be implemented to turn the outputs of an uncertainty-aware object detection model into safe and reliable actions for a robot arm.

Chapter 6

Conclusion

6.1 Future work

Bibliography

[1] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu, "A survey of uncertainty in deep neural networks", *Artificial Intelligence Review*, vol. 56, no. 1, pp. 1513–1589, Oct. 2023, issn: 1573-7462. doi: 10.1007/s10462-023-10562-9. [Online]. Available: https://doi.org/10.1007/s10462-023-10562-9.

[2] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles", in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

[3] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning", in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 2016, pp. 1050–1059. [Online]. Available: https://proceedings.mlr.press/v48/gal16.html.

Chapter 7

GenAI code of conduct for students

Generative AI (GenAI) assistance tools can be used to generate text, image, code, video, music, or combinations of these. It includes typical tools like (but this list is not limited to): ChatGPT, Google Gemini, MS Copilot, Midjourney, Claude.ai, Perplexity.ai, Dall-E, etc.

Student Information

Student name: Thijs Alens

This form is related to my Master thesis.
Title Master thesis: Safety mechanisms for vision-based object localization for robot grasping
Promoter: ______________________
Daily supervisor: ______________________

Use of GenAI Assistance

Please indicate with "X" (possibly multiple times) in which way you were using GenAI:

- ☐ I did not use any GenAI assistance tool.
- ☐ I did use GenAI assistance. In this case, specify which ones (e.g., ChatGPT, . . .): ______________________

GenAI Assistance Used As/For

- ☐ MS Copilot

☐ Language assistance

☐ Search engine

☐ Literature search

☐ Short-form input assistance

☐ Generating programming code

☐ Generating new research ideas

☐ Generating blocks of text

☐ Other (specify): ________________________

Code of Conduct for Different Uses

Language Assistant for Reviewing or Improving Texts

This use is similar to using spelling and grammar check tools, and you do not have to refer to the use of GenAI in the text. Be careful:

- Using GenAI tools on texts you did not write yourself to cover up plagiarism (by paraphrasing original texts) is not allowed.

Search Engine for Information or Existing Research

This use is similar to a Google search or checking Wikipedia. If you then write your own text based on this information, you do not have to refer to the use of GenAI in the text. Be careful:

- Be aware that the output of the GenAI tool cannot be guaranteed as a 100% reliable source of information.
- The output may not be entirely correct and is limited due to the databases it uses. Knowledge evolves and may change over time, and it may be that the database of the GenAI tool is not up to date.

Literature Search

This use is comparable to a Google Scholar search. Be careful:

- Be aware that the output is restricted to the database it is built on. After this initial search, look for scientific sources and conduct your own analysis.
- GenAI tools (like ChatGPT) may output no or wrong references. As a student, you are responsible for further checking and verifying the accuracy of references.

Short-form Input Assistance

This use is similar to Google Docs powered by generative language models.

Generating Programming Code

The use of GenAI for coding should be explicitly allowed by the teacher. If used for coding, correctly mention the use of GenAI assistance and cite it.

Generating New Research Ideas

Further verify whether the idea is novel or not. It is likely that it is related to existing work, which should be referenced.

Generating Blocks of Text

Inserting blocks of text without quotes and a reference to GenAI assistance in your report or thesis is not allowed. Be careful:

- When you literally copy elements from a conversation with a GenAI tool, quote them between quotation marks and refer to them according to the specified reference style.
- Describe the use of the GenAI tool (tool name, version, date, etc.) in the method section and optionally add the full conversation as an attachment.

Other

Contact the professor of the course or the promoter of the thesis. Inform also the program director. Motivate how you comply with Article 84 of the exam regulations. Explain the use and added value of ChatGPT or another AI tool.

Further Important Guidelines and Remarks

- GenAI assistance cannot be used related to data or subjects under Non-Disclosure Agreement.
- GenAI assistance cannot be used related to sensitive or personal data due to privacy issues.
- Take a scientific and critical attitude when interacting with GenAI assistance and interpreting its output.
- As a student, you are responsible for complying with Article 84 of the exam regulations: your report or thesis should reflect your own knowledge, understanding, and skills.

Exam Regulations Article 84

"Every conduct individual students display with which they (partially) inhibit or attempt to inhibit a correct judgement of their own knowledge, understanding and/or skills or those of other students, is considered an irregularity which may result in a suitable penalty. A special type of irregularity is plagiarism, i.e., copying the work (ideas, texts, structures, designs, images, plans, codes, ...) of others or prior personal work in an exact or slightly modified way without adequately acknowledging the sources."

Additional Reading

More information about being transparent on the use of GenAI assistance and about citing and referencing GenAI can be found on the student website.

A Few Final Words

If you are uncertain whether or not you should declare your use of AI tools, discuss the matter with your teacher or promoter. It is safer to declare AI use when it is not needed than to withhold that declaration when it is required.

Finally, remember that advanced AI tools are new and that they can do things they could not do until recently. It is important to follow up on the most recent developments in AI technologies and communicate openly with your teachers, assistants, supervisors, and peers.