Universidade Federal de Santa Catarina

Centro Tecnológico

Departamento de Engenharia Elétrica e Eletrônica

Curso de Graduação em Engenharia Elétrica

APLICAÇÃO DE REDES NEURAIS CONVOLUCIONAIS PARA PREDIÇÃO DE QUALIDADE NO AGARRE DE PEÇAS AERONÁUTICAS

Luisa Torquato Niño

Florianópolis, Março/ 2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Niño, Luisa Torquato
Application of Grasp Quality Convolutional Neural Network for robotic grasping of aircraft parts : Aplicação de rede neural convolucional para predição de qualidade no agarre robótico de peças aeronáuticas / Luisa Torquato Niño ; orientador, Herberth Birck Fröhlich, orientador, Kilian Geiger, coorientador, Joceli Mayer, 2022.
103 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Elétrica, Florianópolis, 2022.

Inclui referências.

1. Engenharia Elétrica. 2. automated grasping systems. 3. object detection. 4. convolutional neural network. 5. computer vision. I. Fröhlich, Herberth Birck. II. Geiger, Kilian . III. Mayer, Joceli. IV. Universidade Federal de Santa Catarina. Graduação em Engenharia Elétrica. V. Título.

Luisa Torquato Niño

APLICAÇÃO DE REDES NEURAIS CONVOLUCIONAIS PARA PREDIÇÃO DE QUALIDADE NO AGARRE DE PEÇAS AERONÁUTICAS

Trabalho de Conclusão do Curso de Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Herberth Birck Fröhlich, Dr. Eng.

Orientador no instituto: Kilian Geiger, M.Sc.

Coorientador: Prof. Joceli Mayer, Dr. Eng.

Florianópolis

2022

Luisa Torquato Niño

APPLICATION OF GRASP QUALITY CONVOLUTIONAL NEURAL NETWORK FOR ROBOTIC GRASPING OF AIRCRAFT PARTS

This course completion work was judged in the context of the discipline *EEL7890: Trabalho de Conclusão de Curso* and approved in its final form by the Electrical Engineering Course.

Florianópolis, 25th March, 2022.

Prof. Jean Vianei Leite, Dr. Eng.
Electrical Engineering Department Coordinator

Appraising Committee:

Prof. Marcio Holsbach Costa, Dr. Eng.
JFSC

Prof. Henrique Simas, Dr. Eng.
UFSC

Herberth Birck Fröhlich, Dr. Eng.
Advisor
Dell Technologies

Florianópolis
2022

RESUMO

Agarre automatizado de objetos por robôs têm como um de seus principais desafios a necessidade de agarrar seguramente uma grande variedade de objetos. Para vencer esse desafio, bases de dados grandes são necessárias, especialmente em sistemas que adotam modelos de aprendizagem profunda. Entretanto, gerar esses dados com medições demanda tempo e dinheiro. Pesquisas atuais mostram que é possível agarrar uma grande variedade de objetos com alta precisão utilizando redes neurais convolucionais treinadas em dados sintéticos. Assim, este trabalho apresenta a implementação e análise da capacidade de generalização de uma rede convolucional pré-treinada em dados sintéticos, a Grasp Quality Convolutional Neural Network. Essa implementação também passa pelo sistema de aquisição de nuvens de pontos e imagens coloridas através de uma câmera RGB-D, o processamento desses dados, e a avaliação da qualidade das predições de agarre. Foi feita uma comparação entre a predição utilizando somente a nuvem de pontos e outra acompanhada de uma imagem de segmentação binária, constatando-se que esta última auxilia na fase de alinhamento para o reconhecimento do objeto e consequentemente impacta positivamente nos resultados obtidos. Apesar de ser testada em objetos com geometrias distintas daquelas vistas durante seu treinamento, a rede neural conseguiu generalizar e obteve 41.46% de predições de posição de agarre corretas utilizando como entrada nuvem de pontos e segmentação binária, o que é considerado um resultado positivo dada as milhares de possibilidades para escolha de posição de agarre.

Palavras-chave: agarre automatizado por robôs, detecção de objetos, rede neural convolucional, visão computacional, dados sintéticos, peças aeronáuticas.

ABSTRACT

Robotic grasping of objects faces the challenge of reliably grasping a wide variety of objects. In order to overcome this, large datasets are required, especially in systems that adopt deep learning models. However, generating such data with measurements demands time and capital. Recent research shows that it is possible to grasp a wide variety of objects with high accuracy using Convolutional Neural Networks trained on synthetic data. Therefore, this work presents the deployment and analysis of the generalization ability of a convolutional network pre-trained on synthetic data, the Grasp Quality Convolutional Neural Network. This deployment also goes through the acquisition system of point clouds and colour images using an RGB-D camera, the processing of this data, and the quality evaluation of the grasp predictions. A comparison between the prediction using only the cloud of points and another one accompanied by a binary segmentation image was made, finding that the latter helps in the alignment phase for the recognition of the object and consequently impacts positively on the results obtained. Despite being tested on objects with different geometries from those seen during its training, the neural network was able to generalize and obtain 41.46% of correct grasp position predictions using point clouds and binary segmentation images as input, which can be regarded as a positive result, given the amount of possible positions for grasping an object.

Keywords: automated grasping systems, object detection, convolutional neural network, computer vision, synthetic data, aeronautical parts.

SUMÁRIO

1 INTRODUÇÃO

Este trabalho foi desenvolvido a fim de atender à necessidade de um método de agarre automatizado capaz de manipular peças aeronáuticas em uma cadeia de manufatura aditiva. As atividades foram parte do projeto IDEA de Industrialização da Engenharia Digital e Manufatura Aditiva na cadeira de Metrologia de Produção e Gestão da Qualidade, departamento de pesquisa de Sistemas Baseados em Modelos do Werkzeugmaschinenlabor (WZL).

Para este fim, foi realizado um estudo sobre os métodos de planejamento de agarre de objetos para robôs. Considerando a complexa geometria das peças aeronáuticas e que a maioria destes projetos são treinados em conjuntos de dados com objetos comuns do cotidiano, tais como utensílios de cozinha, foi selecionada uma metodologia que poderia generalizar bem entre classes de objetos. Portanto, foi adotado o modelo de Aprendizagem Profunda desenvolvido por Mahler et al. em Dex-Net 4.0 [1], chamado Grasp Quality Neural Network (GQ-CNN), que utiliza uma rede neural convolucional para predizer uma posição de agarre robusta.

Além disso, para poder comparar adequadamente o desempenho da rede, foi realizada a configuração do sistema de modo semelhante à feita em Dex-Net 4.0, com sensor RGB-D, garra de robô e seu posicionamento.

Técnicas de processamento de sinais foram empregadas para realizar a aquisição de dados, aplicadas tanto em mapas de profundidade quanto em imagens coloridas e, para estas últimas, foram utilizadas técnicas de segmentação binária necessárias na fase de localização do objeto para a rede neural.

Os dados processados foram então utilizados para alimentar a rede neural pré-treinada GQ-CNN. Em seguida, para avaliar a diferença entre a predição de agarre considerando com imagem de segmentação e sem ela, foi realizado um estudo visual.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo, exposto em detalhe no apêndice em inglês, foram expostas as metodologias utilizadas para planejamento de agarre, incluindo uma introdução de redes neurais convolucionais, sistemas de aquisição, aumento e processamento de dados, além de *benchmarks* utilizadas para avaliar diferentes métodos de planejamento de agarre.

Na Seção 2.4 do documento em anexo, falou-se que a implementação de agarre robótico tem três fases: planejamento do agarre, planejamento da trajetória e execução. Este trabalho focou somente no planejamento de agarre, que é um problema de reconhecimento visual que usa sensores para achar a posição em objetos para a garra do robô. Segundo Kumra et al. [2], ele inclui tarefas como:

- Localização do objeto: alcançada através da detecção do objeto e utilizando imagens de segmentação.
- Estimação da pose: que utiliza nuvens de pontos.
- Detecção de posição de agarre: com métodos analíticos, empíricos ou híbridos, sendo este adotado pela GQ-CNN que utiliza modelos analíticos e aprendizagem profunda, e escolhe a posição de agarre com base em métrica de qualidade

A rede GQ-CNN é um método robusto de planejamento de agarre, que considera o mesmo problema do método simples, entretanto na presença de perturbações, tanto nas propriedades dos objetos, quanto mecânicas e de sensoriamento.

Enquanto na Seção 2.5, a fim de facilitar o progresso dessa ciência e a reprodução dos resultados, foram desenvolvidos *benchmarks* por Mahler et al. [3], com os quais é possível comparar diferentes sistemas baseado nos códigos desenvolvidos e variação na adoção de protocolos, sensores, luzes, objetos, braços e garras de robôs.

A primeira delas se chama média de agarres por hora, em inglês *Mean Picks Per Hour* (MPPH). Ela é definida como o produto entre a velocidade e a taxa de sucesso. A velocidade é o inverso da soma das médias de tempo de sensoriamento, cálculo computacional e movimento do robô. Enquanto a taxa de sucesso é a média da confiabilidade do agarre. Outra métrica utilizada é a de precisão, que calcula o quanto o modelo consegue classificar corretamente seus resultados como positivos. Também se avalia a sensibilidade, ou seja, a fração de predições corretas dentre todas as avaliações positivas. A partir

destas duas últimas métricas é possível obter a curva de precisão versus sensibilidade. Um método de classificação preciso mantém tanto precisão quanto sensibilidade perto de 1. Por último há a acurácia, que é a porcentagem de predições corretas dentre todas as predições.

Na Seção 2.6 do documento em anexo, falou-se sobre métodos robustos de planejamento de agarre. Considerando essa métricas, foi utilizada uma avaliação feita em Dex-Net 4.0 [1] com os seguintes métodos:

- Dex-Net 2.0 [4]: que estima a probabilidade de sucesso de agarres considerando pinças paralelas, classificando posições candidatas para agarre a partir de nuvem de pontos e a rede GQ-CNN.
- Dex-Net 3.0 [5]: semelhante a Dex-Net 2.0, entretanto considerando uma garra de sucção.
- Dex-Net 4.0 [1]: um sistema ambidestro, que pode utilizar tanto pinças paralelas, quanto garras de sucção, e é robusto porque considera perturbações no sistema.
- Método analítico para sucção: que classifica posições de agarre baseado na distância do ponto central do objeto.
- Método analítico ambidestro: além do método de sucção, utiliza pontos antipodais ao centro do objeto para agarre por pinça paralela

Para o teste dos métodos foram utilizadas duas bases de dados distintas. A primeira se refere a objetos prismáticos e circulares. A segunda se refere a objetos de uso doméstico. Ao comparar os resultados, utilizando as métricas apresentadas, foi possível perceber que a confiabilidade, ou seja a probabilidade de sucesso do agarre, e a precisão é muito maior para a Dex-Net 4.0. Além disso, ela apresentou o menor número de falhas. Portanto ela e sua versão da rede neural GQ-CNN foram selecionadas como método de planejamento de agarre.

Em Dex-Net 4.0, a rede GQ-CNN foi treinada separadamente para cada tipo de garra, paralela e de sucção. Como a princípio só serão utilizadas pinças paralelas no projeto IDEA, somente esta rede neural foi explicada em detalhe.

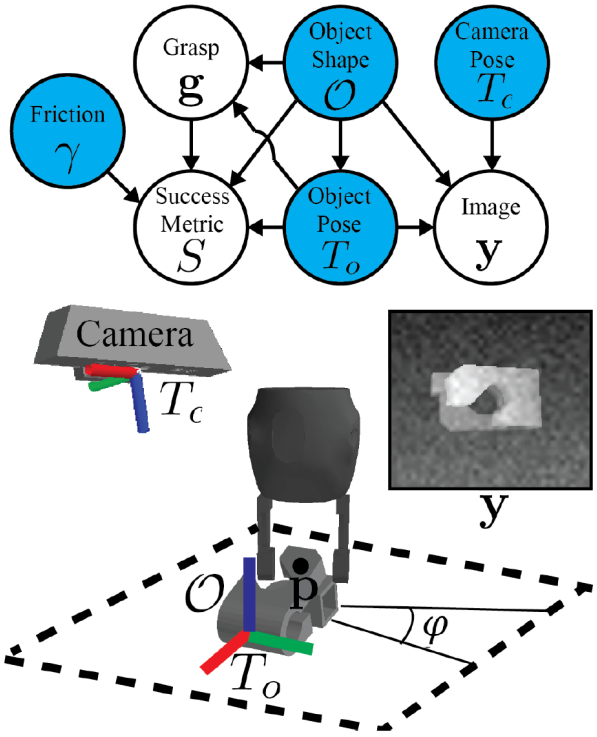
Então, na Seção 2.7 explicou-se em detalhe a rede *Grasp Quality Convolutional Neural Network* (GQ-CNN). A rede GQ-CNN recebe dados obtidos a partir de uma

câmera RGB-D, sendo eles uma nuvem de pontos e os dados intrinsecos da câmera. Opcionalmente é possível utilizar uma imagem de segmentação binária obtida a partir da imagem RGB, sendo esta utilizada para a localização do objeto. Então ela tem como saída posições de agarre em pontos antipodais de objetos com uma probabilidade de sucesso para o agarre, considerando incerteza em sensoriamento e controle. A fim de aprender a diretriz, o método utiliza uma base de dados sintética para seu treinamento, cuja geração consiste de um ambiente de treinamento sintético e uma diretriz para aquisição de dados.

Na Subseção 2.7.1 do arquivo em anexo, foram feitas as definições da rede GQ-CNN. A fim de gerar a base de dados de treinamento, é utilizado um ambiente de simulação com uma garra de robô de hastes paralelas, modelos 3D de objetos apoiados sobre uma superfície plana e nuvens de pontos adquiridas a partir de uma câmera RGB-D virtual. Neste ambiente de simulação é possível gerar imagens, posições de agarre e métricas de sucesso para poder treinar a rede.

Na Figura Figure 1 é possível observar estes elementos inseridos no ambiente virtual, além de variáveis explicadas a seguir.

Figure 1: Modelo gráfico para planejamento robusto de agarre com garra paralela de objetos em um ambiente virtual.



Fonte: Dex-Net 2.0 [4].

As propriedades da câmera e dos objetos no ambiente virtual são descritas pelo

estado $\mathbf{x} = (\mathcal{O}, T_O, T_C, \gamma)$, onde $\mathcal{O}$ representa a geometria do objeto e massa, T_O e T_C representam as poses do objeto e da câmera e γ é o coeficiente de fricção entre objeto e garra.

Enquanto as posições de agarre são especificadas como $\mathbf{u} = (\mathbf{p}, \varphi) \in \mathbb{R}^3 \times \mathcal{S}^1$, sendo $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ o centro do objeto como indicado na figura e $\varphi \in \mathcal{S}^1$ o ângulo com relação ao plano da mesa.

As nuvens de pontos são definidas como $\mathbf{y} = \mathbb{R}_+^{H \times W}$.

A métrica de sucesso é $S(\mathbf{u}, \mathbf{x}) \in 0, 1$ e pode ser modelada como:

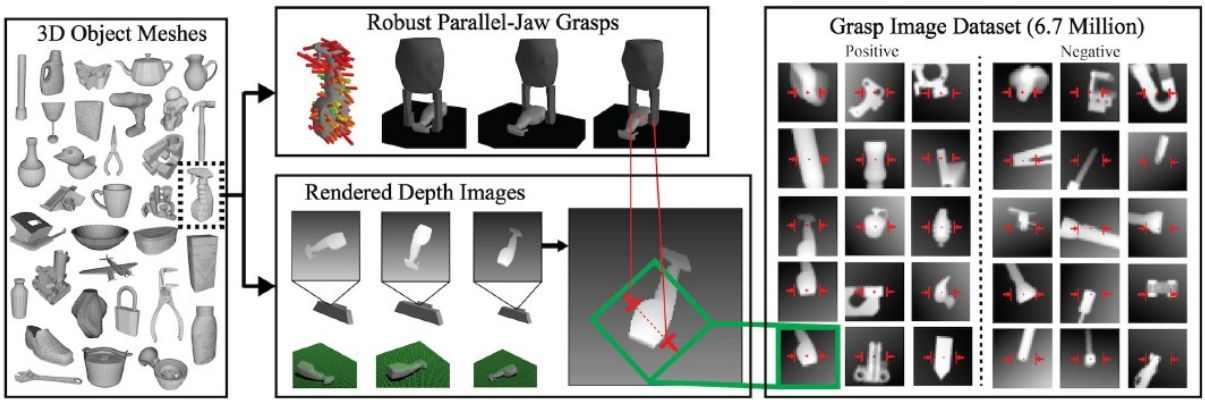
$$S(\mathbf{u}, \mathbf{x}) = \begin{cases} 1, & E_Q > \delta \ e \ collfree(\mathbf{u}, \mathbf{x}) \\ 0 \end{cases} \tag{2.1}$$

A partir dessa métrica de sucesso, é calculada a probabilidade de sucesso $Q(\mathbf{u}, \mathbf{y}) = \mathbb{E}[S|\mathbf{u}, \mathbf{y}]$ considerando incerteza no sensoriamento e controle.

A Figura Figure 2 expõe o processo de geração da base de dados para treinamento da rede neural GQ-CNN. A base de dados é garada a partir de 1500 modelos 3D de objetos, observados na parte esquerda da imagem. Para cada um deles, é calculado um conjunto de poses estáveis, expostas na imagem superior do centro. Cada uma dessas poses recebe pontos para agarre, estimados para serem livres de colisão. Dois pontos antipodais definem uma posição de agarre, cuja qualidade E_Q é avaliada, para definir a métrica de sucesso dela. Então as poses dos objetos são associadas com imagens de profundidade renderizadas a partir de uma câmera RGB-D virtual. Cada imagem passa por processo de aumento de dados, por exemplo rotacionando e transladando ela, e finalmente se obtém uma distribuição de 6.7 milhões de posições de agarre, métricas de sucesso e nuvens de pontos. A partir das posições definidas nessas imagens renderizadas, são executados agarres no ambiente virtual para gravar a recompensa do agarre. Se é possível agarrar o objeto e posicionar ele em outro lugar, a recompensa é igual a 1. Caso contrário, ela é 0. Esses valores de recompensa são utilizados como referência para treinar a rede.

Na Subseção 2.7.2, foi exposta a diretriz para aquisição de dados, que avalia as ações no ambiente de treinamento virtual descrito anteriormente, utilizando aprendizado supervisionado para treinar a diretriz π_θ a fim de maximizar a taxa de recompensa, ou média de agarres por hora MPPH, que é o produto da velocidade de agarre pela probabilidade de sucesso do agarre Q. O processo de aprendizagem da diretriz é descrito a

Figure 2: Fluxo de geração de base de dados para treinamento da rede neural.



Fonte: Dex-Net 2.0 [4].

seguir.

A rede neural é treinada para estimar a probabilidade de sucesso para uma determinada posição de agarre $Q_{\theta,g}(\mathbf{y}, \mathbf{u}) \in [0, 1]$. Os pesos da rede são otimizados de forma a minimizar o erro entre a predição da rede neural e o valor de recompensa obtido para aquela posição, como exposto na Equação 2.2. Os pesos são atualizados com retropropagação.

$$\theta_g^* = argmin_{\ \theta_g \in \Theta} \sum_{(R_i, \mathbf{u}_i, \mathbf{y}_i) \in \mathcal{D}_g} \mathcal{L}(R_i, Q_\theta(\mathbf{y}_i, \mathbf{u}_i)) \tag{2.2}$$

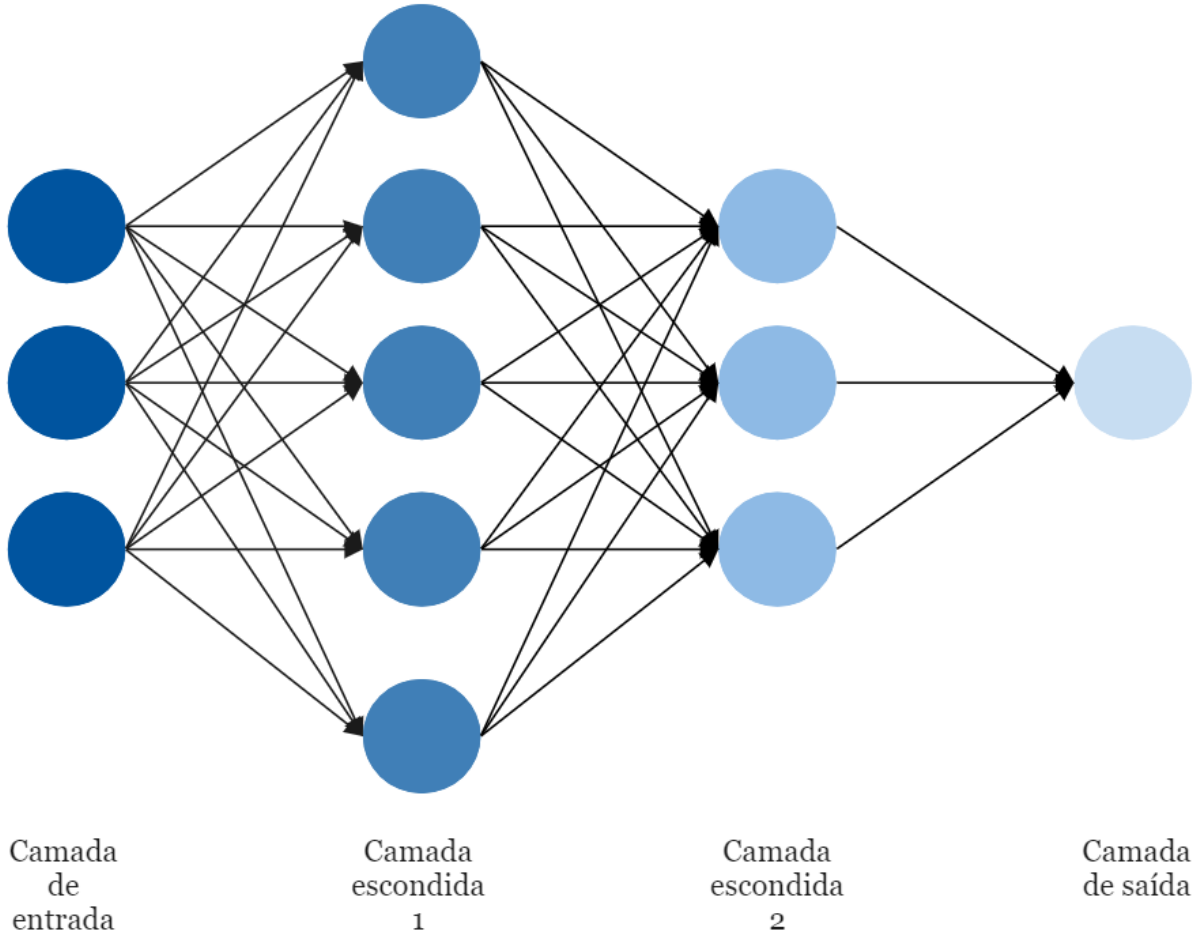
Sendo assim, a diretriz π_θ é obtida ao planejar a posição que maximiza essa métrica de qualidade considerando todas as posições candidatas ao agarre, como descrito na Equação 2.3.

$$\pi_\theta(\mathbf{y}_t) = argmax_{\ g \in \mathcal{G}} \{max_{\ u_g \in \mathcal{U}_g} Q_{\theta,g}(\mathbf{y}_t, \mathbf{u}_g)\} \tag{2.3}$$

Na Subseção 2.7.3 foi exposta a arquitetura da rede GQ-CNN. Utilizando esta base de dados é possível treinar a rede neural GQ-CNN. Como mencionado por Teuwen et al. em [6], redes neurais são uma classe de modelos que são construídos com camadas de neurônios, que são definidos pela entrada multiplicada pelo peso e somada ao viés. Exemplos de redes neurais utilizadas incluem as convolucionais (CNN) e recorrentes (RNN). As convolucionais são utilizadas principalmente para reconhecimento de padrões visuais. Elas tem configuração semelhante a exposta na Figura Figure 3, com uma camada de entrada, camadas escondidas no meio e uma camada de saída.

Funções de ativação são usadas no fim dos neurônios para introduzir não linearidade ao modelo. No caso da GQ-CNN é usada a ReLU (Rectified Linear Unit), porque ela é

Figure 3: Rede neural com uma camada de entrada, duas camadas escondidas e uma camada de saída.



Fonte: Autora

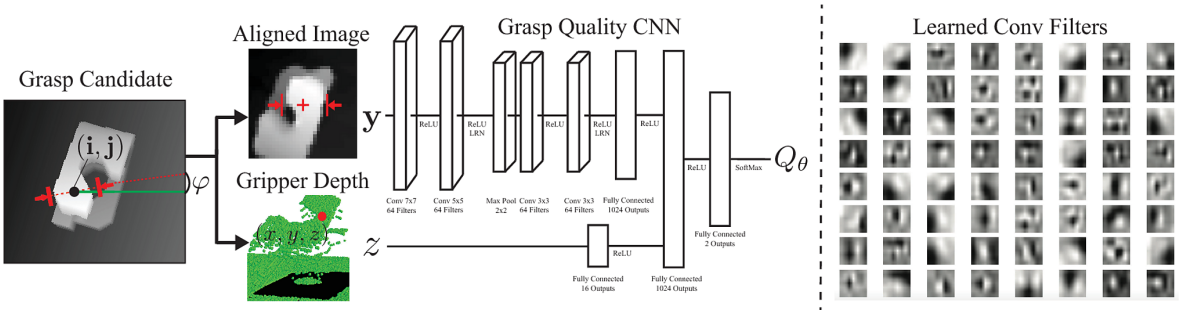
comumente utilizada em classificação de imagens e ela pode aumentar a acurácia da rede ao zerar todos os valores negativos e passar os valores positivos sem modificações, segundo Wu [7].

A rede é treinada com o objetivo de minimizar a função de custo, que avalia a performance de um modelo a partir de uma função de perda. A partir de um conjunto de dados e valores de referência, no caso o valor de recompensa, se realiza propagação para frente a fim de obter a perda entre a estimativa da rede e a referência. No caso da GQ-CNN foi utilizada Entropia Cruzada. Essa perda é retropropagada utilizando Gradiente Estocástico Descendente com momento para atualizar os pesos da rede de modo mais rápido do que o gradiente clássico. E o processo reinicia até que a rede seja completamente treinada.

A arquitetura da GQ-CNN, exposta na Figura Figure 4, tem quatro camadas

convolucionais em pares, que calcula a convolução entre a entrada e uma função de peso, resultando em um mapa de características. Entre os dois pares de camadas convolucionais, há uma camada de *Max Pooling*, que utiliza o valor máximo em uma janela de amostras para decimação. Após elas, há três camadas totalmente conectadas, na qual todos os neurônios são conectados entre si, para otimizar a rede. E também há uma camada totalmente conectada somente para a entrada z, que é a distância da garra.

Figure 4: Arquitetura da rede GQ-CNN



Fonte: Dex-Net 2.0 [4].

3 CONFIGURAÇÃO DO SISTEMA

Considerando as vantagens e desvantagens das técnicas mencionadas na seção anterior, a rede neural GQ-CNN pré-treinada e os procedimentos adotados no sistema original foram replicados na medida do possível. O modelo pré-treinado tende a generalizar bem para objetos não presentes na base de dados de treinamento, sendo assim esperava-se que fosse possível utilizá-lo para as peças aeronáuticas do projeto IDEA.

Na Seção 3.1 do arquivo em anexo em inglês, foi explicada em detalhe a aquisição de dados, resumida a seguiR. Para fazer a aquisição de dados para o teste com a rede neural, foi adotada a camêra RGB-D Intel RealSense D435, a qual conta com uma câmera infravermelho estéreo, um projetor infravermelho e uma câmera RGB. A fim de adquirir uma base de dados para teste, a câmera foi montada em uma mesa de testes, fora do robô, à 600 mm de uma superfície preta e não refletiva, apontando para o centro da mesma. A posição final da câmera será rente à garra, a fim de facilitar a calibração da câmera para o sistema de controle do robô.

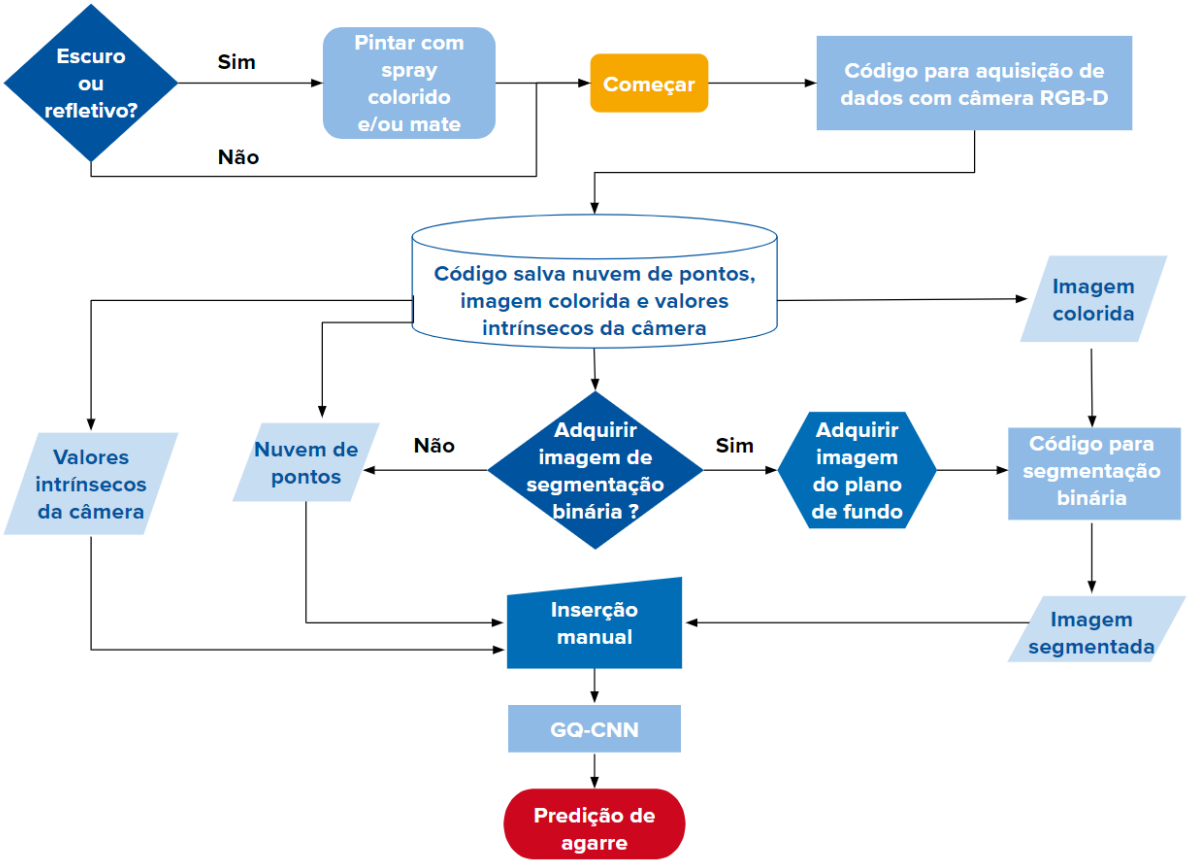
A câmera conta com um software de desenvolvimento com interface para o usuário, o qual pode ser acessado através de código com um uma biblioteca criada pelos desenvolvedores da Intel.

Na Seção 3.3 foi explicada a configuração do robô. Apesar de que não houveram testes com o robô até então, os detalhes que são relevantes para o planejamento do agarre de objetos já foram estabelecidos. Foram manufaturadas pinças paralelas com impressão 3D de PLA. Os modelos 3D delas foram baseadas no trabalho de Guo et al. em [8], que estudou variações no tamanho e textura da superfície para a ponta da garra, de forma a melhorar a aderência ao objeto. Além disso, foi impressa uma estrutura para conectar a garra, a câmera e as pinças. Foi escolhida a garra pneumática Schunk PGN Plus 50/2, que foi configurada para ter abertura máxima de 50 mm, límite definido pelos autores da GQ-CNN porque a rede foi treinada com essa configuração.

Para os testes foram utilizados objetos do projeto IDEA, os quais são peças aeronáuticas manufaturadas de forma aditiva, algumas com poliácido láctico (PLA) e outras com fusão à laser de pó de metal.

Na Seção 3.4, foi apresentado o fluxograma do sistema, a fim de ilustrar como o sistema foi implementado, exposto na Figura Figure 5.

Figure 5: Fluxograma do sistema



Fonte: Autora.

4 PROCESSAMENTO DE SINAIS

A rede neural GQ-CNN necessita como entrada uma nuvem de pontos, uma imagem de segmentação binária opcional e os dados intrísecos da câmera RGB-D. Se a imagem de segmentação não é dada, então ela é derivada da nuvem de pontos.

Na Seção 4.1 do documento em anexo, foi exposto o processamento de nuvem de pontos, e resumido aqui. O tipo de processamento muda de acordo com a finalidade. Para este trabalho, optou-se por obter nuvens de pontos com maior acurácia, do que com maior densidade de informações. A câmera RealSense produz nuvens de pontos muito ruidosas, portanto tentou-se amenizar esse ruído sem perder as informações dos objetos, com os seguintes filtros:

- Filtro espacial: utilizado para preencher buracos na nuvem de pontos a partir de uma média móvel exponencial (EMA), que considera pixels vizinhos dentro de um raio especificado.
- Filtro temporal: que utiliza média móvel também, entretanto aplicada às capturas obtidas em um certo periodo de tempo.

Enquanto que na Seção 4.2 do documento em inglês, foram expostos os filtros utilizados para obter a imagem de segmentação binária. A fim de obter uma imagem de segmentação binária foram utilizadas as seguintes técnicas nesta ordem:

- Conversão de imagem colorida para escala de cinza.
- Máscara de plano de frente: consiste na subtração da imagem de plano de fundo a partir da imagem original.
- Ajuste da escala de cinza: com definição de limite mínimo e máximo.
- Conversão para imagem binária.
- Definição de contornos: pela área mínima de pontos brancos que define um objeto e a distância mínima entre pontos brancos para definir objetos separados

5 RESULTADOS DA AQUISIÇÃO DE DADOS

Neste capítulo foram expostas imagens de profundidade e de segmentação binária obtidas com os métodos de processamento de dados explicados no capítulo de Referências Bibliográficas. Os códigos adotados para tal estão no Apêndice do arquivo em anexo.

6 RESULTADOS DA REDE GQ-CNN

Aplicando os dados obtidos como entrada para a rede neural, obteve-se a predição de posições de agarre, considerando uma garra paralela, e uma métrica de qualidade que mede a probabilidade de sucesso do agarre, denominada valor-Q.

Na Seção 6.1 do arquivo em anexo, foi feita a classificação das predições de posição de agarre de modo visual, que é necessária para prevenir possíveis acidentes caso houvesse predições inadequadas.

Os resultados foram analisados com base em valor limite de 50%, para considerar o valor-Q como positivo, e avaliação das posições indicadas como passíveis de agarre ou não. Para analisar as posições, foi considerada a geometria do objeto, o diâmetro de abertura da garra e as limitações mecânicas tanto do objeto quanto da garra.

Na Seção 6.4 do arquivo em anexo, foi feita a avaliação dos resultados. A fim de analisar a diferença da predição dada a inserção de uma imagem de segmentação binária, foram executadas 164 predições, metade delas com a imagem de segmentação, e a outra metade sem ela. Assim é possível determinar como este método contribui na fase de alinhamento do objeto.

Ao comparar as análises, foi identificado um resultado falso positivo para predição utilizando somente a nuvem de pontos, o que é considerado crítico porque poderia causar a colisão do robô com o objeto. Enquanto a predição com segmentação teve precisão de 100%, ou seja, sem falsos positivos, além de acurácia é maior, 73,17 %. Apesar da complexidade dos objetos utilizados no teste, a posição estimada foi adequada em 41,46% das estimativas. Esse é um resultado positivo também se consideradas as múltiplas possibilidades de posições para agarre.

Visualmente foi possível perceber que os resultados somente com nuvens de pontos são de difícil avaliação visual devido à baixa qualidade da mesma. Essa baixa qualidade levou à perda de informações nas nuvens de pontos e a estimativa falso positiva. Portanto se justifica a adoção da imagem de segmentação.

Para estimar o objeto que foi mais detectado em cenas com diversos objetos, foi feita uma comparação e concluído que a turbina foi o objeto mais identificado, o que pode ser justificado pelo seu tamanho proporcionalmente maior em relação aos outros objetos. Entretanto, ao avaliar os resultados para ela, é dificil de conseguir identificar a posição

adequada para o encaixe das pinças, por isso a maioria resultou em verdadeiro negativo e 0% de precisão e sensibilidade.

Os blocos coloridos e o suporte de turbina tiveram a mesma porcentagem de detecção, e as predições com eles foram mais positivas, alcançando 100% de acurácia. Eles têm geometria um pouco mais simples e portanto mais fácil de estimar uma posição de agarre. Além disso, os blocos coloridos tem geometria similar aos objetos vistos durante treinamento da rede neural, o que justifica os resultados.

Com base nesses resultados expostos, foram identificados pontos de melhoria. Apesar da imagem de segmentação melhorar a predição, o sensor RGB-D empregado produz nuvens de pontos ruidosas e não é recomendado em projetos de pesquisa deste gênero. Acredita-se que haveria um impacto positivo nas predições se for mudada a câmera para um sensor mais preciso, como a câmera RGB-D industrial Phoxi Neo, empregada em Dex-Net 4.0 [1].

Outra limitação é o diâmetro de abertura da garra, estabelecido em 50 mm, porque é o diâmetro para o qual a rede foi treinada. Levando em consideração o tamanho dos objetos expostos, seria interessante retreinar a rede ou adotar garra de sucção para objetos maiores.

A fim de aumentar a acurácia dos resultados, ou seja, obter mais predições corretas poderiam ser adotadas as posições que ocasionaram esse resultado, ou gerar a base de dados sintética somente com os objetos do projeto e utilizar aprendizagem por transferência, porque não há como corrigir somente os falsos negativos obtidos.

7 CONCLUSÃO

Este trabalho visou desenvolver um sistema de agarre automatizado de peças aeronáuticas em uma linha de produção, para atender a uma demanda do Projeto IDEA, no instituto WZL. Dentre os métodos disponíveis para planejamento de agarre, optou-se por utilizar redes neurais. Mas foi desafiante encontrar uma rede que generalizasse bem em objetos novos, principalmente considerando peças aeronáuticas, que têm geometrias complexas e são distintas dos objetos utilizados na vida cotidiana. Escolheu-se a rede GQ-CNN e foi analisada a capacidade de generalização dela.

Para este fim, foi projetado um sistema de aquisição de dados, com a câmera RGB-D Intel RealSense D435. Assim como processamento de dados para a nuvem de pontos, com aplicação de filtro espacial e temporal, e para a imagem colorida, para obter segmentação binária, com máscara de plano de frente e detecção de contorno.

Finalmente foi feita a avaliação da rede GQ-CNN para 164 predições, de modo qualitativo, metade delas feitas apenas com a nuvem de pontos obtida com a câmera RGB-D e a outra metade também utilizando a imagem de segmentação binária. Ao comparar os resultados, houve uma melhora notável no reconhecimento do objeto ao utilizar a segmentação binária, o que se traduziu em um aumento na métrica de qualidade valor-Q obtida e posições adequadas para o agarre dos objetos. A adoção da imagem de segmentação levou a precisão de 100% na predição e 73,17% de acurácia.

A rede não foi capaz de generalizar bem como defendido por seus autores em relação à sua métrica de qualidade, o que pode ser atribuído tanto à complexidade dos objetos testados que diferem das classes vistas durante seu treinamento, quanto à falta de qualidade da nuvem de pontos obtida com a câmera utilizada.

8 FUTUROS TRABALHOS

Considerando as limitações relativas à câmera utilizada, outro modelo de sensor RGB-D, com a mesma qualidade do utilizado pelos autores do projeto GQ-CNN, poderia ser testado a fim de obter melhores nuvens de pontos e, consequentemente, melhorar a predição da rede.

Caso o GQ-CNN ainda não generalize bem para os objetos do projeto IDEA, o autor sugere usar o método apresentado por Mahler et al. [9] em Dex-Net 1.0 para construir um conjunto de dados sintéticos com os objetos do projeto IDEA e usá-lo durante transferência de aprendizado com a rede FC-GQ-CNN apresentada por Mahler et al. em [10]. Posteriormente, os resultados entre o modelo pré-treinado e aquele treinado com o conjunto de dados IDEA poderiam ser comparados.

Também seria interessante o desenvolvimento de uma integração ROS entre a câmera, a rede GQ-CNN e o robô e uma avaliação posterior dos resultados em um robô real. Finalmente, a fim de melhorar a mobilidade do robô, deveria ser implementado um sistema de comunicação sem fio para a câmera.

REFERÊNCIAS

[1] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, K. Goldberg, Learning ambidextrous robot grasping policies, Science Robotics 4 (26). arXiv:https://robotics.sciencemag.org/content/4/26/eaau4984.full.pdf, doi:10.1126/scirobotics.aau4984.
URL https://robotics.sciencemag.org/content/4/26/eaau4984

[2] S. Kumra, C. Kanan, Robotic grasp detection using deep convolutional neural networks, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2017, pp. 769–776.

[3] J. Mahler, R. Platt, A. Rodriguez, M. Ciocarlie, A. Dollar, R. Detry, M. A. Roa, H. Yanco, A. Norton, J. Falco, K. v. Wyk, E. Messina, J. , D. Morrison, M. Mason, O. Brock, L. Odhner, A. Kurenkov, M. Matl, K. Goldberg, Guest editorial open discussion of robot grasping benchmarks, protocols, and metrics, IEEE Transactions on Automation Science and Engineering 15 (4) (2018) 1440–1442. doi:10.1109/TASE.2018.2871354.

[4] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, K. Goldberg, Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics (2017). arXiv:1703.09312.

[5] J. Mahler, M. . Matl, X. Liu, A. Li, D. Gealy, K. Goldberg, Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning, Proceedings of the IEEE (2018) 1–8.

[6] J. Teuwen, N. Moriakov, Chapter 20 - convolutional neural networks, in: S. K. Zhou, D. Rueckert, G. Fichtinger (Eds.), Handbook of Medical Image Computing and Computer Assisted Intervention, The Elsevier and MICCAI Society Book Series, Academic Press, 2020, pp. 481–501. doi:https://doi.org/10.1016/B978-0-12-816176-0.00025-9.
URL https://www.sciencedirect.com/science/article/pii/B9780128161760000259

[7] J. Wu, Introduction to convolutional neural networks, National Key Lab for Novel Software Technology. Nanjing University. China 5 (23) (2017) 495.

[8] M. Guo, D. V. Gealy, J. Liang, J. Mahler, A. Goncalves, S. McKinley, J. A. Ojea, K. Goldberg, Design of parallel-jaw gripper tip surfaces for robust grasping, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 2831–2838. doi:10.1109/ICRA.2017.7989330.

[9] J. Mahler, F. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kroeger, J. Kuffner, K. Goldberg, Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a Multi-Armed Bandit model with correlated rewards, IEEE International Conference on Robotics and Automation (2016) 1957–1964.

[10] V. Satish, J. Mahler, K. Goldberg, On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks, IEEE Robotics and Automation Letters 4 (2) (2019) 1357–1364. doi:10.1109/LRA.2019.2895878.

Federal University of Santa Catarina

School of Technology

Department of Electrical and Electronics Engineering

APPENDIX: APPLICATION OF GRASP QUALITY CONVOLUTIONAL NEURAL NETWORK FOR ROBOTIC GRASPING OF AIRCRAFT PARTS

Luisa Torquato Niño

Florianópolis, March/ 2022

Luisa Torquato Niño

APPLICATION OF GRASP QUALITY CONVOLUTIONAL NEURAL NETWORK FOR ROBOTIC GRASPING OF AIRCRAFT PARTS

Course completion work submitted to the Federal University of Santa Catarina as a requirement for approval in the compulsory discipline ***EEL7890: Trabalho de conclusão de Curso*** of the Electrical Engineering Undergraduate Course

Advisor: Herberth Birck Fröhlich, Dr. Eng.

Institute Advisor: Kilian Geiger, M.Sc.

Co-advisor: Prof. Joceli Mayer, Dr. Eng.

Florianópolis

2022

DISCLAIMER

The following thesis contains confidential data of Werkzeugmaschinenlabor WZL der RWTH Aachen and their project partners. Publications and reproductions of this thesis - even in excerpts - are not permitted without the express permission of Werkzeugmaschinenlabor WZL der RWTH Aachen. This thesis is only available to the proofreaders and the members of the examination board.

ACKNOWLEDGEMENTS

When I was told that Electrical Engineering would be challenging, I had no idea it would be that much. I have always studied hard, but to get this far, I had to change my choices, learn to say no, give up meaningful experiences and moments with my family and friends. I want to ask for my family and friends' apology for my absence, but it is about achieving my goals and reaching my purpose in life. Fortunately, I have very comprehensive people by my side, who have always supported me in the toughest moments of my degree, celebrated and shared good times with me. I am sure that I would not have been able to get this far without these people. Therefore, I dedicate this acknowledgement to them.

To my father, who taught me how to read and write in Portuguese, even though it was not his native language, and opened the doors of knowledge for me. Thank you for being the best professor I have ever had and advising me on this project, among others. To my mother, the woman who inspires me, who raised me while working, studying, and taking care of my two siblings. I hope to have as much energy and determination as you do. To my brother David, now a fellow professional, thank you for your pieces of advice and for giving me an example of a career to follow. It is a privilege to watch you excel, and I want to be like that one day too. To my sister, who co-founded my first company when I was 7 years old and participated in so many other ventures in my life. Thank you for being my first friend and for taking care of me.

In my degree, I met outstanding professors who opened doors for me, like Prof. Mauricio da Luz, to whom I am eternally grateful for believing in my potential and guiding me during my trajectory. I also would like to thank Prof. Walter Carpes, who is unanimously loved by his students and who helped me in difficult situations. Prof. Márcio Costa, one of the best professors I ever had, was always thoughtful and willing to help students.

At LabMETRO, I met my advisor Herberth Fröhlich, who supported me since the beginning of the project and always listened to me and shared experiences. I am very grateful for all the assistance you gave me. Prof. Armando Albertazzi, who believed in me and made possible the realization of my dream of going to Germany to work, I will always be grateful for that. There I also met Artur, Bernardo and Lucas, who also supported me to go to Germany and encouraged me in other moments of my career.

I would like to express my gratitude to the WZL institute for giving me the opportunity to accomplish this internship and develop myself professionally and personally. My special thank you to Dipl.-Ing. Philipp Nienheysen and Kilian Geiger, M.Sc. RWTH, for believing in my potential to develop this project, sharing their knowledge with me, and always supporting me. It was a pleasure for me to work in this excellent institution and share this experience with the research department of Model-Based Systems. I hope that my work has yielded improvements in the IDEA project and also helped other projects within the research group.

To my childhood friends, Betina, Gabriela, Victória, Luiza, Sachi, Moani, Helena, Amanda and Natália, who have always been with me at all phases of my life, you were essential for me to have got this far.

To my "sisters" Christiane, Beatriz, Mirella, Marjorie, Sara, Fernanda, Rebecca, Nayana and Ana, my engineering and lifetime friends, who shared with me the best moments in the university and put up with me grumbling a lot in the last months before finishing this work. Thank you for everything.

To my degree mates, those who have accompanied me since the beginning in the Academic and Athletics Centre, and those who have shared the nights and weekends studying, but who have also been at the best moments of my degree: Caroline, Luana, Louise, Isabela, Milena, Júlia, Maria Júlia, Eduarda, Henrique, Eduardo, Fernando, Vinicius, Rodrigo, João, Artur, Daniel, Geovani, Evandro, Pedro Paulo and Alexandre.

To my friends Pedro and Luana, who have always inspired and encouraged me to do my best every day.

To my eternal neighbours, Carolina, Maria Fernanda and Felipe, thank you for being my family in Ile de France.

To Louisa, surf, work and thesis buddy, who followed me through the craziest ideas of productivity to finish this work, but also helped to make this period lighter.

To the friends I made on my exchange and who were my family in Aachen: Stephanie, Marco, Gabriel, Marc, Gabriella, Leonardo, Caroliny, Paulo, Jânio, Zomer, Ivan, Roson, Gustavo, Igor, Ana, Elisa, Daniela, Felix and Nico.

Humanity needs practical men, who get the most out of their work, and, without forgetting the general good, safeguard their own interests. But humanity also needs dreamers, for whom the disinterested development of an enterprise is so captivating that it becomes impossible for them to devote their care to their own material profit.

Marie Curie

ABSTRACT

Robotic grasping of objects faces the challenge of reliably grasping a wide variety of objects. In order to overcome this, large datasets are required, especially in systems that adopt deep learning models. However, generating such data with measurements demands time and capital. Recent research shows that it is possible to grasp a wide variety of objects with high accuracy using Convolutional Neural Networks trained on synthetic data. Therefore, this work presents the deployment and analysis of the generalization ability of a convolutional network pre-trained on synthetic data, the Grasp Quality Convolutional Neural Network. This deployment also goes through the acquisition system of point clouds and colour images using an RGB-D camera, the processing of this data, and the quality evaluation of the grasp predictions. A comparison between the prediction using only the cloud of points and another one accompanied by a binary segmentation image was made, finding that the latter helps in the alignment phase for the recognition of the object and consequently impacts positively on the results obtained. Despite being tested on objects with different geometries from those seen during its training, the neural network was able to generalize and obtain 41.46% of correct grasp position predictions using point clouds and binary segmentation images as input, which can be regarded as a positive result, given the amount of possible positions for grasping an object.

Keywords: automated grasping systems, object detection, convolutional neural network, computer vision, synthetic data, aeronautical parts.

RESUMO

Agarre automatizado de objetos por robôs têm como um de seus principais desafios a necessidade de agarrar seguramente uma grande variedade de objetos. Para vencer esse desafio, bases de dados grandes são necessárias, especialmente em sistemas que adotam modelos de aprendizagem profunda. Entretanto, gerar esses dados com medições demanda tempo e dinheiro. Pesquisas atuais mostram que é possível agarrar uma grande variedade de objetos com alta precisão utilizando redes neurais convolucionais treinadas em dados sintéticos. Assim, este trabalho apresenta a implementação e análise da capacidade de generalização de uma rede convolucional pré-treinada em dados sintéticos, a Grasp Quality Convolutional Neural Network. Essa implementação também passa pelo sistema de aquisição de nuvens de pontos e imagens coloridas através de uma câmera RGB-D, o processamento desses dados, e a avaliação da qualidade das predições de agarre. Foi feita uma comparação entre a predição utilizando somente a nuvem de pontos e outra acompanhada de uma imagem de segmentação binária, constatando-se que esta última auxilia na fase de alinhamento para o reconhecimento do objeto e consequentemente impacta positivamente nos resultados obtidos. Apesar de ser testada em objetos com geometrias distintas daquelas vistas durante seu treinamento, a rede neural conseguiu generalizar e obteve 41.46% de predições de posição de agarre corretas utilizando como entrada nuvem de pontos e segmentação binária, o que é considerado um resultado positivo dada as milhares de possibilidades para escolha de posição de agarre.

Palavras-chave: agarre automatizado por robôs, detecção de objetos, rede neural convolucional, visão computacional, dados sintéticos, peças aeronáuticas.

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Acronyms

Symbol	Description	Units
2D	Two Dimensional	–
3D	Three Dimensional	–
AI	Artificial Intelligence	–
ANN	Artificial Neural Network	–
CAD	Computer Aided Design	–
CEM	Cross-Entropy Method	
CNN	Convolutional Neural Networks	–
D	Depth	–
Dex-Net	Dexterity Networks	–
EMA	Exponential Moving Average	–
FC-GQ-CNN	Fully Convolutional Grasp Quality Convolutional Neural Network	
FOV	Field-of-View	–
FPS	Frames Per Second	–
GCT	Grasp Computation Time	–
GG-CNN	Generative Grasping Convolutional Neural Network	–
GMM	Gaussian Mixture Model	–
GQ-CNN	Grasp Quality Convolutional Neural Network	–
GWS	Grasp Wrench Space	–

IR	Infrared	–
kPAM	Keypoint affordances for category-level robotic manipulation	–
L-PBF	metal laser powder-bed fusion	–
MPPH	Mean Picks Per Hour	–
MSE	Mean Squared Error	–
NPY	Numpy	–
PCL	Point Cloud Library	–
PJ	Parallel-Jaw	
PLA	Polylactide	–
PLY	Polygon File Format	–
POMDP	Partially Observable Markov Decision Process	–
PPV	Positive Prective Value	–
PReLU	Parametric Rectified Linear Unit	–
R-CNN	Region Convolutional Neural Networks	–
RAM	Random Access Memory	–
ReLU	Rectified Linear Unit	–
RGB	Red, Green and Blue	–
RGB-D	Red, Green, Blue and Depth	–
ROS	Robot Operating System	–
RReLU	Randomized Rectified Linear Unit	–
RWTH	Aachen University of Technology (German: *Rheinisch-Westfälische Technische Hochschule Aachen*)	–

SDK	Software Development Kit	–
STEP	Standard for the Exchange of Product Data	–
STL	Stereolithography	–
UFSC	Federal University of Santa Catarina (Portuguese: *Universidade Federal de Santa Catarina*)	–
WZL	Laboratory for Machine Tools (German: *Werkzeugmaschinenlabor*)	–

Greek Symbols

Symbol	Description	Units
α	smoothing factor	–
β	harmonization factor	–
δ_{thresh}	depth threshold	disparity
γ	friction coefficient between the object and gripper	–
μ	training dataset generation distribution	–
$\omega[n]$	kernel	–
ω	weight of a neural network	–
ϕ	activation function	–
$\pi_\theta(\mathbf{y}_t)$	grasp policy	–
θ_g	grasp policy's weights	–
φ	grasp axis orientation in the table plane	$\underline{\mathrm{o}}$
$y[n]$	feature map	–

Roman Symbols

Symbol	Description	Units
$\boldsymbol{\mathcal{S}}^1$	table plane	–

$\hat{\mathbf{y}}$	rendered depth image	pixels
$\mathbf{p}$	parallel-jaw grasp's center	pixels
$\mathbf{u}$	grasp	pixels
$\mathbf{v}$	direction along which the jaws close	–
$\mathbf{x}$	state	–
$\mathbf{y}$	point cloud	–
$\boldsymbol{b}$	bias	–
$\boldsymbol{c}_1$	contact point 1	
$\boldsymbol{c}_2$	contact point 2	
$\boldsymbol{g}_{i,k}$	grasp sample	–
$\boldsymbol{K}$	intrinsic matrix	–
$\boldsymbol{R}$	orthonormal 3D rotation matrix	–
$\boldsymbol{T}$	3D translation vector	–
$\boldsymbol{t}$	target of a neural network	–
$\boldsymbol{T}_{camera-gripper}$	transformation from camera's to gripper's coordinates	–
$\boldsymbol{T}_{camera-world}$	transformation from camera's to world's coordinates	–
$\boldsymbol{T}_{grasp-gripper}$	transformation from grasp's to gripper's coordinates	–
$\boldsymbol{T}_{grasp-world}$	transformation from grasp's to gripper's coordinates	–
$\boldsymbol{T}_{world-grasp}$	transformation from world's to grasp's coordinates	–
$\boldsymbol{W}$	neural network weights	–
$\boldsymbol{x}$	tensor and input of a neural network	–
$\mathbb{R}$	1D real space	–

$\mathbb{R}^2$	2D real space	–
$\mathbb{R}^3$	3D real space	–
$\mathbb{R}_+^{H \times W}$	2.5D point cloud space	–
$\mathbb{S}^2$	friction cone plane	–
$\mathcal{D}$	training dataset	–
$\mathcal{G}$	set of grippers	–
$\mathcal{L}$	cross entropy loss	–
$\mathcal{O}$	object's geometry and mass properties	–
$\mathcal{S}$	object surface	–
$\mathcal{U}_g$	set of grasp candidates	–
$\mathbf{R}_g$	gripper's 3D rigid position	–
$\mathbf{T}_g$	gripper's 3D pose	–
$\mathbf{t}_g$	gripper's orientation	–
c_x	principal point in the x axis	
c_y	principal point in the y axis	
$collfree(\mathbf{u}, \mathbf{x})$	function that verifies collision given a state and a parallel-jaw grasp	–
$E[\rho]$	mean picks per hour	picks/hour
E_Q	robust epsilon quality metric	–
f	frequency	Hz
f_x	focal length in the x axis	pixels
f_y	focal length in the y axis	pixels
g	gripper	–

H	height of an image	pixels
$MaxZ$	maximum distance of a camera from a target	mm
$MinZ$	minimum distance of a camera from a target	mm
$p(S, \mathbf{u}, \mathbf{x}, \mathbf{y})$	joint distribution on grasp success, grasps, states, and point clouds	–
$Q(\mathbf{u}, \mathbf{y})$	robustness of a grasp, or probability of success under uncertainty in sensing and control	%
q	success rate or mean grasp reliability	%
R	binary reward for a grasp	–
$S(\mathbf{u}, \mathbf{x})$	success metric	–
s	skew	–
S_t	Exponential Moving Average value at period t	–
$T(\boldsymbol{x})$	affine function	–
T	total grasp attempts	
t	grasp attempt	
T_C	camera's 3D pose	–
t_c	average time for computation	s
T_O	object's 3D pose	–
t_r	average time for robot motion	s
t_s	average time for sensing	s
v	grasp rate	grasp/s
W	width of an image	pixels
x_c	object x coordinate in the camera's coordinate system	m-

x_w	object x coordinate in the world's coordinate system	m
y_c	object y coordinate in the camera's coordinate system	m
y_w	object y coordinate in the world's coordinate system	m
z	gripper depth from the camera	m
z_c	object z coordinate in the camera's coordinate system	m
z_w	object z coordinate in the world's coordinate system	m

SUMMARY

1 INTRODUCTION

Robotic grasping has a long history of research that is increasing due to interest from industry, as mentioned by Mahler et al. [2]. In this field of research, the biggest challenge is the "Universal Picking", in which the ability of an industrial robot to reliably grasp a wide variety of objects for applications in assembly lines is tested in warehouses and houses. In real-life applications, objects have different materials and textures, and there might be poor lighting conditions, which can lead to errors in the object analysis or identification. To overcome high failure rates and train industrial robots for real-life applications, a large and robust dataset would be required, with different types of objects, from the same class, different workspace configurations, with changes in light conditions and background, and differences in the quality of the data acquired. This dataset can be used to train deep learning models and avoid bias, but generating this data through measurements can be time-consuming, costly and labour-intensive.

Recent results, exposed by Mahler et al. [3], suggest that it is possible to grasp a wide variety of objects with high precision using Convolutional Neural Networks (CNNs) trained on synthetic data sources, which is a method of data acquisition faster than acquiring data through measurements. Therefore, using data augmentation methods and synthetic data can be a solution for the problem of time and costs of generating a proper dataset for a Deep Learning model.

Within this scope, to develop an automatic grasping system for a production line of aircraft parts with complex geometries, this work used the techniques pointed out above, such as CNNs trained on synthetic data, to make predictions of grasping positions for robots. The specific objectives of this project will be explained below.

1.1 Objective

This work was developed to meet the need for an automated grasping method capable of manipulating additive manufactured aircraft parts in a production line. The activities were part of the IDEA project for Industrialization of Digital Engineering and Additive Manufacturing in the chair of Production Metrology and Quality Management, research department of Model-based Systems at Werkzeugmaschinenlabor (WZL).

1.1.1 Methodology

To this end, a comprehensive study was conducted on methods for calculating the grasp position of objects for robots. Considering the complex geometry of the aircraft parts and that most of these projects are trained in datasets with ordinary objects of daily life, such as kitchen utensils, a methodology that could generalize well between classes of objects was selected. Therefore, the Deep Learning model developed by Mahler et al. [1], called Grasp Quality Neural Network (GQ-CNN), was adopted, using a CNN to predict a robust grasp position.

1.1.2 Specific Objectives

- System configuration: to compare the performance of the network properly, an analysis of the configuration of sensors, end-effector and positioning required for grasping was performed.
- Signal processing techniques: employed to perform data acquisition, applied to both depth and colour images, and for the latter, techniques of binary segmentation necessary for object localization by the grasp planning method were used.
- Test with GQ-CNN: feed the pre-trained GQ-CNN neural network with the processed data.
- Visual evaluation: assess the difference between grasp prediction considering inputs with segmented images, besides the point cloud, and inputs without it.

1.1.3 Work outline

Therefore, to briefly outline this work, the chapters are presented as follows:

- Chapter 3: Background about CNNs, Data Acquisition, Benchmarks, Grasp Planning methods, Robust Grasp Planning, Grasp Quality Convolutional Neural Network and Data Processing.
- Chapter 4: System setup covering the data acquisition system, software and hardware used, the robot setup and overview of the grasping pipeline.

- Chapter 5: Code implementation of the signal processing methods presented in the background chapter.
- Chapter 6: Data acquisition results.
- Chapter 7: GQ-CNN results and evaluation of its predictions.
- A conclusion and outlook of the work developed.

2 BACKGROUND

In the following chapter, an overview of the methodologies used for grasp planning will take place, covering an introduction to CNN, data acquisition systems, data augmentation and processing methodologies and the benchmarks used to evaluate different methods of grasping planning.

2.1 Deep Learning

In the early days of artificial intelligence (AI), the field rapidly tackled complex problems for humans but straightforward for computers. The true challenge to AI proved to be solving intuitive problems, such as recognizing spoken words or faces in images.

The solution was to allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined through its relation to more straightforward concepts. A graph showing these concepts built on each other is deep and full of layers. For this reason, this approach is called Deep Learning, as mentioned by Goodfellow et al. in [4]. The Deep Learning models consist of multiple layers or stages of nonlinear information processing, and the methods can be either supervised or unsupervised learning.

It is split into three major classes:

- Deep networks for unsupervised or generative learning, which capture correlations of the observed data for pattern analysis or synthesis purposes when no information about target class labels is available.

- Deep networks for supervised learning or deep discriminative networks, which directly provide discriminative power for pattern classification, characterise class distributions based on visible data. In this case, target label data is always available.

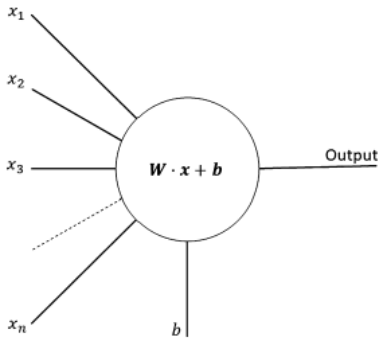
- Hybrid deep networks that unify the assisted discrimination with the outcomes of generative networks.

Deep learning is in the intersections among the research areas of artificial neural networks, artificial intelligence, graphical modelling, optimization, pattern recognition, and signal processing, as mentioned by Deng et al. in [5].

An Artificial Neural Network (ANN) consists of multiple units called neurons, connected similarly to the brain neurons. As mentioned by Teuwen et al. in [6], a neuron is composed by the input features, defined by a vector of values $\boldsymbol{x}$, weights, $\boldsymbol{W}$, and bias, $\boldsymbol{b}$. An activation function, ϕ, is applied to the neuron to provide nonlinearity, resulting in an affine function exposed in Equation 2.1. Therefore, a neuron would look like the representation in Figure 1.

$$T(\boldsymbol{x}) = \phi(\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}) \tag{2.1}$$

Figure 1: Schematic version of a neuron.



Source: [6].

The neurons are organized into layers of neurons. The output of a neuron is the input of another neuron in the next layer, therefore they are connected. This network consists of an input layer, one or more hidden layers and an output layer.

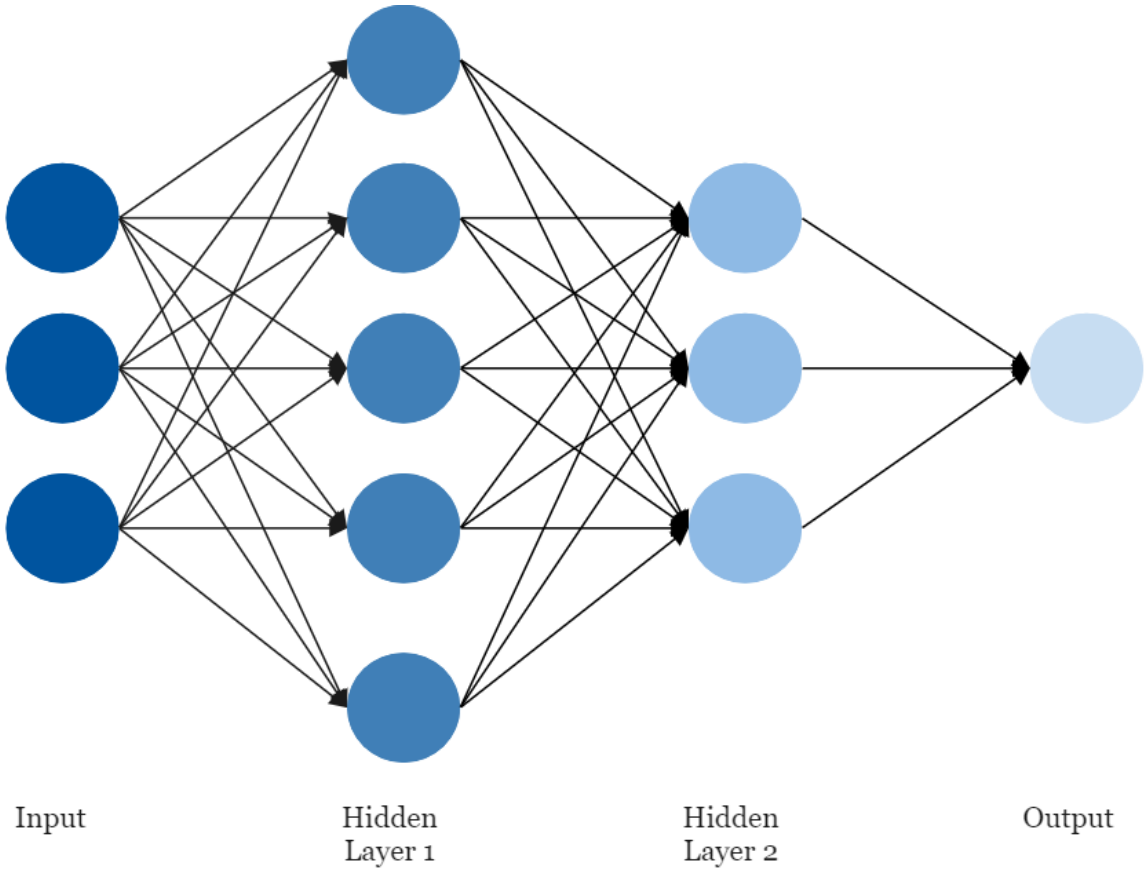
In order to use the neural network for predictions, it is necessary to find the values for the parameters $\boldsymbol{W}$ and $\boldsymbol{b}$, which is done during the training part of the ANN model. An example of a network is given in Figure 2.

2.2 Convolutional Neural Networks

CNN is a type of deep learning architecture used mainly to recognize visual patterns directly from image's pixels in tasks such as image classification, image segmentation and object detection in images. This neural network tends to outperform other pattern recognition methods, as mentioned by Kuo in [7].

As mentioned in the previous section, neural networks use affine transformations,

Figure 2: A 3-layer neural network with three inputs, two hidden layers of respectively 5 and 3 neurons, and one output layer.



Source: Author.

which can be applied to any input if the data is a vector. However, images have properties that are not used when an affine transformation is applied. A discrete convolution can substitute the affine function to preserve the image's properties. Therefore the name convolutional for this type of network comes from the mathematical operation convolution integral, used in signal processing among other engineering applications. According to Lathi [8], the convolution integral of two functions $x(t)$ and $w(t)$ is defined by:

$$y(t) = x(t) * w(t) \equiv \int_{-\infty}^{\infty} x(\tau)w(t-\tau)d\tau \tag{2.2}$$

However, the data used in deep learning is discrete. Therefore, the discrete convolution is calculated as follows:

$$y[n] = x[n] * w[n] = \sum_{m=-\infty}^{\infty} x[m]w[n-m] \tag{2.3}$$

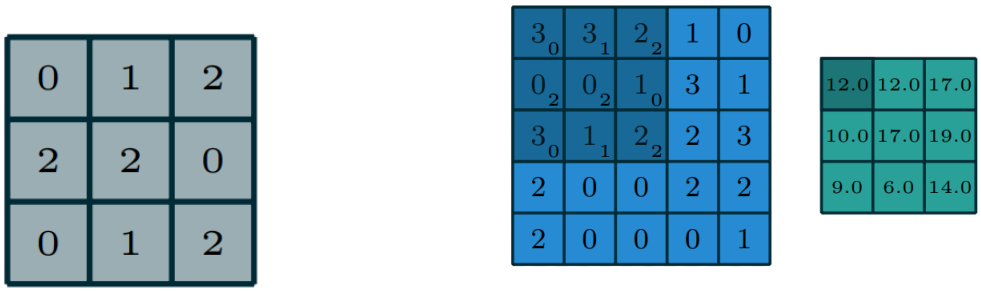
In convolutional networks, this operation averages several measurements, $x[n]$, with a weighting function, $w[n]$, also defined as kernel, which gives more weight to recent measurements. The output is a feature map, $y[n]$.

When applied to images, both input and kernel are two-dimensional. Therefore, the discrete convolution is expressed as:

$$Y[i,j] = \sum_m \sum_n X[m,n]W[i-m, j-n] \tag{2.4}$$

An example of a discrete convolution is given in Figure 3 to illustrate how it works. This example represents a single input feature map in which a kernel slides through. The product between kernel elements and overlapped input elements is computed in each location, and the results are added up to obtain the output in the current location.

Figure 3: Computing the output values of a discrete convolution: a) Kernel; b) Feature map in blue with the kernel (shaded area) and the output feature map in green.



(a) (b)

Source: [9].

Sequence 2.5 shows how a CNN runs.

$$x^1 \rightarrow w^1 \rightarrow x^2 \rightarrow ... \rightarrow x^{L-1} \rightarrow w^{L-1} \rightarrow x^L \rightarrow w^L \rightarrow z \tag{2.5}$$

The network's input is the tensor x^1, a tensor of order three in the case of a three-channel image. Following, w^1 represents the weights from the first layer, also represented by a tensor. The output of the first layer is x^2, which is also the input for the second layer. The process forwards until the layer for Backward Error Propagation, where a cost or loss function measures the discrepancy between the CNN prediction, x^L and a target,

ground truth, denoted by $\boldsymbol{t}$, to adjust the weights and minimize the cost function. Both $\boldsymbol{t}$ and x^L are probability mass functions.

A commonly used strategy is to output x^L as a C dimensional vector whose i-th entry encodes the prediction. The processing in the $(L-1)$-th layer can be set as a Softmax regression of x^{L-1}, to make x^L a probability mass function.

The Softmax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1, as mentioned by Ng et al. in [10].

Other aspects of CNNs may vary between algorithms. Therefore, they will be explained in more detail in Section 2.7.

2.3 Data Acquisition

As described in the book "The signal and the noise", from Nate Silver [11], a signal is the actual pattern aimed to learn from data, while noise refers to the irrelevant information or randomness in a dataset. Usually, when the dataset is large, it is easier to find patterns and relationships between the parameters and their values or signal. Machine and Deep Learning helps us find those patterns in data, used to make predictions about new data. The dataset must contain high quality and well-transformed data to get those predictions right, considering that inaccurate data may cause the model to learn wrong patterns. Accordingly, data acquisition and processing matter more than the model in use.

This section will introduce measurement systems for data acquisition and synthetic datasets applied to robust grasp planning, methods that use Deep Learning to detect grasps. Further details about robust grasp planning will be explained in a separate section afterwards.

2.3.1 Measurement Systems

RGB-D cameras are widely used in object detection, 3D mapping and location, path planning, autonomous navigation and people tracking, among other applications. This measurement system provides colours of objects, red, green and blue (RGB), with their related distance (D) from the sensor on a per-pixel basis. It was used, for example,

in robust grasp planning methods by Mahler et al. [12] and Manuelli et al. [13].

2.3.2 Synthetic Datasets

State-of-the-art deep neural networks require large datasets for their training. However, acquiring a dataset large enough to train the neural network from the beginning requires extensive acquisition time in measurements. In order to reduce the time and cost spent on data collection for these models, traditional methods of data augmentation were adopted in renowned CNN models, like AlexNet [14]. The techniques used in their work consisted of image translations, horizontal reflections, and patch extractions.

Another data augmentation technique is synthetic data generation, also used to generate whole synthetic datasets. Currently, synthetic data is used to train autonomous cars, streamline software development, simulate clinical situations to develop medical solutions and reconstruct images based on synthetic data, as was the case of the generation of the first image of a black hole [15]. Therefore, synthetic data is crucial to developing new technologies. This data acquisition method has shown to be a good solution, but there is the need to ensure that performance generalizes well between natural and rendered scenes.

2.3.3 Dex-Net

Dexterity Network (Dex-Net) [16], [12], [17] and [1] is a research project by Mahler et al. including code, datasets, and algorithms for generating datasets of synthetic point clouds, robot parallel-jaw grasps, and metrics of grasp robustness based on physics for thousands of 3D object models to train deep learning methods to plan grasps for robots. The goal of Dex-Net is to provide reliable robot grasping across a wide variety of rigid objects.

This project produced a synthetic dataset of 6.7 million point clouds and analytic grasp quality metrics with parallel-jaw grasps planned using robust quasi-static Grasp Wrench Space (GWS) [18] analysis on a dataset of 1500 3D object models in randomized poses on a table in a virtual environment.

Before explaining the dataset in detail, the definition of the variables necessary to understand it is given below.

2.3.3.1 Dataset Variables Definition

Dex-Net 4.0 considers the problem of planning a robust planar parallel-jaw grasp for a robot with an overhead depth camera and a heap of objects in a bin. Therefore, a network that takes as input a depth image and outputs a grasp position and its robustness, meaning the probability of success under uncertainty in sensing and control, needs to be trained.

In order to generate a training dataset for this network, a virtual environment with a parallel-jaw gripper, rigid objects singulated on a planar work surface, and single-view point clouds taken with a depth camera is used to generate images, grasps and success metrics. Figure 4 presents a graphical model with this setup and the dataset's variables, which will be explained below.

The variable properties of the camera and objects in the virtual environment are described by the state $\mathbf{x} = (\mathcal{O}, T_O, T_C, \gamma)$, where $\mathcal{O}$ represents the object's geometry and mass properties, T_O and T_C represent the object's and camera's 3D pose, respectively, and γ is the coefficient of friction between the object and gripper.

Parallel-jaw grasps are specified as $\mathbf{u} = (\mathbf{p}, \varphi) \in \mathbb{R}^3 \times \mathcal{S}^1$, $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ is its centre and $\varphi \in \mathcal{S}^1$ is an angle in the table plane.

Point clouds are defined as $\mathbf{y} = \mathbb{R}_+^{H \times W}$, represented as a depth image with height H and width W taken by an RGB-D camera.

The grasp's success metric is $S(\mathbf{u}, \mathbf{x}) \in 0, 1$ and is modeled as in Equation 2.6:

$$S(\mathbf{u}, \mathbf{x}) = \begin{cases} 1, & E_Q > \delta \; and \; collfree(\mathbf{u}, \mathbf{x}) \\ 0, & otherwise \end{cases} \tag{2.6}$$

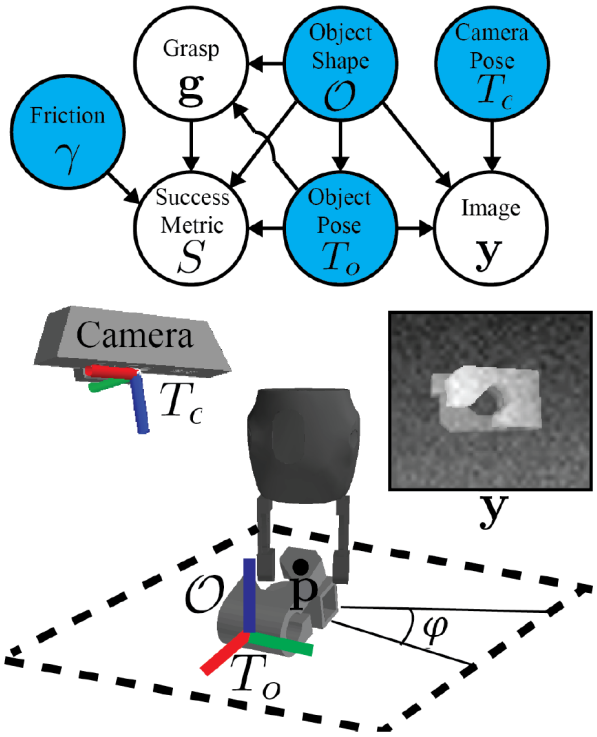
where E_Q is the robust epsilon quality metric that includes uncertainty in friction and gripper pose, and $collfree(\mathbf{u}, \mathbf{x})$ indicates if the gripper does not collide with the object or table.

Based on the success metric, a probability of success under uncertainty in sensing and control is calculated in Equation 2.7:

$$Q(\mathbf{u}, \mathbf{y}) = \mathbb{E}[S|\mathbf{u}, \mathbf{y}] \tag{2.7}$$

The graphical model illustrated in Figure 4 models a joint distribution $p(S, \mathbf{u}, \mathbf{x}, \mathbf{y})$ on grasp success, grasps, states and point clouds. The network uses this distribution to learn a robustness function, which will be explained further in Section 2.7.

Figure 4: Graphical model for robust parallel-jaw grasping of objects on a table surface based on point clouds.



Source: Dex-Net 2.0 [12].

2.3.3.2 Dataset Composition

Dex-Net's [12] dataset composition and generation is illustrated in Figure 5 and described below:

1. **3D models:** each mesh model is aligned to a standard frame of reference using the principal axes, rescaled to fit within a gripper width of 50 mm and assigned a mass of 1.0 kg centred in the object bounding box.

2. **Stable poses:** A set of stable poses for each mesh is computed with a probability of occurrence above a threshold. Each stable pose is associated with a set of collision-free grasps perpendicular to the table, considering a gripper model.

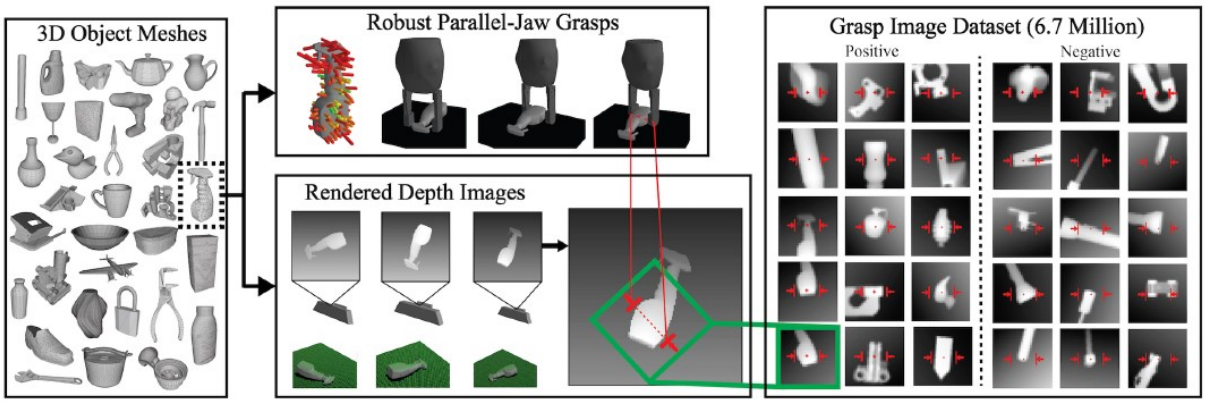
3. **Parallel-jaw grasps:** each object is labelled with a set of parallel-jaw grasps. A quality metric called Robust Epsilon Quality, E_Q, is evaluated for each grasp under object pose, gripper pose, and friction coefficient uncertainty using Monte-Carlo sampling.

4. **Rendered Depth Images:** afterwards, each object's stable pose is paired with a rendered depth image $\hat{\mathbf{y}}$. The images are rendered using a pinhole camera model and perspective projection with the camera's intrinsic values. Each image is rotated, translated, cropped, and scaled to align the grasp pixel location with the image centre and the grasp axis with the middle row of the image.

5. **Robust Analytic Grasp Metrics:** joint distribution $p(S, \mathbf{u}, \mathbf{x}, \mathbf{y})$ on grasp success, grasps, states and point clouds.

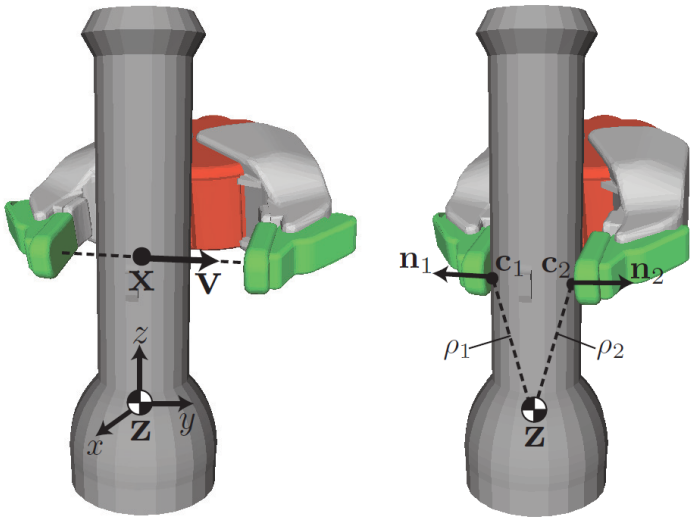
In order to find grasps, Dex-Net uses a modification of the algorithm developed by Smith et al. [19]. To sample a single grasp, it first generates a contact point $\boldsymbol{c}_1$ by sampling uniformly from the object surface $\mathcal{S}$. Next, a random direction along which the jaws close $\mathbf{v} \in \mathbb{S}^2$ is sampled uniformly from a friction cone, and $\boldsymbol{c}_2$ is computed on the line $\boldsymbol{c}_1 + t\mathbf{v}$ where $t \geq 0$. This yields a grasp $\boldsymbol{g}_{i,k} = (0,5(\boldsymbol{c}_1 + \boldsymbol{c}_2), \mathbf{v})$ which is added to the candidate set if the contacts are antipodal. The grasp parameters are illustrated in Figure 6.

Figure 5: Dex-Net 2.0 pipeline for training dataset generation. The 3D models are in the left image; at the top image, the parallel-jaw grasps are in the left and the stable poses on the right side; at the bottom, there are the point clouds rendered from the stable pose; the resulting dataset is in the right.



Source: Dex-Net 2.0 [12].

Figure 6: Grasp parameterization and contact model.



Source: Dex-Net 1.0 [16]

2.4 Grasp Planning

Kumra et al. [20] mentioned that a robotic grasping implementation has three phases: grasp planning, trajectory planning, and execution. This work will focus on the first phase, grasp planning, a visual recognition problem that uses sensors to detect graspable objects in an environment. The objective is to find a gripper position that maximizes a quality or success metric, considering the object shape and environmental factors, like the object's pose on a table or an obstacle between the robot and the object.

Grasp planning includes the tasks of object localization, pose estimation and grasp detection. Object localization is achieved using object detection and image segmentation, while pose estimation uses point clouds for its methods. Grasp detection methods are analytical, empirical or hybrid, based on their success metric.

2.4.1 Analytical Methods

Analytical methods match images to predefined 3D models. For example, the kPAM method by Manuelli et al. [13] adopts four main points of an object to represent it: a central point at the bottom, a central point at the top, an axial vector to the object and an external point to place the object. Afterwards the object representation is matched to known 3D objects.

2.4.2 Empirical Methods

On the other hand, empirical methods generally use machine learning to predict directly from the robot sensors readings to success labels from humans or physical trials. The work developed by Gualtireri et al. [21] uses deep reinforcement learning for pick-and-place and re-grasping tasks. The difference between their model and others is that the exact geometry of the objects to be handled is unknown. A downside of this method is that it has to be retrained to perform a similar task for other classes of objects and cannot work with a more diverse set of objects.

2.4.3 Hybrid Methods

The best aspect of both methods is united in hybrid approaches, which utilizes massive synthetic datasets generated with analytical models and Deep Learning. These approaches utilize policies that query a neural network to determine the highest quality grasp, and they fall into two categories, as shown in Figure 7.

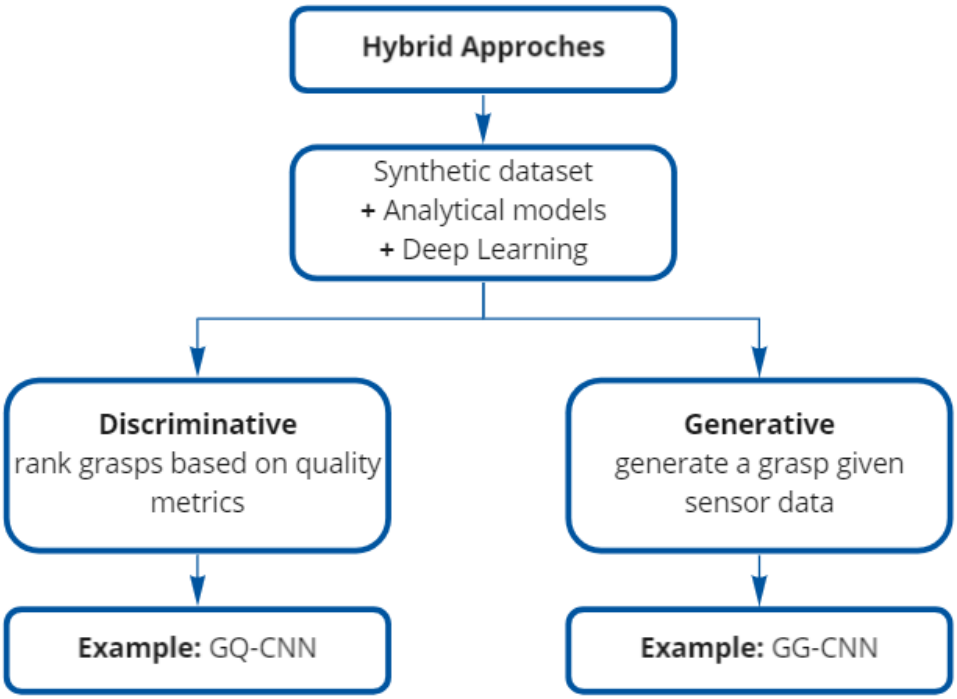
The first category is discriminative, which utilizes a neural network to rank grasps based on a quality metric and optimization techniques to search for high-quality grasp candidates, like the policy developed in Dex-Net 4.0 [1]. This approach will be explained later in this chapter.

The second one is generative, which comprises approaches that directly generate a grasp set, given sensor data, and may use analytical methods to select the optimal grasp from this set, such as Generative Grasping Convolutional Neural Network (GG-CNN) by Morrison et al. [22].

2.4.4 Grasp Evaluation

In order to evaluate grasp candidates, E_Q can be obtained through the analysis of the Grasp Wrench Space [18], which describes the force and momentum applied at the grasp points. In Dex-Net 4.0 [1], the framework evaluates all grasps according to the ability to resist random disturbing forces and torques (wrench).

Figure 7: Hybrid approaches fall into two categories: discriminative and generative.



Source: Author.

2.5 Benchmarks

Before evaluating methods available nowadays for grasp planning, an overview of benchmarks used to evaluate them is necessary. Although the benchmarks used to compare methods are still an open discussion, one that seems adequate is the one proposed by Mahler et al. in [2], used in this work.

The number of objects successfully grasped per hour is measured by Mean Picks Per Hour (MPPH). Defined as $E(\rho)$, it is the combination of speed and success rate, given by the following Equation 2.8:

$$E(\rho) = v \cdot q \tag{2.8}$$

The mean grasp, or speed, rate, is given by the following equation:

$$v = \frac{1}{t_s + t_c + t_r} \tag{2.9}$$

where t_s, t_c and t_r are the average times for sensing, computation, and robot motion, respectively, in seconds. The average time for computation is referred to as grasp com-

putation time (GCT).

The measure of MPPH also depends on the Success Rate (q), which is the mean grasp reliability, or probability that each grasp attempt is successful.

Another metric used to evaluate different methods for grasp planning is the Precision-Recall curve. This curve is determined by plotting the precision against the recall for a single classifier considering a threshold, defined as the predicted probability of an observation belonging to the positive class. In a binary classification problem, a prediction relative to labelled data can be a true positive, false positive, true negative or false negative.

Precision estimates how well a model predicts the positive class. Therefore, it is also known as Positive Predictive Value (PPV) and, as mentioned by Cook et al. in [23], it can be calculated as follows in Equation 2.5, where TP is the number of true positives and FP is the number of false positives.

$$precision = PPV = \frac{TP}{TP + FP} \tag{2.10}$$

While recall, also known as sensitivity, is the fraction of positive predictions out of all positive instances and is represented in Equation 2.5.

$$recall = sensitivity = \frac{TP}{TP + FN} \tag{2.11}$$

An accurate classifier maintains precision and recalls at a high level, near 1.

2.6 Robust Grasp Planning

Robust grasp planning considers the same problem as grasp planning but in the presence of perturbations in the object properties, such as shape and pose, or mechanical properties, such as friction, caused by the imprecision in perception and control, as mentioned by Mahler et al. in Dex-Net 1.0 [16].

In Dex-Net 2.0 [12], Mahler et al. developed a model called Grasp Quality Convolutional Neural Network (GQ-CNN), that predicts the probability of a successful grasp by classifying grasp candidates sampled from an RGB-D image and ranking them individually using CNNs. The grasp planner receives as input a depth image, an optional

binary image and the RGB-D camera intrinsic values and outputs a grasping position for the gripper, with two antipodal points and a probability of success under uncertainty in sensing and control. The grasps are specified as a Cartesian position, an angle, and a grasp distance relative to the RGB-D sensor. It uses robust analytic grasp metrics as supervision, using the gripper distance from the camera in predictions.

Afterwards, in Dex-Net 3.0 developed by Mahler et al. [17], a compatible suction contact model was proposed which computes the quality of the seal between the suction cup and the target object and determines whether the suction gripper can resist the forces external to the object, such as those caused by gravity. A dataset of 2,8 million point clouds, suction grasp data, and grasp robustness identifiers, calculated using 1.500 3D object models, were used for training the CNN.

The resulting system was evaluated in 375 physical tests at ABB YuMi equipped with a pneumatic suction arm. The model achieved 99 % accuracy in a dataset of known objects with adverse geometries such as sharply curved surfaces.

The most recent dexterous network developed by Mahler et al. [1] is Dex-Net 4.0. It is an ambidextrous system, because it uses two end-effectors, a suction cup gripper and a parallel-jaw, and evaluates grasps with a standard metric: expected wrench resistance, or the ability to resist task-specific forces and torques, such as that caused by gravity, under random perturbations. Each grasp is planned based on a depth image from an overhead RGB-D camera, using a ray tracing algorithm for 3D reconstruction. Dex-Net 4.0 policy achieves 95% reliability on a physical robot with 300 MPPH.

In Table 1 and Table 2, Mahler et al. compared Dex-Net 4.0 alone, Dex-Net 3.0 and 2.0 composite, a heuristic for suction only and a heuristic composite for a suction cup and parallel-jaw, using bin-picking benchmark [2], for five trials on level 1 and level 2 datasets of 25 novel objects each. Level 1 refers to prismatic and circular solids. Level 2 refers to everyday household objects with varied geometry and masses up to 500 g. Each grasp was planned based on a depth image from an overhead RGB-D camera. The minimum number of grasp attempts was 125.

The heuristic suction refers to a hand-coded function for suction cups. It ranks planar grasps based on the inverse distance to the central point of an object, where the object's central point is estimated as the mean pixel of an object instance segmentation mask, obtained using Euclidean clustering segmentation algorithm from the Point Cloud

Library (PCL). Planarity was determined by evaluating the Mean Squared Error (MSE) of all 3D points within a sphere with a radius of 10 mm, based on the suction cup size, to the best-fit plane for the points. Grasps were considered planar if the MSE was less than an absolute threshold or within the top 5 % of all grasps candidates.

The heuristic compound refers to a hand-coded function to select between the suction jaws and the parallel-jaws. It classifies the planned grasp points with the suction function mentioned above and a parallel-jaws heuristic based on antipodal points, that is, two opposing points. This method classifies antipodal points based on the inverse distance to the estimated central point of an object, determining the antipodal points based on the estimated point cloud surface norms. The grasping point closest to the estimated centre of the object through both parallel-jaws is selected for execution.

Table 1: Comparison between methods using the level 1 dataset. The best results are highlighted in bold.

Level 1					
Policy	Reliability (%)	MPPH	AP (%)	No. of attempts	No. of failures
Heuristic (suction)	93	**331**	95	135	10
Heuristic (composite)	91	281	93	139	14
Dex-Net 2-3 composite	91	306	93	135	10
Dex-Net 4.0	**97**	309	**100**	**129**	**4**

Source: Dex-Net 4.0 [1].

Table 2: Comparison between methods using the level 2 dataset. The best results are highlighted in bold.

Level 2					
Policy	Reliability (%)	MPPH	AP (%)	No. of attempts	No. of failures
Heuristic (suction)	80	304	87	159	31
Heuristic (composite)	76	238	83	168	43
Dex-Net 2-3 composite	76	255	64	168	43
Dex-Net 4.0	**95**	**312**	**99**	**131**	**6**

Source: Dex-Net 4.0 [1].

From this comparison, Dex-Net 4.0 was selected for implementation in the grasping pipeline of this project.

In the recent work from Vishal [24], the deployment of a Fully Convolutional Network generated an alternate faster GQ-CNN, called Fully Convolutional Grasp Quality Convolutional Neural Network (FC-GQ-CNN). The FC-GQ-CNN outperforms the GQ-CNN, GQ-CNN(∗), and the Parallel-Jaw (PJ) heuristic in rate and reliability. With FC-GQ-CNN, it was possible to increase the number of successful grasps significantly compared to the prior policy, DexNet 4.0, based on iterative grasp sampling and evaluation, as shown in Table 3, where GQCNN(∗) is a version of GQ-CNN with an increased Cross-Entropy Method (CEM) sampling, increasing the performance of CEM at the cost of rate.

Table 3: Comparison between a parallel-jaw (PJ) heuristic, GQ-CNN, and FC-GQ-CNN on bin picking with 25 novel objects on a physical robot.

Policy	Reliability (%)	AP (%)	GCT (s)	MPPH
PJ Heuristic	53,4	77,1	2,0	162
GQ-CNN	75,8	**96,0**	1,5	250
GQ-CNN (∗)	81,2	93,8	3,0	236
FC-GQ-CNN	**85,6**	95,2	**0,6**	**296**

Source: [24].

Considering the values in Table 3, the GQ-CNN model optimization using Fully Convolutional Network would be the best option for this project. However, the computer used for this work did not have enough processing power, and the ones available at the institute that had the required processing power could not have a Linux system installed for cybersecurity reasons. Therefore, the latest version of GQ-CNN, presented in Dex-Net 4.0 [1], was used in this project.

2.7 Grasp Quality Convolutional Neural Network

In Dex-Net 4.0, GQ-CNN was trained separately for each kind of gripper, parallel-jaw and suction-cup. Considering that this project used a parallel-jaw gripper, the corresponding policy will be explained in detail in this section, from the type of data used for its training, its architecture, the results generated and how it can be employed.

In order to learn the policy, their method uses a training dataset generation distribution μ, which consists of two stochastic components:

- **Synthetic training environment:** similar to the one previously exposed for Dex-Net 2.0 in Subsection 2.3.3, with the addition of a binary reward label, evaluated according to the ability of a grasp to resist forces and torques due to gravity and random perturbations.
- **Data collection policy:** evaluates actions in the synthetic training environment using supervised learning to train the policy π_θ.

2.7.1 Definitions

Considering Dex-Net 4.0 uses the synthetic training environment of Dex-Net 2.0, the variables defined in Subsection 2.3.3 are the same for this problem, with the addition of a partially observable Markov decision process (POMDP) framework in which a robot plans grasps to maximize expected reward, the probability of grasp success, given imperfect observations of the environment.

A robot with an overhead depth camera views a pile of novel objects in a bin. On grasp attempt t, the robot observes a point cloud $\mathbf{y}_t$ from the depth camera and uses a policy $\mathbf{u}_t = \pi(\mathbf{y}_t)$ to plan a grasp action $\mathbf{u}_t$ for a gripper g. The gripper consists of a 3D rigid position and orientation $\mathbf{T}_g = (\mathbf{R}_g, \mathbf{t}_g)$. After executing the grasp, a binary

reward $R_t = 1$ is recorded if it successfully picks and places the object in a target position, otherwise $R_t = 0$. The reward depends on the state $\mathbf{x}_t$, which includes geometry, pose, a centre of mass and material properties of each object. This process continues until the bin is empty or the total grasp attempts T is reached.

2.7.2 Policy

The goal of the GQ-CNN developed in Dex-Net 4.0 is to learn a policy, π_Θ, that maximizes the rate of reward, given by Equation 2.8, considering a constant time per grasp and a maximum of T grasp attempts.

The learning process is illustrated in Figure 8, and the training dataset generation pipeline was exposed in detail previously in Figure 5. First, a training dataset $\mathcal{D} = (R_i \mathbf{y}_i \mathbf{u}_i)_{i=1}^N$ is sampled from the distribution μ. Then, a quality function learnt in GQ-CNN, $Q_{\theta,g}(\mathbf{y}, \mathbf{u}) \in [0, 1]$, is used to estimate the probability of success for a given grasp with gripper g. The weights θ_g are optimized to minimize the CEM loss $\mathcal{L}$ between the GQ-CNN prediction and the true reward over the dataset $\mathcal{D}$:

$$\theta_g^* = argmin_{\ \theta_g \in \Theta} \sum_{(R_i, \mathbf{u}_i, \mathbf{y}_i) \in \mathcal{D}_g} \mathcal{L}(R_i, Q_\theta(\mathbf{y}_i, \mathbf{u}_i)) \tag{2.12}$$

where $\mathcal{D}_g$ denotes the subset of the training dataset $\mathcal{D}$ containing only grasps for gripper g. Finally, the robot policy π_θ is constructed by planning the grasp that maximizes quality $Q_{\theta,g}$ across all available candidates $\mathcal{U}_g$ and grippers $\mathcal{G}$, sampled from the depth image:

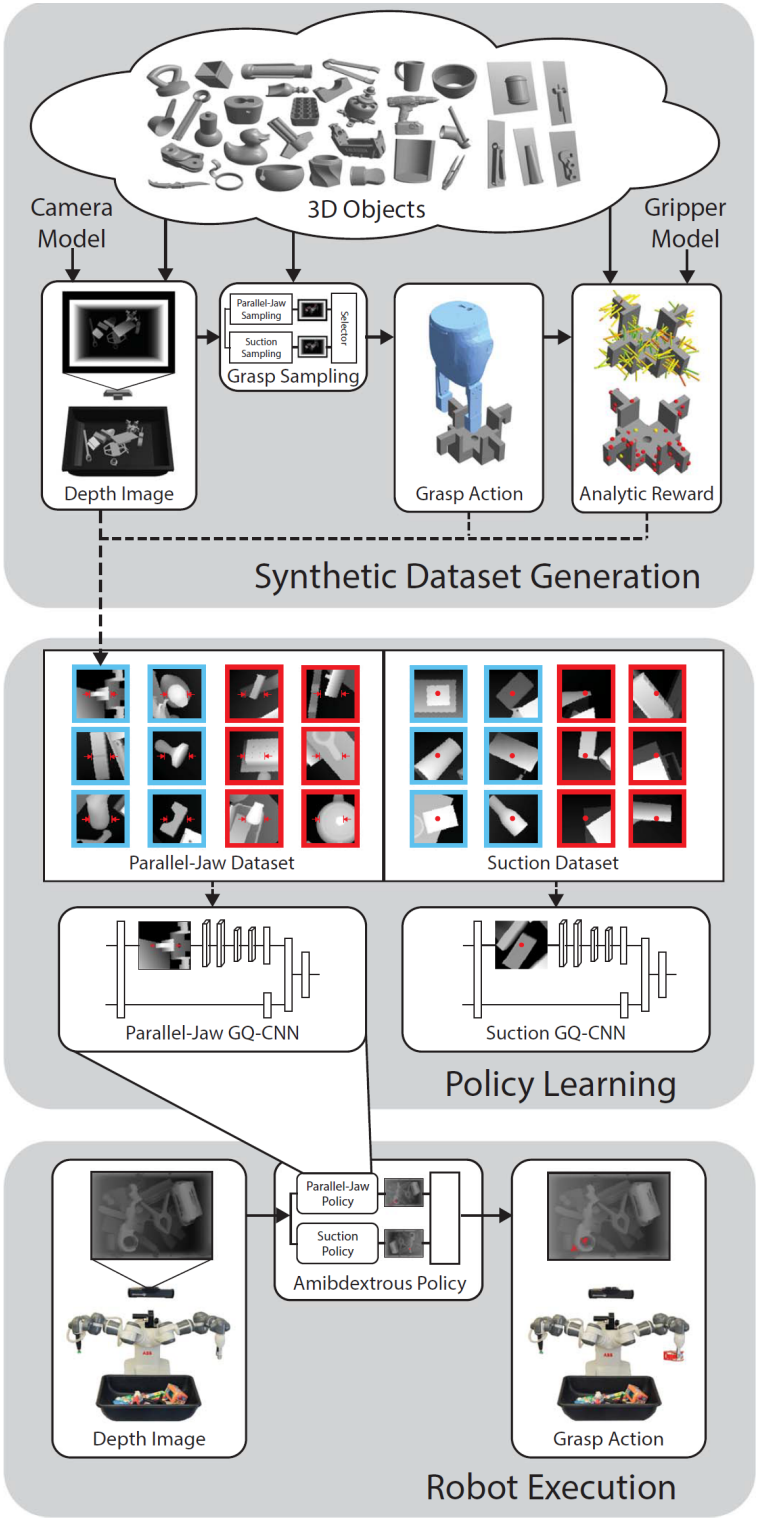
$$\pi_\theta(\mathbf{y}_t) = argmax_{\ g \in \mathcal{G}} \{ max_{\ u_g \in \mathcal{U}_g} Q_{\theta,g}(\mathbf{y}_t, \mathbf{u}_g) \} \tag{2.13}$$

2.7.3 Architecture

As previously explained in this section, GQ-CNN is trained separately for each gripper in Dex-Net 4.0. Its architecture is similar to the one used in Dex-Net 2.0 [12], illustrated in Figure 9, and Dex-Net 3.0 [17], with changes in the sizes and pooling of the layers.

The neural network takes as input the gripper depth from the camera z and a depth image centred at the grasp centre pixel (i, j) and aligned to the grasp axis orientation φ.

Figure 8: Dex-Net 4.0 [1] framework is divided in three phases: 1) Synthethic dataset generation (top); 2) Policy learning (middle); 3) Robot execution (bottom).

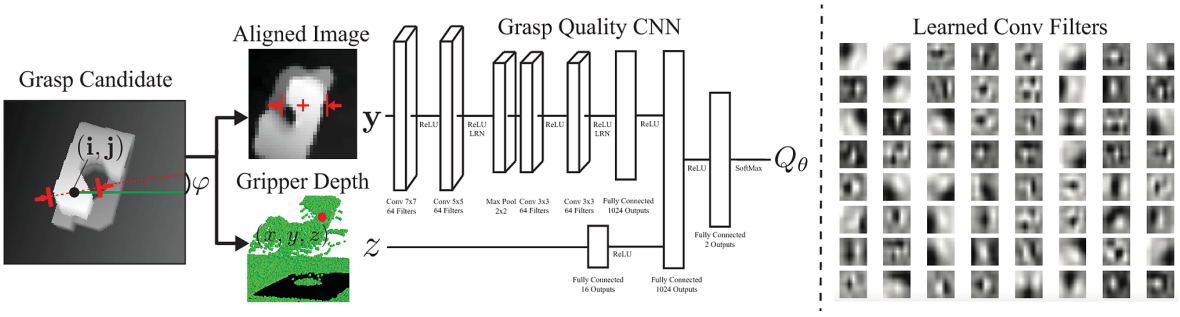


Source: Dex-Net 4.0 [1].

Then the network receives the data to estimate grasp robustness Q_θ. The architecture in Figure 9 has four convolutional layers in pairs of two separated by Rectified Linear Unit

(ReLU), explained in Subsubsection 2.7.3.1, followed by three fully connected layers and a separate input layer for z.

Figure 9: Original GQ-CNN architecture from Dex-Net 2.0.

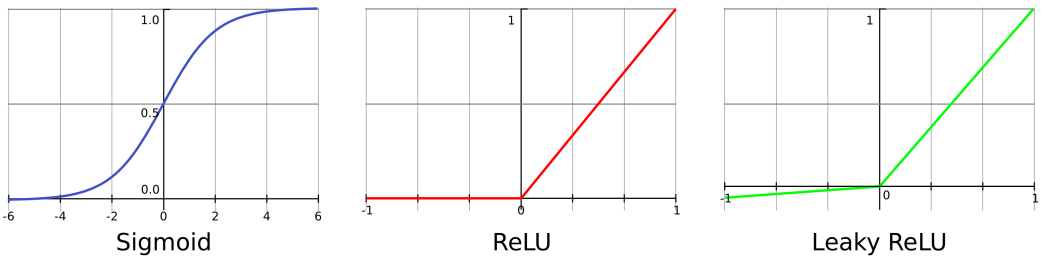


Source: Dex-Net 2.0 [12].

2.7.3.1 Activation functions

An image's semantic information is a highly nonlinear mapping of pixel values in the input. Therefore, in CNNs, activation functions are assigned to bring nonlinearities so that the network can learn any functionality. If those functions are removed, it is experimentally observed that the system performance drops by a large margin. As mentioned by Kuo [7], three activation functions are commonly used by CNNs: logistic sigmoid function, ReLU, and leaky ReLU with its variants, the Parametric Rectified Linear Unit (PReLU) and the Randomized Rectified Linear Unit (RReLU). They are shown in Figure 10.

Figure 10: Three nonlinear activation functions adopted by CNNs: the sigmoid function (left), the ReLU (middle) and the leaky ReLU (right).



Source: Kuo [7]

Logistic sigmoid works worse than ReLU in CNN learning, as mentioned by Wu [25], because it causes the magnitude of the gradient to reduce in the Backward Error

Propagation process significantly. On the other hand, ReLU can increase the network's accuracy by zeroing all the negative values and forwarding the positive ones as they are.

In the recent work of Xu et al. [26], it was proven that Leaky ReLU and its variants are better than ReLU, specially RReLU, due to its randomness in training, which reduces the risk of overfitting. In contrast to ReLU, in which the negative part is dropped, the leaky ReLU assigns a non-zero slope. For the variant PReLU, the slopes of negative parts are learned from data, while for RReLU, the slopes are randomized in a given range in training and then fixed in the testing.

2.7.3.2 Pooling

As mentioned by Teuwen et al. [6], the goal of a pooling layer is to produce a summary statistic of its input and reduce the feature map's spatial dimensions. Besides ReLU, two pairs of convolutional layers are separated by a max-pooling operation. Therefore, the max-pooling operation reports the maximum output within a rectangular neighbourhood of each point per input. It helps to make the representation approximately invariant to small input translations, as mentioned by Theodoris [27], which improves the computational efficiency of the network because the layer has fewer inputs to process.

2.7.4 Training

A CNN uses its weights as a supervision signal, optimized to minimize the network output loss value. The goal of a CNN is to match its prediction to a ground truth label.

In case of GQ-CNN, the network weights are initialized using a Kaiming initializer, created by Kaiming et al. [28], an initialization method for neural networks that consider the non-linearity of activation functions, such as ReLU. Therefore, the weights are initialized by sampling from a zero-mean Gaussian with a standard deviation of $\sqrt{\frac{2}{\eta_i}}$, where η_i is the number of inputs to the i-th network layer. The network parameters are optimized using back-propagation with Stochastic Gradient Descent and momentum.

2.7.5 Analysis

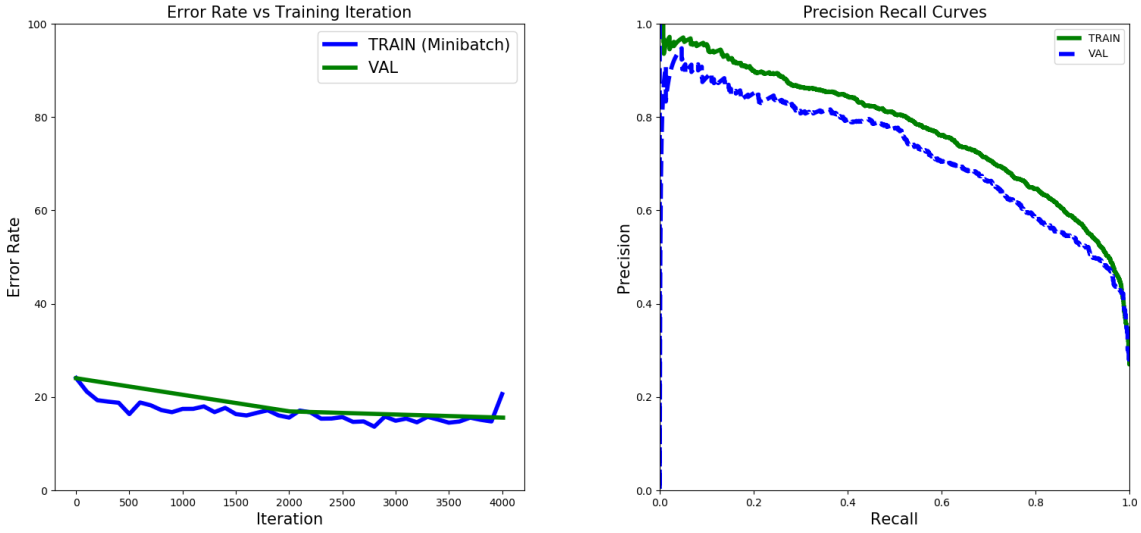
In order to analyze the performance of a trained GQ-CNN, it is helpful to check the training and validation loss and classification error in a precision-recall curve, which

can be observed in Figure 11a and Figure 11b. The plot "train" refers to the training, and "val" refers to the validation.

The learning curve in Figure 11a is used to evaluate how the algorithm learns over time. The training learning curve, in blue, is calculated from the training dataset and gives an idea of how well the model is learning. While the validation learning curve, in green, is calculated from a validation dataset that gives an idea of how well the model is generalizing. This learning curve's shape can be considered a good fit because the training and validation loss decreases to the point of stability with a minimal gap between their values. However, there is still a gap called the generalization gap.

While in Figure 11b, it can be observed that the system returns few positive results, but most of its predicted labels are correct compared to the training labels. An ideal system would have high precision and high recall.

Figure 11: GQ-CNN training analysis.



(a) Error rate versus training iteration. (b) Precision recall curves.

Source: Dex-Net 2.0 [12].

2.7.6 Cross-Entropy Robust Grasping Policy

In Dex-Net 4.0, the dataset collection policy samples grasp actions from the point cloud using the sampling techniques of Dex-Net 2.0 [16] for parallel-jaw gripper. Grasp policies with CNNs can only evaluate a limited number of grasps in a time interval.

Therefore, this approach uses CEM to optimize for the most robust grasp, in other words, the highest quality grasp.

As mentioned by Levine et al. [29], the CEM policy samples an initial set of candidates, sort them, and fit a Gaussian Mixture Model (GMM) to the top candidates. Then, the grasps are re-sampled from the new distribution, and the previous steps are repeated for K iterations, returning the best candidate from the final sample set.

2.8 Data Processing

Many classical grasping pipelines consist of an alignment phase, in which 3D Computer-Aided Design (CAD) models are used in the training set, or scans are matched to the input point clouds received by RGB-D sensors.

Object segmentation without prior models of the objects is complex due to sensor noise and occlusions. Projects such as the one developed by Schwarz et al. [30] and Mask R-CNN [31] use deep learning to segment objects for the alignment phase. Mask R-CNN is a deep learning method used to segment specific categories of objects in RGB data, but it requires a massive hand-labelled dataset. It uses semantic segmentation, linking each pixel in an image to a class label. Danielczuk et al. [32] mentioned that these techniques require time-consuming human labelling to generate training data, and existing datasets consist of RGB images of natural scenes, different from an industry's routine. Therefore, in robotics, pixel-wise object segmentation is often avoided, being used only for a small number of object classes.

This section will explain the methods employed for pre-processing the colour and depth data obtained with an RGB-D camera and the post-processing for acquiring a binary image used in the robust grasp planning policy.

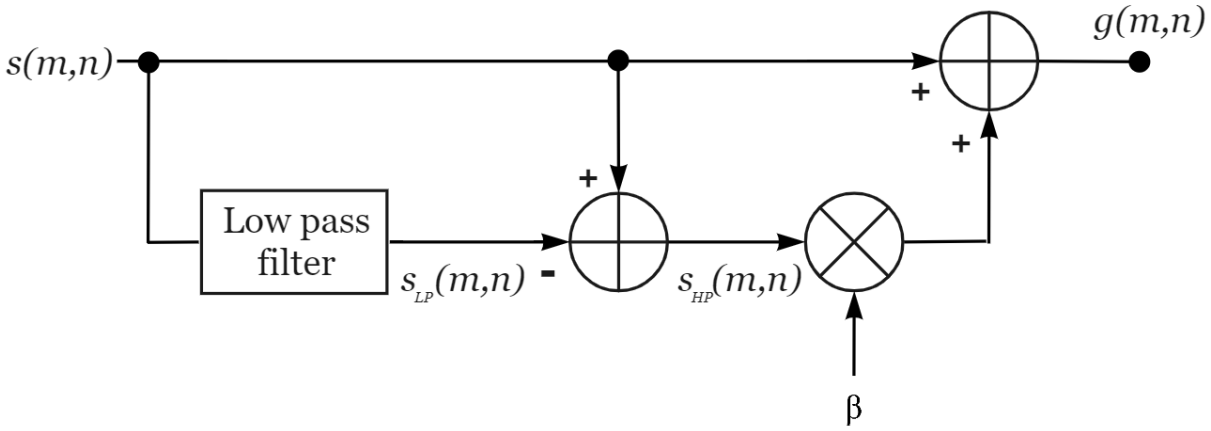
2.8.1 Unsharp mask

In medical applications, edge detection algorithms identify regions of relevant information within an image. Techniques used to enhance the contrast of images first separate the high or low-frequency components of an image, manipulating them separately and then recombining them together with different weights, as mentioned by Badamchizadeh et al. [33]. Based on the image grey levels histogram, high levels usually give relevant

information, and the low-frequency parts of an image give the background. A method called Unsharp Mask can emphasize the high-frequency parts of an image, considered an edge detection filter. It is illustrated in Figure 12, where $s(m, n)$ is the input image, $g(m, n)$ is the output image, and $s_{LP}(m, n)$ is the input image with a low pass filter. The background information is extracted by the difference between the $s(m, n)$ and $s_{LP}(m, n)$, working as a high-pass filter, which produces a detailed image. This image is weighted by a harmonization factor, represented by β, typically equal to 0.8. Finally, the output is the addition between the input and the filtered image, resulting in an enhanced image. Figure 13 illustrates the results obtained with the method described above.

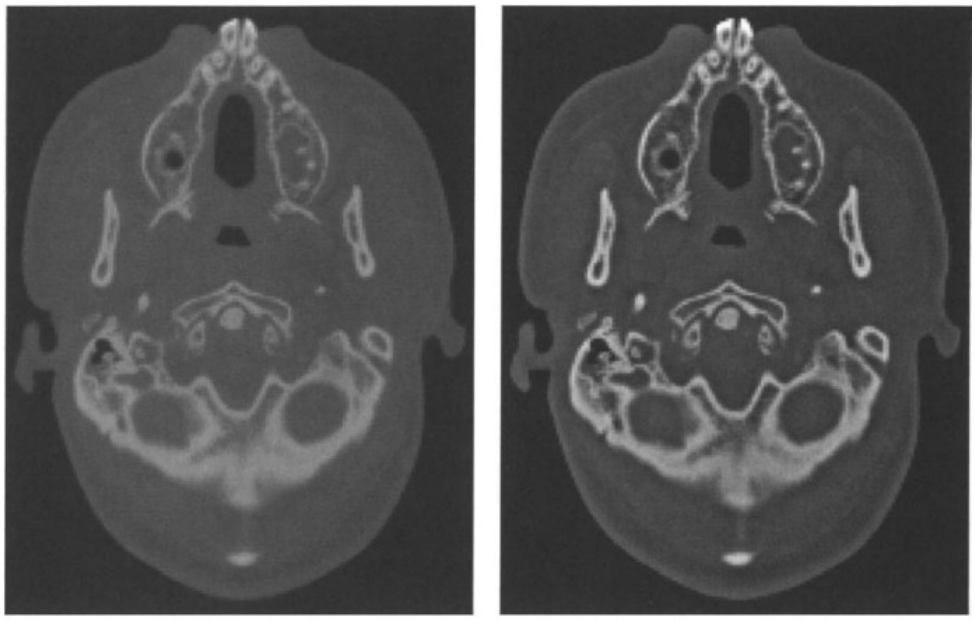
Instead of subtracting an image with a low pass filter from the original one, a Laplacian filter could be used directly as a high pass filter in the original image, and the resulting image could be added to the original one. Nevertheless, as mentioned by Badamchizadeh et al. [33], this filter is susceptible to noise present in the original image. Therefore, it was not employed.

Figure 12: Circuit of an unsharp mask used to enhance edge detection algorithms.



Source: Author

Figure 13: A filter of CT data using unsharp masking. The high-pass information of the original image (left) is twice as high in the resulting image (right). It is noticeable how details have been amplified. This technique works well due to the lack of noise in the image



Source: Kikinis [34]

2.8.2 Border following technique

The border following is one of the fundamental techniques for binary segmentation and was developed by Suzuki [35]. Considering an image in black and white, it derives a sequence of the coordinates from the border between a connected component of 1-pixels, which could be an object in this case, and a connected component of 0-pixels, which could be the background or holes.

With this approach, a binary image, in which the background is black and the observed objects are highlighted in white, can be obtained, exemplified in Section 5.1.

2.8.3 Exponential Moving Average

A low pass filter is a basis for most image smoothing methods. An image is smoothed by decreasing the disparity between pixel values by averaging nearby pixels. The filter passes low frequencies and attenuates high frequencies.

A type of low pass filter is the edge-preserving filter, which can be applied to a depth image to smooth depth noise and preserve edges. This filter raster the depth

map in x-axis and y-axis and back again, twice, while calculating the one-dimensional Exponential Moving Average (EMA) using an α parameter that determines the amount of smoothing. The recursive equation 2.14 is as follows:

$$S_t = \begin{cases} \mathbf{y}_1, & t = 1 \\ \alpha \mathbf{y}_t + (1-\alpha) S_{t-1}, & t > 1 \ and \ \Delta = |S_t - S_{t-1}| < \delta_{thresh} \\ \mathbf{y}_t, & t > 1 \ and \ \Delta = |S_t - S_{t-1}| > \delta_{thresh} \end{cases} \tag{2.14}$$

Where $\mathbf{y}$ is the newly recorded instantaneous value, and S_t is the value of the EMA at any period t. The threshold parameter used to identify the edges is δ_{thresh}. If the depth value between neighbouring pixels exceeds δ_{thresh}, then α is set to 1, so no filter is applied. If $\alpha = 1$, no filter is applied, while $\alpha = 0$ means an infinite history for the filtering.

2.8.4 Spatial filtering

Spatial filtering is a hole-filling method, where the neighboring left or right pixels within a specified radius are used to fill holes with the EMA filter explained above.

2.8.5 Temporal filtering

Temporal filtering is another filter for depth images, which is used to improve the depth map by time averaging. It uses EMA filter as well, in which α represents the extent of the temporal history, in other words, the frames that should be averaged.

Additionally, a persistence filter can be used, which fills a hole with the last valid value seen given a set of frames.

2.8.6 Rigid Transformation

In order to align an object being observed by a camera, a rigid transformation has to be calculated with equation 2.15:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \boldsymbol{R} \times \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \boldsymbol{T}, \tag{2.15}$$

in which x_c, y_c, z_c are the object's coordinates in the camera's coordinate system and x_w, y_w, z_w are the object's coordinates in the world's coordinate system. $\boldsymbol{R}$ and $\boldsymbol{T}$ are the orthonormal 3D rotation matrix of 3×3 and a 3D translation vector, respectively, representing the camera's location in the 3D scene.

The rotation matrix is given by:

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.16}$$

The translation vector is given by:

$$\boldsymbol{T} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{2.17}$$

Together they form the extrinsic matrix of a camera:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{2.18}$$

Singh et al. [36] mentioned that a camera could be calibrated with its intrinsic and extrinsic values. The camera's intrinsic values are its optical centre, in pixels, also called the principal point c_x, c_y, its focal length f_x, f_y and the skew s coefficient, which is non-zero if the image axes are not orthogonal. An intrinsic matrix, also called projection matrix, has the following shape:

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.19}$$

3 SYSTEM SETUP

Considering the advantages and limitations of the techniques mentioned earlier, the chosen method for this project was Dex-Net 4.0's pre-trained model and its grasping pipeline. The pre-trained model tends to generalize well in predictions of novel objects, i.e., not contained in the training set. Therefore, it is expected to make reliable predictions for the IDEA's aircraft parts.

Benchmarks were developed to facilitate progress and reproduction and focus on gaps in the state of the art of robust grasp planning systems. Although nowadays several researches in this area openly share code and data, it is still challenging to compare grasp planning methods and reproduce experimental results to identify which aspects of each approach work better due to variations in experimental assumptions and protocols, as in sensors and lighting, robot arms, end-effectors and objects characteristics. This work tried to reproduce the experimental setup of the adopted policy. In order to standardize the experiments, this work will report the metrics specified by Mahler et al. in [2] in the following chapter.

The following sections will specify how the system setup in this work was carried out to achieve the proposed objectives.

3.1 Data Acquisition

Cameras are available with variations in mode, for example, RGB vs RGB-D, lenses, resolution, light sensitivity and noise levels. In addition, robot jaws can have force and tactile sensors.

In order to acquire 3D data from objects, an RGB-D camera was used. This section will explain how the data collection was executed, the system's operational aspects and the type of data acquired.

3.1.1 RGB-D Sensors

The camera has an RGB-D sensor that unites traditional colourful images (Red, Green and Blue - RGB) with a depth sensor, providing shape information and colour on a per-pixel basis. This combination leads to improved object detection, recognition, 3D

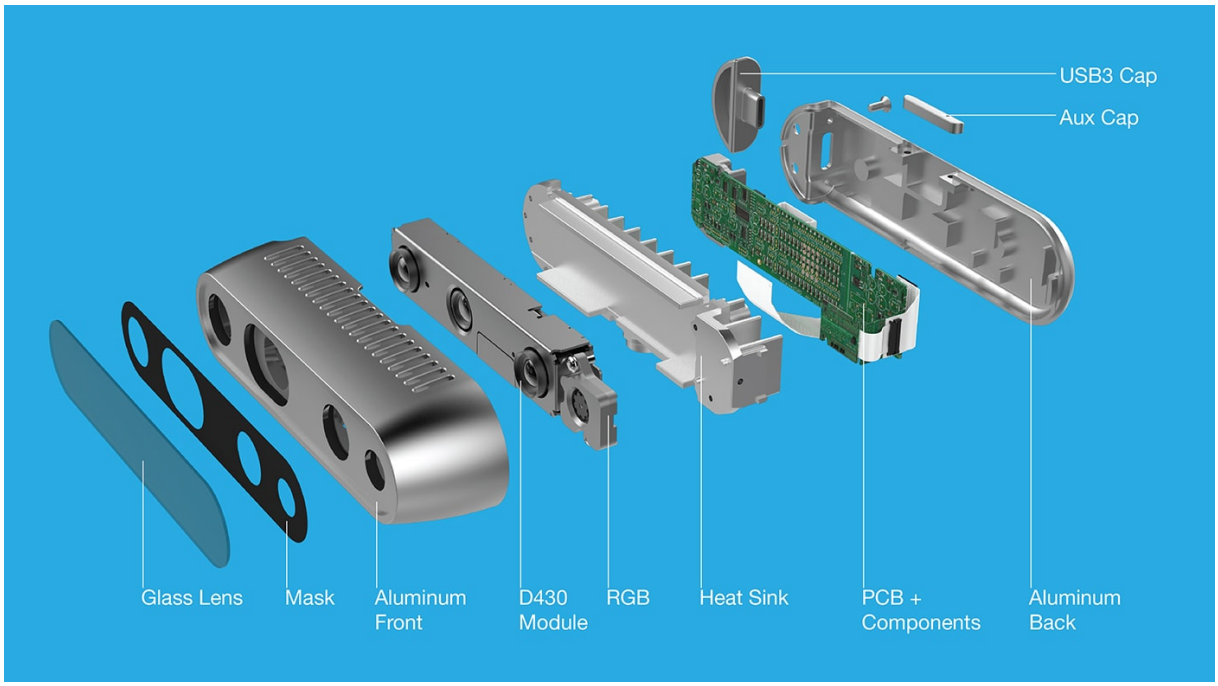
mapping and location, path planning, autonomous navigation and people tracking since some objects can only be differentiated with both information. For example, an orange can be differentiated from a lime by its colour since they have the same shape.

This project used the Intel RealSense D435 RGB-D camera to acquire data, shown in Figure 14. It has an Infrared (IR) Stereo Camera and an IR Projector.

At first, to acquire a test dataset, the camera was mounted externally to the robot in a fixed point of view 600 mm above the workspace, pointing to the centre of the workspace.

At its final configuration, for the grasping pipeline, the camera was mounted on the robot's end-effector, an eye-in-hand configuration, with a fixed initial position of 600 mm above the workspace. The data is acquired only in this position, so it has one Field-of-View (FOV) parallel to the workspace plane. The resolution used was 1280×720 with 30 Frames per Second (FPS).

Figure 14: Internal components of the RGB-D camera D435.



Source: [37].

A downside experienced by Morrison et al. [22] is that the RealSense RGB-D camera cannot provide any valid depth data on many black or reflective objects or if the object is positioned less than 300 mm away from the camera. In order to solve this problem, a matte colour spray could be applied to the object, which is a solution that

was employed in this work.

Regarding the technical aspects of the Stereo Depth Camera, in order to acquire good depth data, some parameters deserve attention during the measurement:

- **Distance**: Stereo depth cameras consist of two fixed cameras pointing in the same direction but separated by a certain distance, namely the baseline. The minimum ($MinZ$) and maximum ($MaxZ$) distance of a Stereo depth camera depend on its baseline and its focal length as follows:

$$MinZ(mm) = \textit{focal length (pixels) x Baseline(mm)}/126 \tag{3.1}$$

  Every pixel of each image needs to be shifted to match the images produced by the right and left cameras. Those shifts are quantified by the number of pixels shifted, known as the disparity. The disparity shift can be increased to decrease the minimum distance, but this would also decrease the maximum distance. Therefore, the minimum distance adopted was 300 mm.

- **Depth resolution**: Since the input resolution directly affects the input image and depth precision, the highest available resolution, which is 1280x720, was used in this project. The selected resolution also affects how many frames per second (FPS) the camera can obtain.

3.1.2 Types of Data

There are different formats available to store 3D object models: PCL, Stereolithography (STL), Standard for the Exchange of Product Data (STEP), Polygon File Format (PLY), among others. The choice of format is based on the use case. For this project, the object models obtained with RGB-D were saved as PLY, containing both colour and depth information, and depth images were stored in Numpy array (NPY) format. This subsection will explain the aspects considered in each data format.

- **PLY**:

  Analysing the data type that RGB-D sensors output, they are an RGB description of the image with the correspondent depth information, given on a per-pixel basis.

Therefore the data obtained can be stored in PLY format, a type of structured mesh, where the data can be accessed by indexes and the vertices positions are explicitly stored. In this format, the object definition comes as a list of X, Y, Z, triples for vertices and a list of faces described by indices into the list of vertices.

- **NPY**:

  In order to process the data generated by measurements, it needs to be converted into an easily readable format. The format adopted here is the Numpy array [38]. It is a binary format that stores all the shape and data type information necessary to reconstruct the array correctly in machines with different architectures. Therefore, it was chosen to store the depth information.

3.2 Software and Hardware

In order to achieve the goal of this project, the following software and hardware were used. Regarding the adopted software:

- **Intel RealSense SDK**: it is a Software Development Kit (SDK) created to use the Intel RealSense cameras, capture depth images, RGB images, export point clouds, produce videos, post-process the signal of both colour and depth data.
- **SolidWorks**: used to visualize CAD models chosen to 3D print.
- **PyCharm**: software used to program in Python and to interact with the project Git repository.

Regarding the adopted hardware:

- **Computer System:**

  A notebook with Intel(R) Core(TM) i7-4500U Processor and 8 GB installed memory, Random Access Memory (RAM), and the operating system Linux Ubuntu 16.04 LTS. The computer has 3 USB 3.2 terminals.

- **Cables:** An USB-C cable was used to connect the camera to the computer. It was purchased together with the camera.

3.3 Robot Setup

This section will expose the setup adopted in the project for a future application with the robot. No grasp test has been performed in the robot during this work because this system was developed in the IDEA project initial phase. Therefore, other parts, such as trajectory planning and grasp execution, are under development at the WZL institute, which are essential for the grasping tests with the robot. It should be regarded that access to the laboratory was restricted due to the pandemic, consequently the development of the subsequent parts of this work was delayed.

3.3.1 Gripper

Two types of robot end-effectors were used in the robust grasp systems mentioned before: parallel-jaw [12], [16] and vacuum suction [17]. Both require calculating the force that the gripper and suction cup must apply to hold an object without damaging it and checking which points are appropriate for grasping the objects.

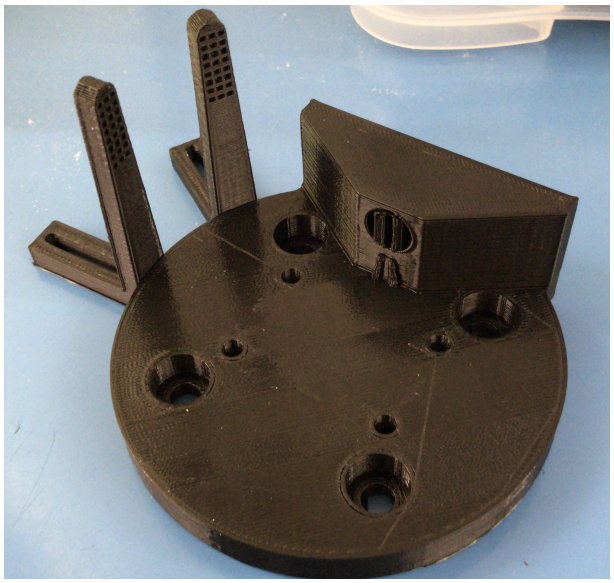
Vacuum suction grippers are widely used in industry and are often preferred over parallel jaw claws and multi-finger claws due to their ability to lift objects with a single point of contact. This ability simplifies planning and the selection of suction points in objects. However, this gripper is mainly used on flat and smooth surfaces.

Works such as Dex-Net 4.0 [1] explore the possibility of using two or more heterogeneous grippers, which is called "ambidextrous" robot grasping. The deep learning model adopted in this project, Dex-Net 4.0, can be used for suction and parallel-jaws grasping points predictions, but only the prediction of parallel grasps was explored. Therefore, parallel jaws were designed, and 3D printed using polylactide (PLA) as material. The tips for the parallel-jaws were manufactured considering the design studied by Guo et al. in [39], with variation in the size and type of surface texture. Figure 15 shows one of the tip models, manufactured with 3D printing, and the pairing socket to connect the robot, the Schunk PGN Plus 50/2 gripper, the claws and the RGB-D camera.

3.3.2 Workspace

The robot's workspace consists of a black background approximately 600 mm away from the camera, which is located in the robot end-effector. This background is black and

Figure 15: 3D printed parallel-jaw at the left and pairing socket for the parallel-jaw at the right.



Source: Author.

not reflective to avoid obstructing the measurement of objects by the RGB-D camera, as explained previously in this chapter.

It is also relevant to notice that the workspace has to consider the robot FOV, in which the robot kinematics allow it to perform a vertical grasp.

Concerning the lighting system for the project, an indoor ambient artificial light was adopted.

3.3.3 Test set

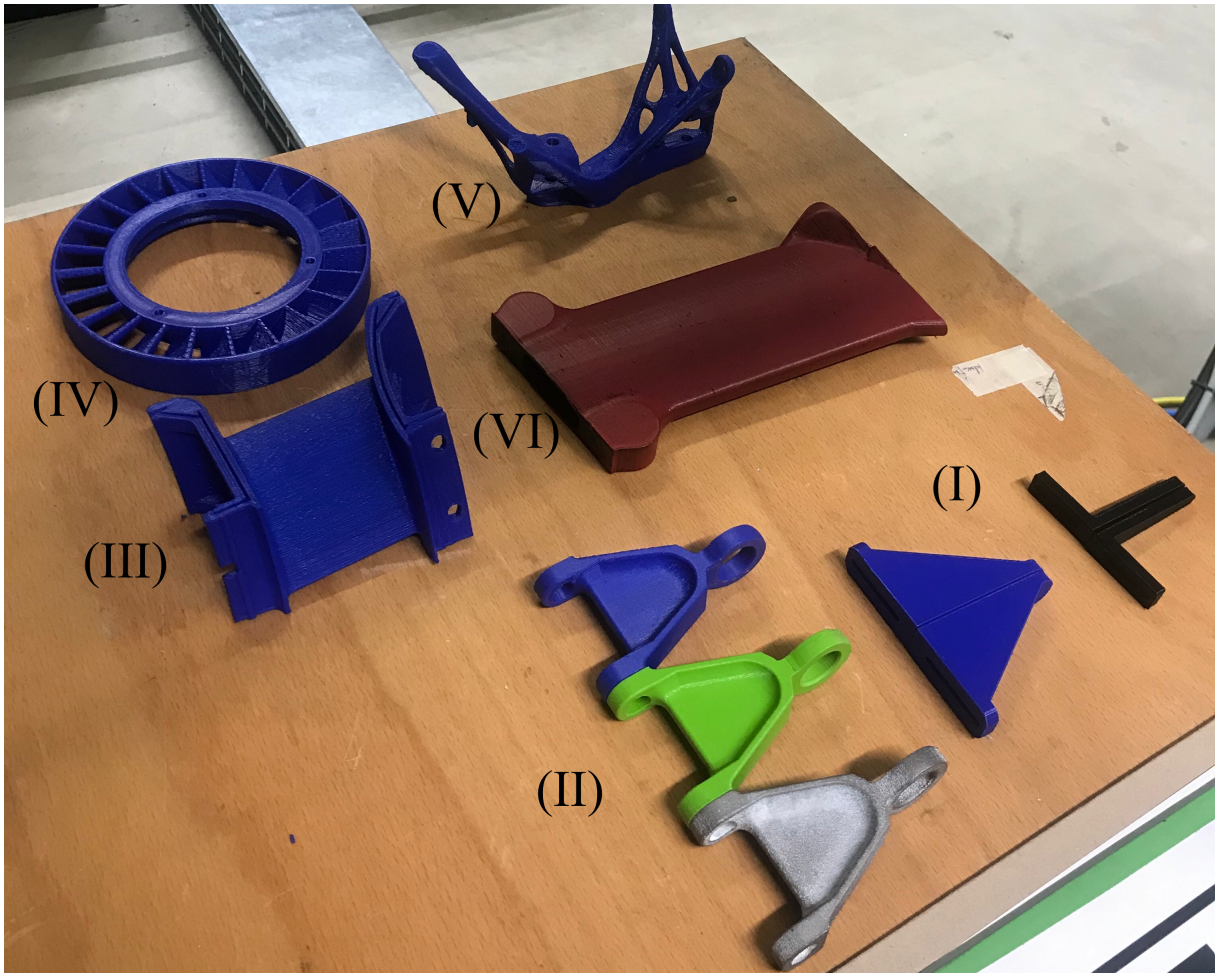
The aircraft parts from the IDEA project were manufactured by multiple sources, with a 3D printer using PLA and metal laser powder-bed fusion (L-PBF), and they were used in the data acquisition process for the policy input. Additional objects with simple geometries made of wood were used as well.

Once the objects from the Dex-Net 4.0 training set have been analyzed, geometry similarities with the IDEA project objects were found. For example, the shape of a blisk can be simplified to the one of a doughnut. The CNN tends to perform well in objects with geometries similar to the ones contained in the training dataset. Consequently, successful predictions were expected to be achieved.

As explained earlier in this chapter, the colour of an object and its reflection

can cause poor measurements by the Intel Realsense camera. This problem was solved with the application of a matte colour spray. Figure 16 shows the objects that were manufactured as well as two grippers for the parallel-jaw gripper with differences in the support geometry, the contact surface as well as the contact area of the tip, at the extreme of the gripper

Figure 16: All the 3D printed objects used in this project: (I) two models of jaws; (II) aircraft part manufactured in metal and PLA (coloured); (III) and (V) are objects from the IDEA project, (IV) is a type of blisk and (VI) is a turbine support.



Source: Author.

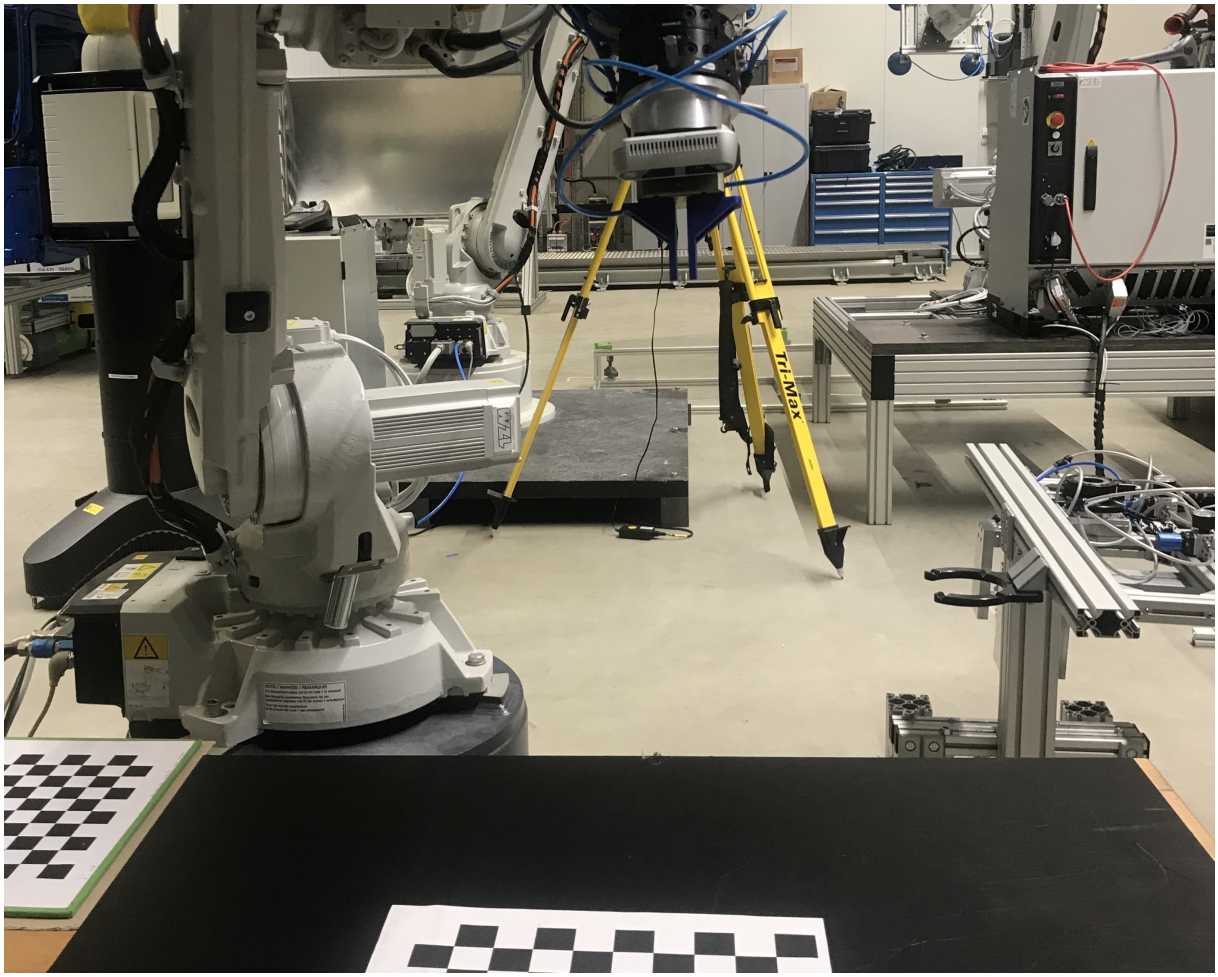
3.3.4 Transformation

The camera was calibrated using the OpenCV's calibration routine based on the method proposed by Zhang [40]. This method estimates intrinsic and extrinsic camera parameters from several views of a known calibration pattern, where every view is described by several 3D-2D point correspondences. Therefore, an easily detectable object

with known geometry shall be used as a calibration pattern. In this method, a chessboard is used as a calibration pattern, as shown in Figure 17.

The transformations calculated with this method are used to determine the exact pose of the object in relation to the robot and make it possible to compute the movement of the robot to perform the grasping.

Figure 17: The chessboard is positioned on the workspace for the hand-eye calibration between camera, robot and workspace.



Source: Author.

In order to facilitate the transformations between camera, robot and gripper, the eye-in-hand position for the camera was adopted.

From Equation 2.19, the intrinsic parameters of the Intel RealSense D435 RGB-D camera for 1280×720 resolution are:

$$\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 637,6 & 0 & 640,2 \\ 0 & 637,6 & 372,0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

The final result of the GQ-CNN prediction is a pose and a probability of successful grasp with the predicted pose, chosen from a sample space of candidate grasps. For the given pose, the following transformations must be applied:

$$|\boldsymbol{T}_{camera-gripper} = \boldsymbol{T}_{camera-world} \times \boldsymbol{T}_{world-grasp} \times \boldsymbol{T}_{grasp-gripper}|, \tag{3.3}$$

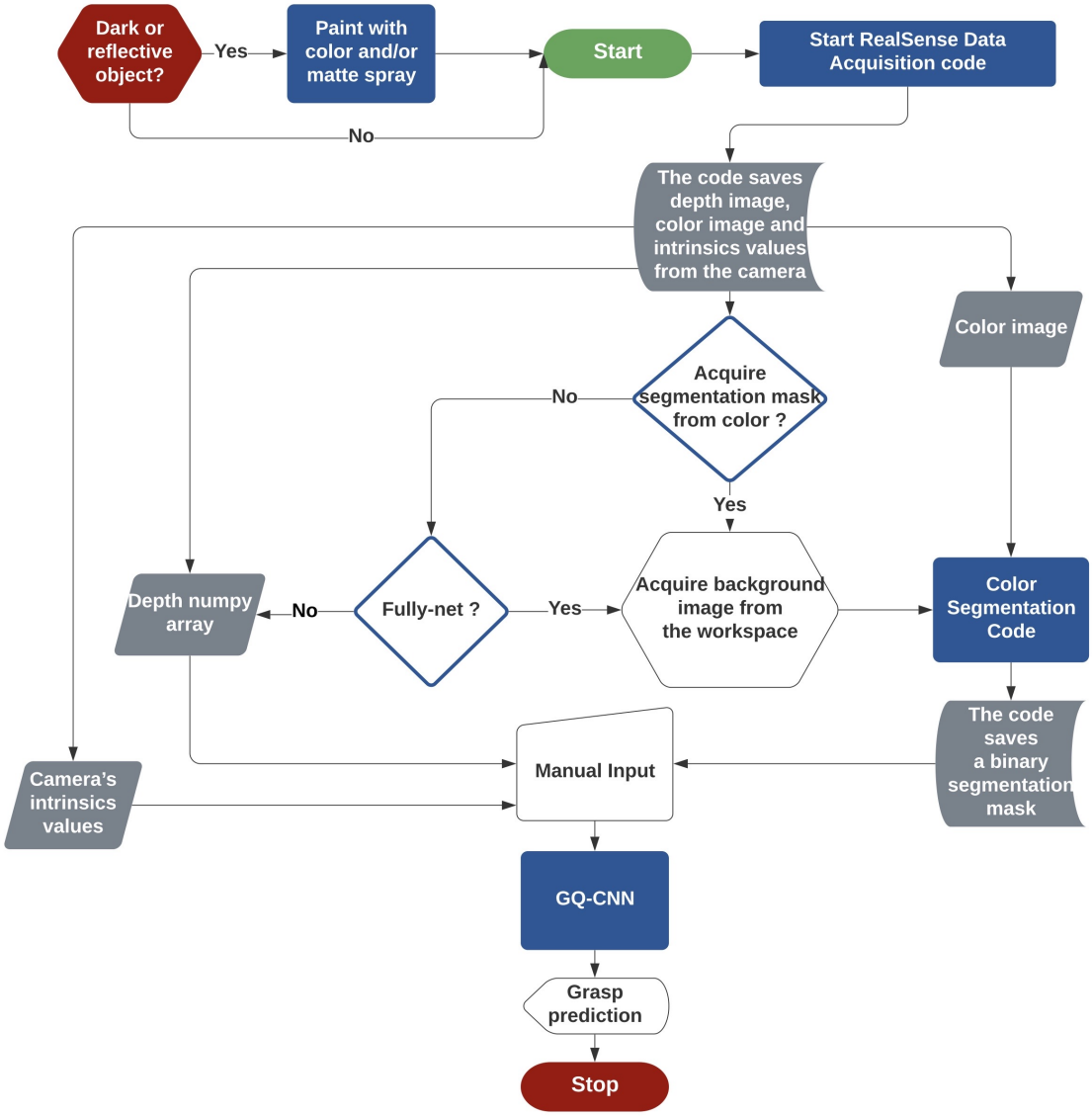
where $\boldsymbol{T}_{grasp-world}$ is given by the rotation quaternion and the translation.

3.4 Grasp Detection Pipeline

The grasping pipeline comprises three stages: image processing, evaluation of networks and computation of grasp pose. The best grasp pose is calculated from a single viewpoint and executed by the robot open-loop.

In order to facilitate the understanding of the process, the flow chart presented in Figure 18 was created.

Figure 18: System's open-loop control flow chart.



Source: Author.

4 SIGNAL PROCESSING

This chapter refers to the previously proposed data acquisition system's test and application of the signal processing methods.

The GQ-CNN model from Dex-Net 4.0 needs to be fed with a mandatory depth image, the camera intrinsic values and an optional binary segmentation mask. If the segmentation mask is not provided, one is derived from the depth image with the inverse of the zero-valued pixels, called invalid pixels.

The following sections will expose the methods used to post-process the depth and colour image.

The post-processing method applied in Dex-Net 4.0 for acquiring the binary segmentation mask for the GQ-CNN input uses a point cloud acquired by an RGB-D camera. The method applies Euclidean Clustering to the point cloud to find objects based on the nearest neighbour points within a threshold area.

Since the depth image generated with the Intel RealSense D435 contains more noise than the one obtained in Dex-Net 4.0, which uses Photoneo Phoxi S, an industrial depth camera with high precision, a better segmentation mask could be obtained using the coloured image instead.

The methods used for noise reduction in the depth image and binary segmentation will be explained.

4.1 RGB-D data pre-processing

The way to process data depends on its application. For this work, high accuracy is better than high information density. Therefore, the filters chosen for the RGB-D data provide a depth map that is less dense, which is essential in decision-making tasks.

The RealSense camera was expected to be straightforward to use. However, the camera produces noisy point clouds, and special attention had to be given to this situation by investigating which filters would be best for this application.

According to the Intel RealSense developers [41], a proper set of default values for the spatial filtering would be $\alpha = 0,6$ and $\delta = 8$, where δ is in units of $1/32$ disparities, so 8 means $8/32$ disparities.

Temporal filtering was also adopted to smooth data, but its persistency mode was

avoided in this case, because it would be favourable in a scene where poor light conditions and the appearance of holes are common due to shadowing. In the current project, where high accuracy is required, an optimal light source is provided, and the system stays in a fixed position, it is not needed. The values adopted for the temporal filter were $\alpha = 0,5$ and $\delta = 20$, as suggested by the camera developers [41].

Although being unnoticed by the human eye, a camera can capture the flickering effect in its video frame. In order to avoid this effect, the Intel RealSense camera has its control for power line frequency, which can be changed to 50Hz or 60Hz. The light source in the project measurement set was artificial, so it had a frequency of 50 Hz, considering it was implemented in Germany, which means it turns on and off 50 times per second. A flicker avoidance of 50 Hz was therefore set.

4.1.1 Codes for RGB-D data acquisition and processing

Codes were developed to interact with the RGB-D camera and apply the previously exposed concepts, which are contained in the appendix section and explained below.

A configuration file, presented in Appendix A, set the Intel RealSense camera and its filters with the configurations explained previously in this chapter. The code contains variables with names similar to the parameters that need to be set to configure the filters to facilitate comprehension. Moreover, it is possible to configure in which folder the data should be automatically saved and the respective filenames in this file.

The parameters specified in the configuration file were used in a code created to interact with the Intel RealSense camera without requiring its SDK. Considering that the data should be similar to those adopted by the authors of GQ-CNN, this code was based on another one they developed, which is used for a different RGB-D sensor. This code is shown in Appendix B.

Then, in order to execute the two codes explained above directly from the command line and start the data acquisition loop, the code exposed in Appendix C was developed.

4.2 Binary segmentation

In order to obtain the binary segmentation image from colour images taken from the RGB-D camera, the following techniques were used. This method was applied on a

black matte background, but the results were inaccurate for reflective objects or very dark colours, as seen in Chapter 5.

The Unsharp Mask proposed before was tested, but the results obtained were considerably better while using a background image instead of applying a low pass filter to the original image. In this technique, defined as foreground masking, an image is subtracted from a background image, both in grayscale. The resulting image is filtered with a threshold for the upper and lower grey values. Afterwards, it is converted to binary, with pixel values of 0 and 1. In this binary image, a function to find the contours, with the border following algorithm specified in Subsection 2.8.2, is applied to remove all white connected points with an area bigger than a threshold specified and distance from other objects larger than another threshold value. In the end, the detected objects appear in white contours, and the background is black. This image is used as a segmentation image to feed the neural network.

This method is considered semi-automatic in its level of user interaction, taking into account that it is necessary to provide an updated background image, considering changes in light conditions at the time of measurement.

4.2.1 Codes for binary segmentation

Based on the theory exposed above, two codes were developed to perform the binary segmentation with the colour images obtained from the RGB-D camera.

The code exposed in Appendix D was developed to configure the filter's parameters, which contains information about the threshold values for the grayscale, the distance and the area for the function that finds the contours in images. The parameter values were obtained using as baseline the correct definition of the object's contours, which varies according to the object size because the border following algorithm uses the definition of a minimum area to recognize an object. A minimum and maximum value were also configured for the depth at which an object could be found, in case a filter based on the point cloud obtained by the RGB-D camera was used.

The second code, shown in Appendix E, is a class created to segment coloured images and automatically save them as input for the GQ-CNN neural network.

5 DATA ACQUISITION RESULTS

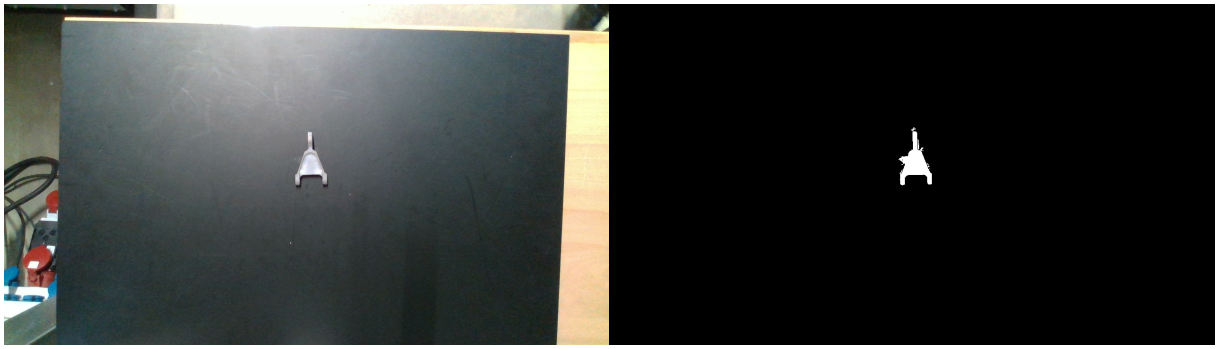
This chapter refers to the results of the previously proposed data acquisition system and signal processing methods.

The figures shown here were taken at different times of the experiment, and the colour image processing was made with the final setup of height for the camera and background described in Subsection 3.1.1.

5.1 Color Image Processing

In order to compare the original output and after adopting the processing methods described previously, Figure 19, Figure 20 and Figure 21 represent the object segmentation with the algorithm still in its initial configuration.

Figure 19: Comparison for the aircraft part without filter, between: a) RGB image; b) binary image.



(a) (b)

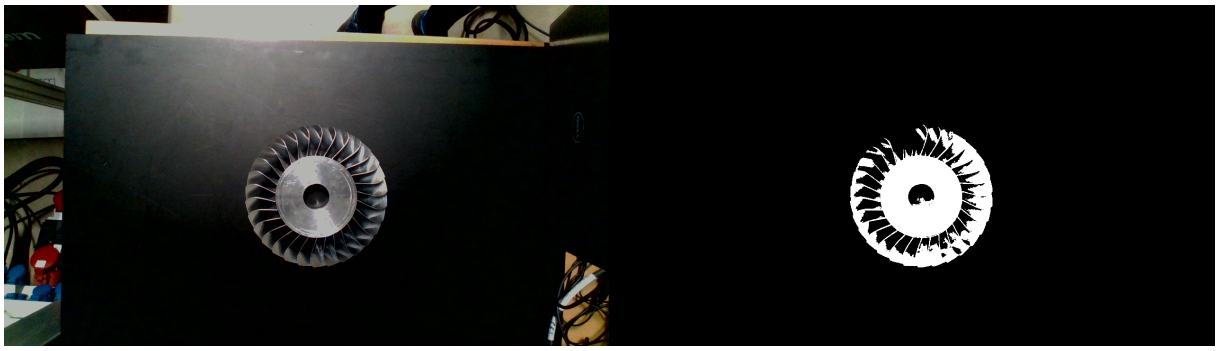
Source: Author

In Figure 20a, it is possible to see how dark the colour image was before the filters were applied, which made it difficult to obtain a correct segmentation image.

In Figure 21, it is possible to see that the initial configuration of the filters for segmentation still left noise around the edges of the objects, so it was necessary to configure the parameters of these filters until reaching optimal values according to the data obtained by the RGB-D camera. This set of parameters can change if another place is chosen for data acquisition with another lighting condition.

Hence, through experimental tests directly on the data obtained, the configuration of parameters shown in Table 4 could be reached, which were used in the code exposed in Appendix D.

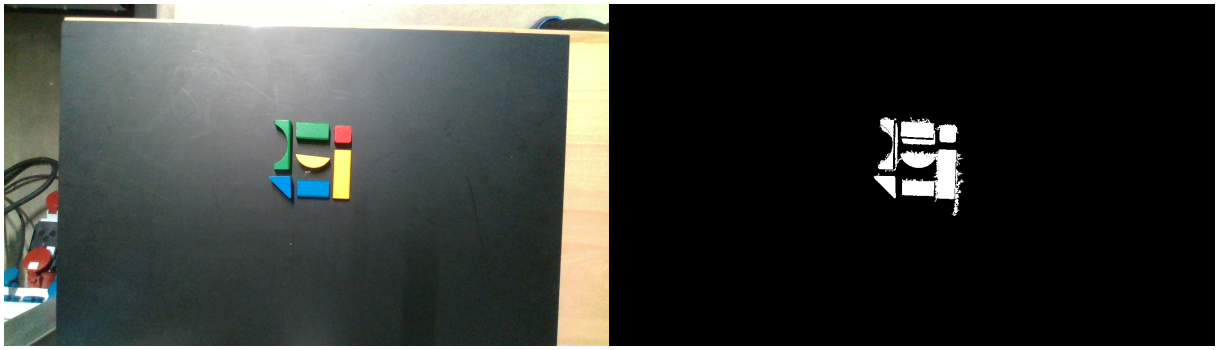
Figure 20: Comparison for the blisk without filter, between: a) RGB image; b) binary image.



(a) (b)

Source: Author

Figure 21: Comparison between (a) RGB image without filtering and its (b) segmentation applied to multiple objects made of wood.



(a) (b)

Source: Author.

Table 4: Segmentation Configuration

Segmentation Configuration					
Grayscale Range		Depth Threshold		Contour Threshold	
Low	Upper	Low	Upper	Distance	Area
100	250	0,3	0,6	1000	300

Source: Author.

Finally, using the code in Appendices D and E, with the parameters exposed in Table 4, it was possible to obtain segmentation images with satisfactory quality for the alignment phase. A sample set of segmented images are exposed in Figure 22, Figure 23, Figure 24 and Figure 25.

In Figure 22, it is possible to observe that the filtering brought significant improvements in the quality of the RGB image and consequently in the binary segmentation.

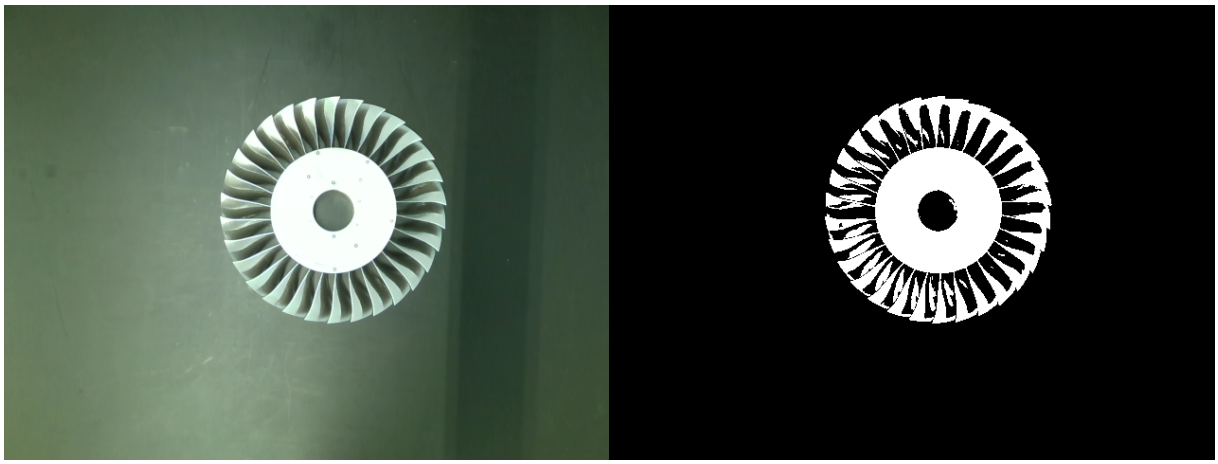
In Figure 24, there was also an improvement, but as seen in the bottom right corner, the dark blue square object could not be segmented because there was no contrast

Figure 22: Comparison between (a) RGB image without filtering and its (b) segmentation applied to the aircraft part.

(a) (b)

Source: Author.

Figure 23: Comparison between (a) RGB image without filtering and its (b) segmentation applied to the blisk.



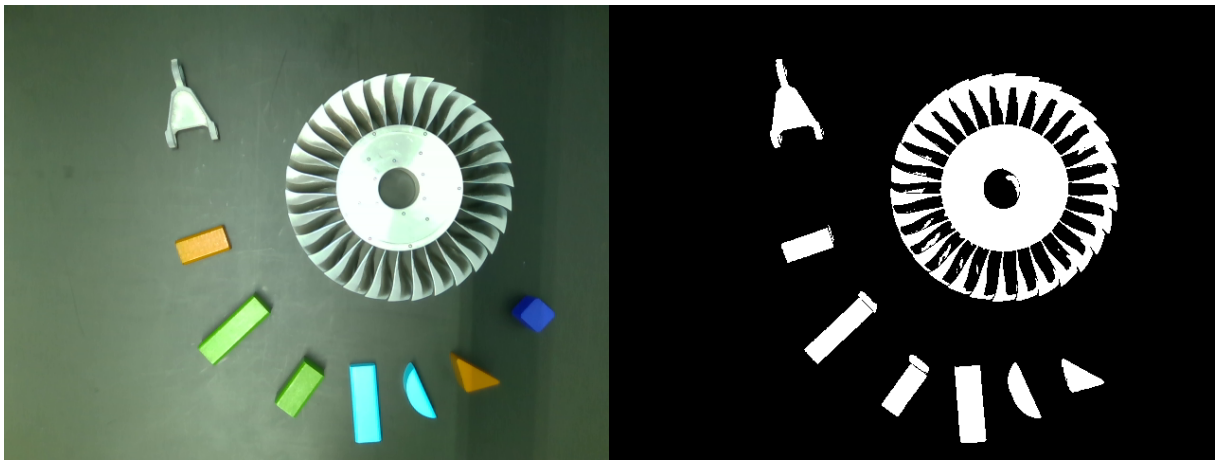
(a) (b)

Source: Author.

between the object and the background since both are dark.

Among all the methods tested and analysed, this one proved to be the most robust, by analysing the silhouette obtained for the objects, and with the best results in the segmentation of objects contained in colour images, considering proper lighting conditions.

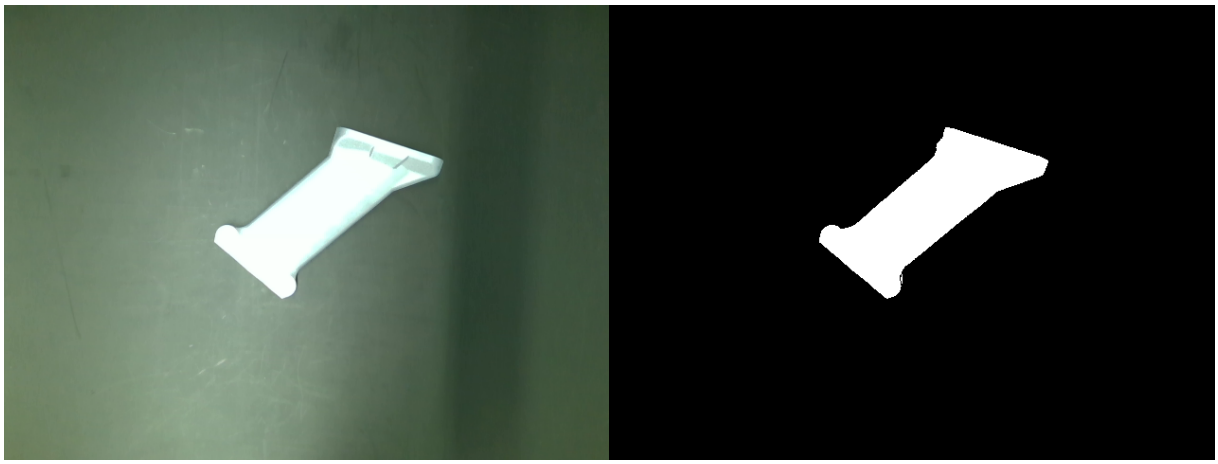
Figure 24: Comparison between (a) RGB image without filtering and its (b) segmentation applied multiple objects with different geometries.



(a) (b)

Source: Author.

Figure 25: Comparison between (a) RGB image without filtering and its (b) segmentation applied to the support object.



(a) (b)

Source: Author.

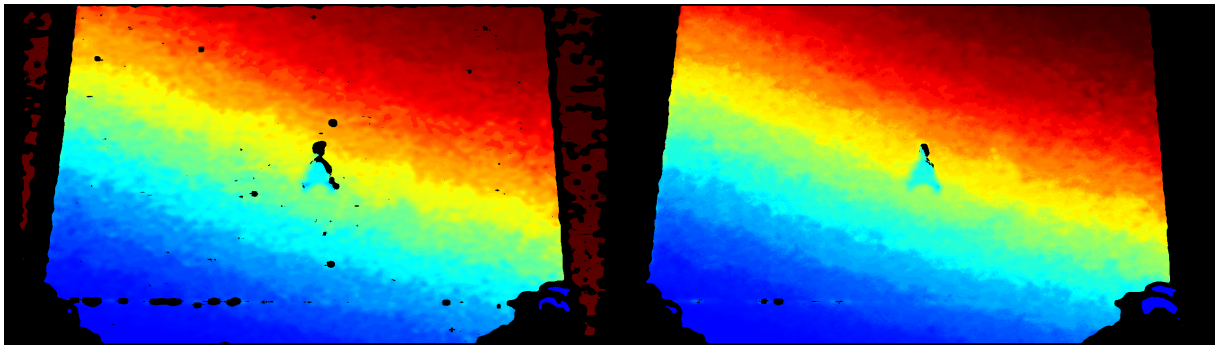
5.2 Depth Data Processing

Figure 26 compares a point cloud without processing and another with processing. As can be seen in Figure 26a, there are black dots in the depth map, that represent depth data not available or that did not meet a confidence metric of the camera, and instead of providing a wrong value, the camera provides a value of zero at that point, as mentioned by Grunnet-Jepsen et al. in [41]. In order to fill the black dots appropriately, hole-filling methods, such as spatial and temporal filters, were applied. Moreover, the minimum distance between the camera and a target was adjusted to meet the setup used in this

project.

It is noteworthy that these images were obtained for a camera position different from the one used at the end of the project, inclined in relation to the table on which the object was positioned. Therefore, the depth colours vary along the image, indicating as blue the points closer to the camera until the most distant points in red.

Figure 26: Set of depth images in which the workpiece was positioned in the middle of the test table: a) without processing; b) processed with the final configuration, which resulted in less data loss, as can be inferred by the fewer number of holes in the image.



(a) (b)

Source: Author.

In order to evaluate which configuration is more accurate to obtain a clearer depth image, with less noise and which better represents the object, several configuration sets were tested. The final configuration is described in Table 5 and the theory and code used for arriving at this configuration were explained in Section 4.1.

Table 5: Depth processing parameters.

Depth Processing Parameters					
Spatial Filter		Temporal Filter		Depth Threshold	
Alpha	Delta	Alpha	Delta	Low	Upper
0,6	8	0,5	20	0,3	0,6

Source: Author.

With this configuration, it was possible to acquire depth maps such as the one exposed in Figure 26, where the black dots were covered, and the contours of the aircraft part became sharper, so the result can be considered satisfactory.

6 GQ-CNN RESULTS

In this chapter, the outputs of the neural network will be exposed, which is the prediction of an antipodal grasp position for a parallel-jaw gripper, and a grasp success probability, called Q-value. The results with low Q-value and with high Q-value will be exposed separately.

Codes developed by the authors of GQ-CNN will not be exposed here. However, they were employed in order to utilize the neural network. The GQ-CNN necessarily receives as input a point cloud containing the object for which it intends to predict a grasp position. However, optionally a binary segmentation image can also be used, which could help in the stage of alignment and localization of the object by the neural network.

In order to analyze the real benefit of using a segmentation image, two predictions were made for each point cloud obtained, one with the segmented image and another without it. Confusion matrices based on the visual analysis of all predictions made will be exposed since it was impossible to perform physical tests in the robot until the conclusion of this work because these tests depend on the robot control system, which the WZL institute is currently developing. Although not conclusive, the manual and visual assessment stands as a preliminary analysis, which is critical to rely on before testing in a real robot, as this could lead to accidents if there were predictions with high Q-value and antipodal points outside the area of the object.

Before analysing the results, the evaluation criteria and adopted thresholds will be explained.

6.1 Grasp Prediction Classification

The visual evaluation of the obtained predictions will be explained below. First, a threshold of 50% for the Q-value was established, defining as positive Q-value predictions above this value and as negative the remaining ones. The Q-value indicates a probability of success linked to a grasp position for the object, given by the neural network. Therefore, the grasping positions indicated were evaluated as susceptible or not.

A prediction is True Positive if it has a high Q-value and indicates a favourable grasping position for the object. However, if the position is inadequate, it is a False Positive. Moreover, given a low Q-value and a suitable grasping position, the prediction

is classified as False Negative, while for an inappropriate position, it is a True Negative. Table 6 shows this classification.

Table 6: Grasp prediction classification.

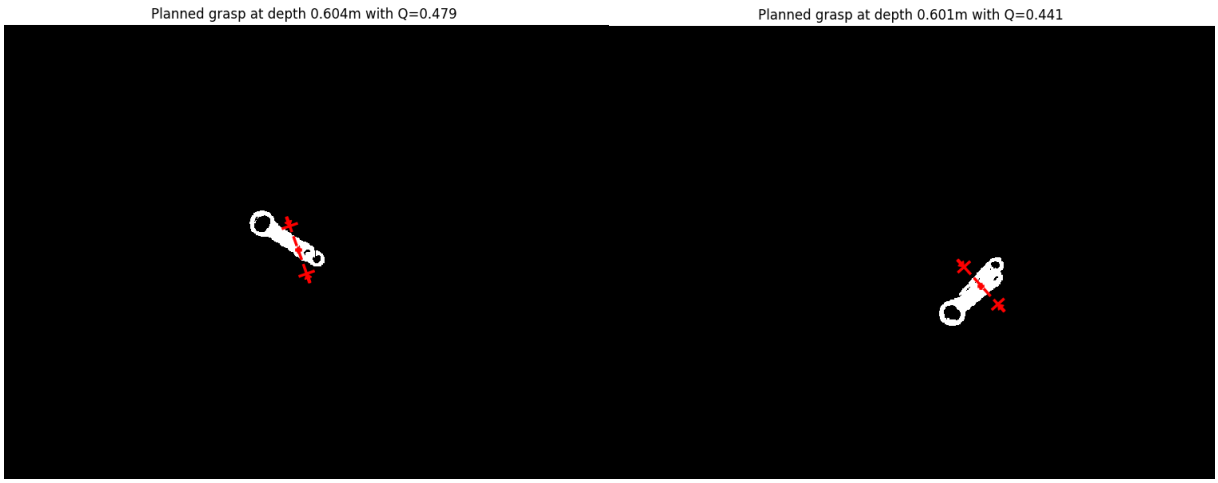
Grasp Prediction Classification		
Grasp Prediction	Q-value	
	High	Low
Feasible	True Positive	False Negative
Not feasible	False Positive	True Negative

Source: Author.

The feasibility analysis of grasping an object given a position considered the object geometry, the gripper width, and its mechanical limitations. Examples with images will be exposed subsequently to facilitate comprehension.

Figure 27a presents the A-link object resting on its side face, for which a significant oblique position for the gripper in relation to the object was estimated. Due to this position, the aircraft part would probably escape from the robot gripper before it could even remove the object from the test bench. Therefore, significant oblique positions relative to a flat surface were considered not ideal for grasping objects. Whereas in Figure 27b, the estimated grasping position for the same object is perpendicular to its flat surface, so there is a higher probability that the robot can grasp the object. This position was seen in other results and therefore considered ideal for grasping.

Figure 27: Comparison for the aircraft part between: a) not feasible grasp position; b) feasible grasp position.



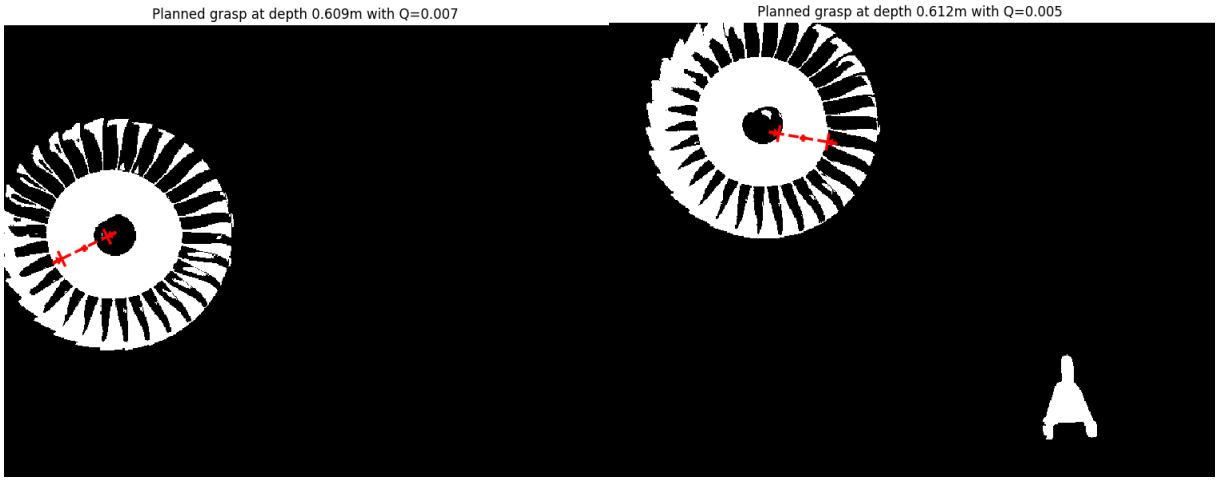
(a) (b)

Source: Author

In Figure 28a, the network estimated an unfavourable position for grasping, with

one jaw in the blisk centre and another in its rotor. This position was observed in other predictions and therefore also considered as negative. Meanwhile, in Figure 28b, it can be noted that the gripper has a jaw positioned in the central and outer circles of the rotor and is therefore considered a correct position to grasp the object. It should be noted that the jaw width is small, so it fits well in this position.

Figure 28: Comparison for the Blisk between: a) not feasible grasp position; b) feasible grasp position.



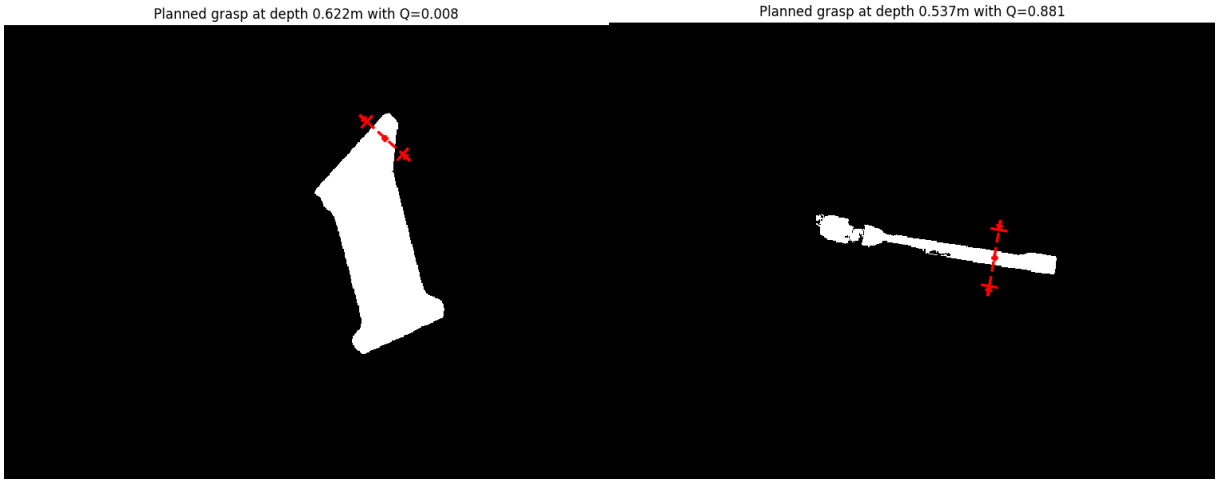
(a) (b)

Source: Author

The estimated grasping position in Figure 29a is on the tip of the turbine support. If both edges were parallel, this position would be feasible. However, since one of them is oblique to the gripper's jaw surface, it will probably slip when lifting the object. Therefore, the object's edges are not a feasible position for grasping. Furthermore, the object's width is greater than the gripper's opening diameter when positioned in this manner. Therefore, it is not possible to grasp in its centre. While in Figure 29b, the object is resting on its side face, which width is smaller than the gripper opening diameter, thereby facilitating the grasp position prediction.

As discussed for the previous objects, oblique edges in relation to the gripper surface can cause the object to slide when grasped. Consequently, the position defined in Figure 30a for the coloured block is not feasible. While in Figure 30b, the estimated position is adequate since it is perpendicular to the edges of the coloured block.

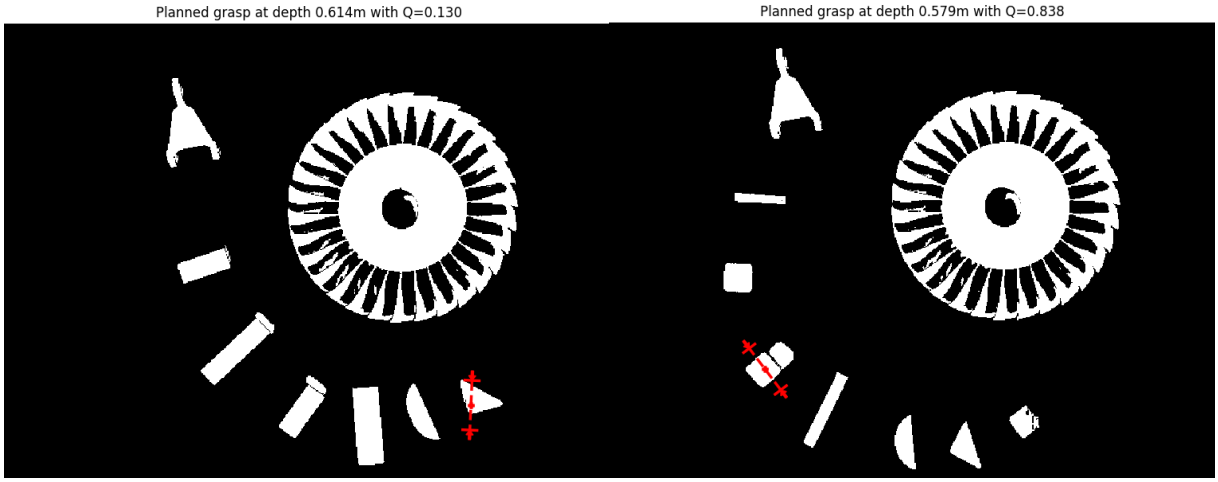
Figure 29: Comparison for the Turbine Support between: a) not feasible grasp position; b) feasible grasp position.



(a) (b)

Source: Author

Figure 30: Comparison for the coloured blocks between: a) not feasible grasp position; b) feasible grasp position.



(a) (b)

Source: Author

6.2 Predictions with High Q-value

The first pair of images in Figure 31 shows the aircraft part lying on the black background. In Figure 31b, it is possible to notice in the binary segmentation image that the indicated grasp position is in a suitable region, although it is slightly oblique in relation to the face of the object, and the Q-value of 70,5% indicates a high probability of grasp success. Whereas in Figure 31a, it is difficult to visualise the grasp position on the object. However, knowing that more white areas indicate higher heights in a depth

image and comparing the depth image to the binary one, it is estimated to be a suitable location to grasp. It must be stressed that the probability of grasping is higher for the segmented image than for the point cloud, indicating that the segmentation provided a more effective prediction due to the improvement in the alignment phase.

Figure 31: Comparison between grasping predictions: a) with depth data; b) with binary image.

(a) (b)

Source: Author.

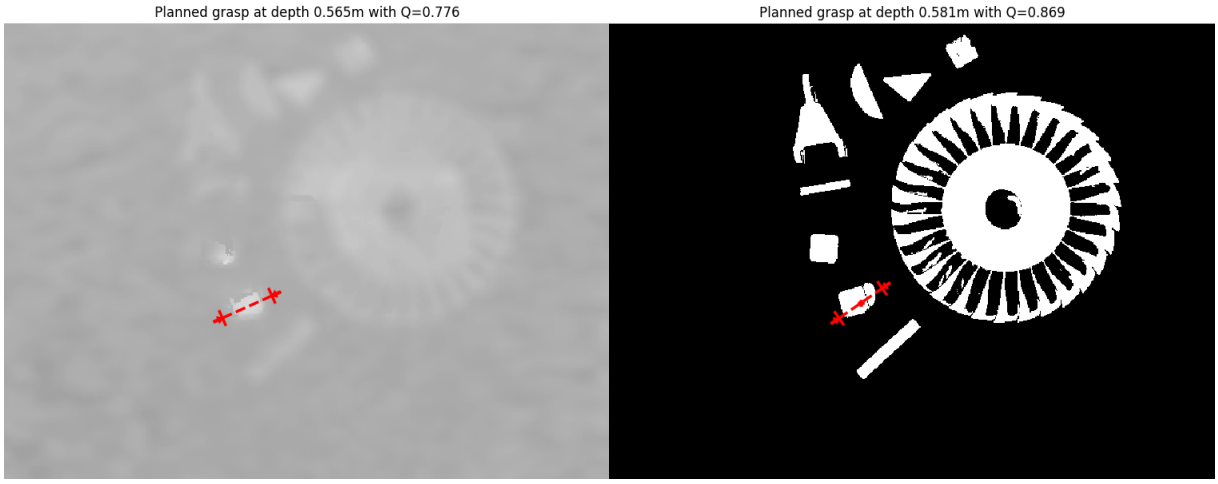
In the second pair of images, Figure 32, it is possible to observe in both images that there was a prediction of grasping in one of the coloured wooden blocks. The Q-value is high due to the simple geometry of the selected object for grasping, which is comparable to those seen during the neural network training. The network estimates the best grasp position among thousands of possibilities in a scene, based on success metrics determined from grasping attempts in a virtual environment used in its training. Thus, it is expected to make better predictions on objects contained in its training database.

Moreover, by comparing the prediction in the image with segmentation, Figure 32b, to the one without segmentation, Figure 32a, it is possible to state again that the segmentation contributed to the improvement in the prediction result.

In the third pair of images, Figure 33, given a cluttered scene, in other words, with the combination of multiple objects, all of them belonging to the IDEA project, a grasping position was identified in the object with simpler geometry, that is, the support with a rectangular shape. To this end, due to the simple geometry, the Q-value estimated was also high.

In Figure 33a, the object was not correctly outlined because of the noisy point

Figure 32: Comparison between grasping predictions: a) with depth data; b) with binary image.

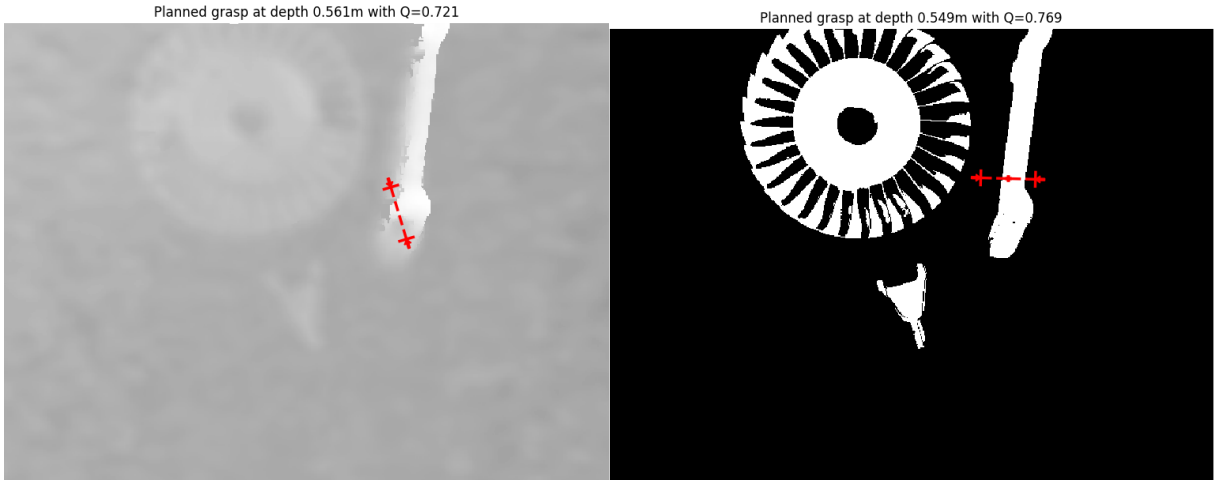


(a) (b)

Source: Author.

cloud, causing a False Positive prediction. It is possible to observe that the position indicated for one of the gripper jaws would collide against the object compared to the corresponding segmentation image in Figure 33b. Hence, the segmentation image notably improved the object identification in the absence of a proper point cloud in this case.

Figure 33: Comparison between grasping predictions: a) with depth data; b) with binary image.



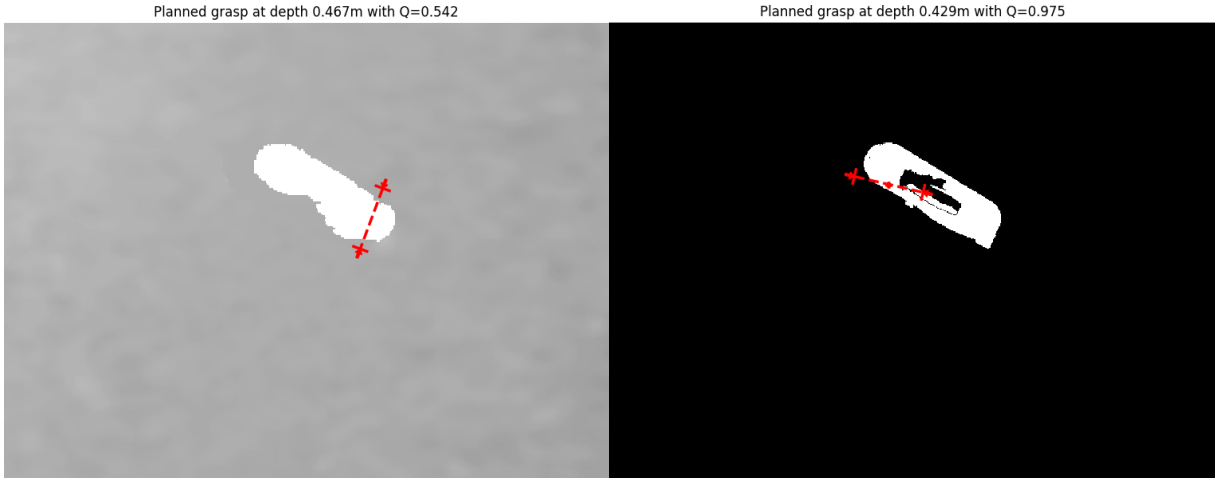
(a) (b)

Source: Author.

Finally, in Figure 34, it is possible to observe the support object of the IDEA project previously presented in Figure 16 as the (VI) object in red, this time standing on its base and therefore visualised from its superior surface. In Figure 34b, it is possible to

see a grasp prediction between the outer part of the object and its central section, which is hollow. While in Figure 34a, the prediction was made for the external edges. Both positions are correct, but the Q-value resulted higher for the segmented, as expected.

Figure 34: Comparison between grasping predictions: a) with depth data; b) with binary image.



(a) (b)

Source: Author.

6.3 Predictions with Low Q-value

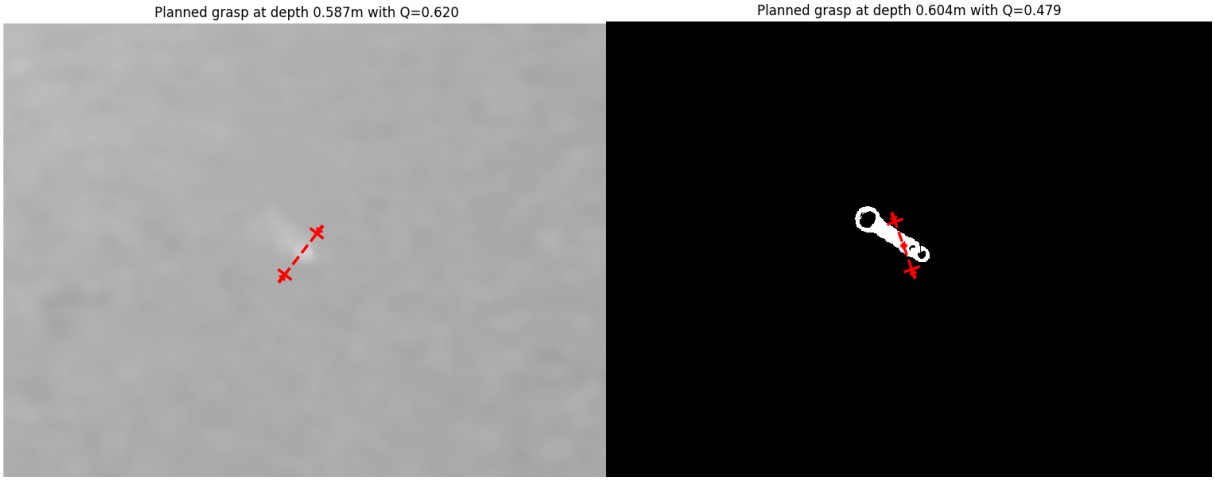
In Figure 35a, it is difficult to identify the estimated position for the grasp without the help of a segmentation mask. However, when compared to Figure 35b, it could be stated that the estimation using only the point cloud is in a correct position, and the Q-value is high, indicating a True Positive prediction.

As for the prediction made in Figure 35b, as explained previously, extremely oblique positions in relation to the object are inadequate because they can cause the object's escape given the attempt to grasp it. Given this fact, considering the threshold of 50%, the Q-value is low, and the estimation is regarded as True Negative.

A feasible grasp position for the aircraft part piece was identified, represented in Figure 36, but the resulting Q-value is extremely low, which can be justified by considering that from this specific point of view, this piece has a different geometry from those identified in the neural network training set and perhaps it was not able to generalize as well as expected.

In Figure 37b, the third pair of images, the prediction was given for the blisk, at a

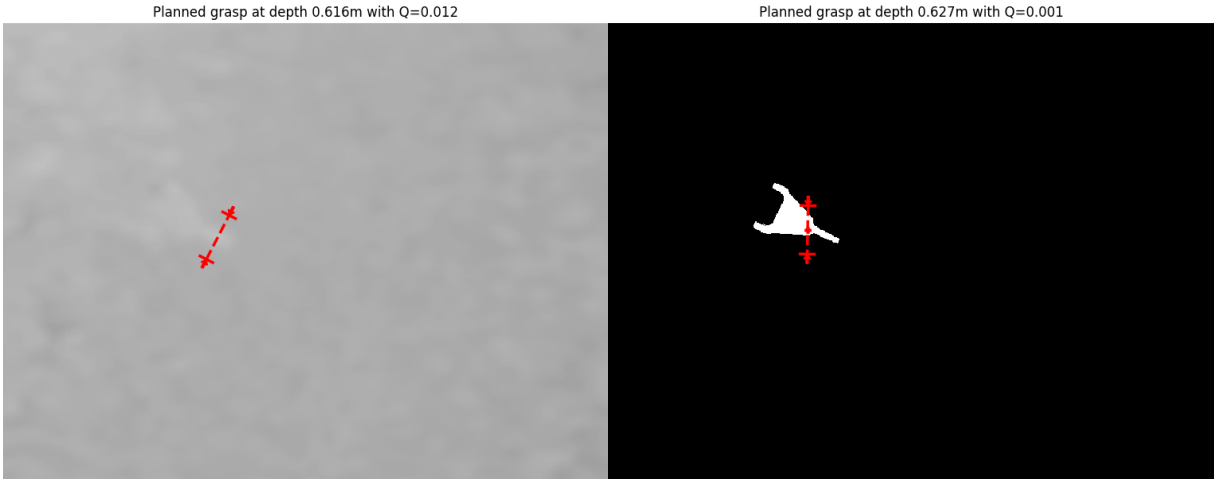
Figure 35: Comparison between grasping predictions: a) with depth data; b) with binary image.



(a) (b)

Source: Author.

Figure 36: Comparison between grasping predictions: a) with depth data; b) with binary image.



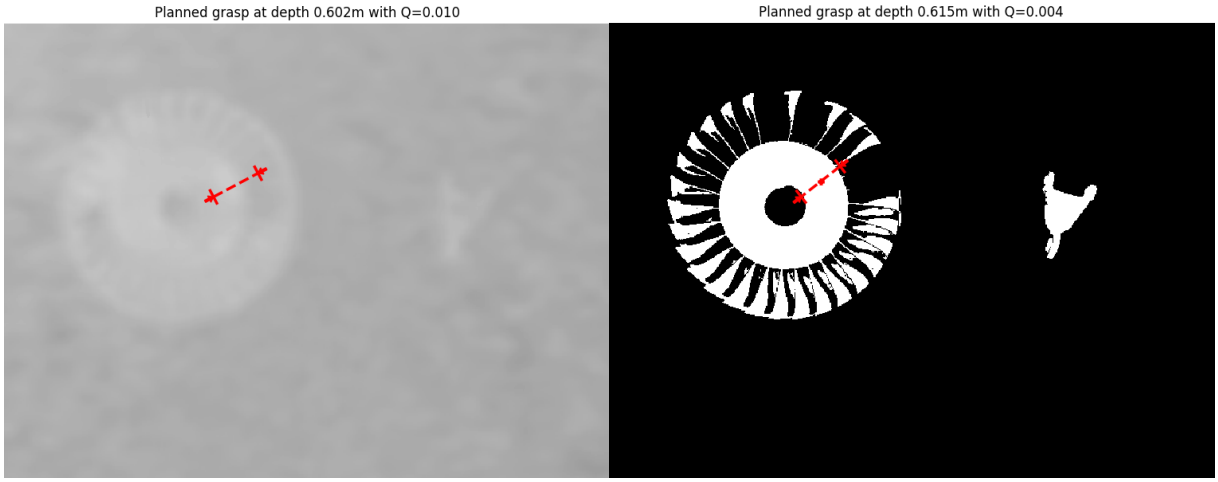
(a) (b)

Source: Author.

point feasible to grasp the object, but the Q-value given is low, indicating a False Negative prediction. In contrast, the prediction in Figure 37a is a True Negative because it would not be possible to grasp the object in the indicated region.

For the following cluttered scene, in Figure 38, it is possible to observe that the prediction was made again for objects with simple geometry, represented by the wooden blocks. However, the Q-value resulted low, with and without the segmentation image, estimating that the indicated position would probably not cause a successful grasp. An explanation for this could be the low quality of the point cloud provided, as shown in

Figure 37: Comparison between grasping predictions: a) with depth data; b) with binary image.

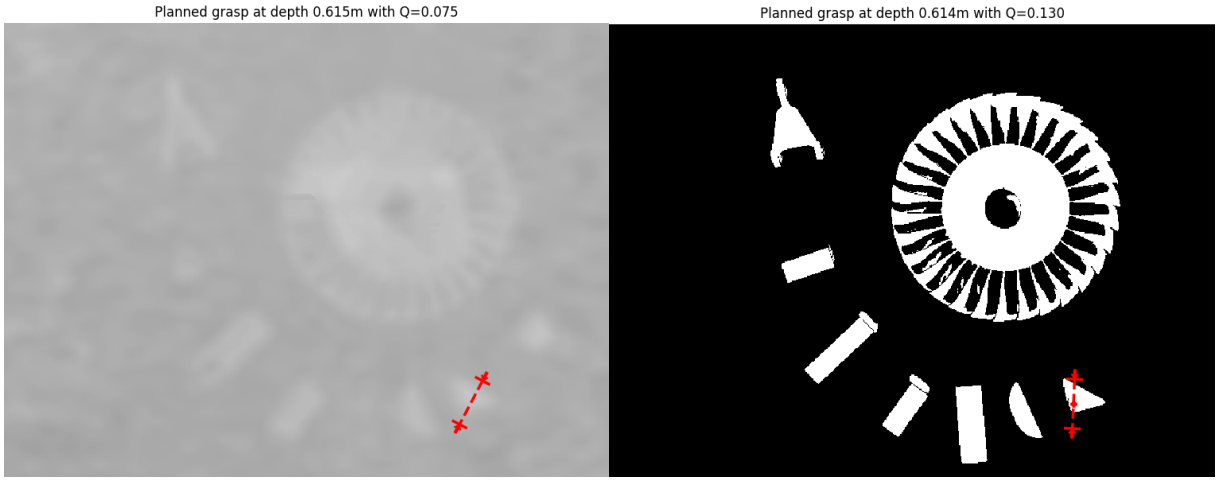


(a) (b)

Source: Author.

Figure 38a. Despite identifying the object, it is probably not defined enough for the network to estimate the grasp quality correctly.

Figure 38: Comparison between grasping predictions: a) with depth data; b) with binary image.



(a) (b)

Source: Author.

The results of the GQ-CNN authors stated that the neural network would generalize properly on new and complex objects, with an 80% success rate and 100% precision, zero false positives, over 50 trials on ten novel test objects. However, it is reasonable to state that the low Q-value happens because the objects used as test set differ from those used during the neural network training.

Moreover, the point cloud obtained with the RealSense camera contains considerable noise, and the neural network was trained for another RGB-D camera model, producing less noisy point clouds.

6.4 Evaluation

Based on the evaluation of the previously exposed results, two Tables were made analysing the general results of the neural network with and without the segmentation image to determine if the addition of this image processing technique contributes towards the improvement of the results.

Table 7 shows the analysis considering 82 predictions using segmentation image, while Table 8 shows the same analysis considering 82 predictions without binary segmentation. When comparing them, it is possible to see that both obtained similar results, but Table 8 shows False Positive results, which could cause collisions. For this reason and to provide better visualization of the results, the use of the binary segmentation image can be justified.

Among the objects previously exposed in Subsection 3.3.3, only the objects denoted as aircraft part, Baby Blisk, colour blocks and Turbine support were used for this evaluation.

Table 7: Confusion matrix for prediction outputs using binary segmentation.

		Grasp Prediction	
		Positive	Negative
Target	Positive	True Positive 14,63%	False Negative 26,83%
	Negative	False Positive 0%	True Negative 58,54%

Source: Author.

Table 8: Confusion matrix for prediction outputs without binary segmentation.

		Grasp Prediction	
		Positive	Negative
Target	Positive	True Positive 23,17%	False Negative 29,27%
	Negative	False Positive 1,22%	True Negative 46,34%

Source: Author.

As seen in Table 7, despite the complexity of the objects used to test the neural network, in 41,46% of them, the grasping position indicated was correct. 26,83% of the results were False Negatives, which means that the given position was appropriate for grasping, but the probability of successfully grasping the object was lower than the threshold defined for a positive Q-value. Moreover, 14,63% of the results were True Positives, in other words, with a high Q-value and indicating a suitable position for grasping the object. This result indicates that despite the complex geometries of the objects, GQCNN was able to predict adequate positions for grasping them. Also, the complexity of identifying the correct point to grasp an object, among millions of possibilities, should be considered. There were 58,54% True Negatives, that is, predictions with low Q-value and that indicated an inadequate grasp position, and there were no False Positives, that is, high Q-values for an inadequate grasping position, which could cause accidents to the robot if there were, therefore the precision was of 100%.

To estimate the objects most identified by the neural network, in the predictions with segmentation image and without it, Table 9 exposes the percentage of times each object was identified. The predictions were made both in scenes with only one object and scenes with several objects. In the two types of prediction, the blisk was the most recognised, which can be justified by its proportionally larger size than other objects, leading to easier detection.

Table 9: Objects detected in cluttered scenes.

Objects Detected in Cluttered Scenes		
Prediction	With segmentation	Without segmentation
Aircraft Part	11,36%	24,44%
Blisk	61,36%	51,11%
Coloured Blocks	13,64%	15,56%
Turbine support	13,64%	8,89%

Source: Author.

However, prediction results for each object must be evaluated because identifying the object is insufficient since the goal is to acquire an adequate position for the grasp. Hence, the results will be divided by object, considering cluttered scenes. Only the outputs using the segmentation image will be presented to avoid an extensive analysis since the quality of the prediction given a segmentation image has already been analysed.

Despite being the most detected object in a cluttered scene, the Baby Blisk did not present predictions with high Q-value, so there were no positive predictions, as shown in

Table 10. However, more than half of the predictions were False Negatives. Thus, despite conferring a grasp position to the blisk, the probability of grasp conferred by the neural network was low.

Table 10: Confusion matrix of prediction outputs using binary segmentation and the blisk in a cluttered scene

	Grasp Prediction		
		Positive	Negative
Target	Positive	True Positive 0%	False Negative 22,22%
	Negative	False Positive 0%	True Negative 77,78%

Source: Author.

Following this, both the coloured blocks and the turbine support had the same detection percentage, as shown in Table 11 and Table 12, respectively. There were more positive results for the coloured blocks, being 66,67% of True Positives results to 33,33% of True Negatives results, which is justified by the simple geometry of the objects and assimilation to the objects used during the neural network training. While for the Turbine Support, more than half of the results were False Negatives, and 33,33% were True Positives. For the case of the Turbine Support, the positions identified as True Positives were obtained when it stood on its base or its side, exposing its thinner rectangular view to the camera and facilitating grasp prediction. When laid down, this object has a diameter larger than the maximum allowed by the robot's gripper. Consequently, it is not possible to estimate good positions for grasping.

Table 11: Confusion matrix of prediction outputs using binary segmentation and the colour blocks in a cluttered scene

	Grasp Prediction		
		Positive	Negative
Target	Positive	True Positive 66,67%	False Negative 0%
	Negative	False Positive 0%	True Negative 33,33%

Source: Author.

Finally, for the aircraft part, 80% of the predictions were obtained as False Negatives. In other words, the position estimated by the network was correct, yet the calculated probability of a successful grasp was low, as exposed in Table 13.

Table 12: Confusion matrix of prediction outputs using binary segmentation and the Turbine Support in a cluttered scene

	Grasp Prediction		
Target	Positive	Positive True Positive 33,33%	Negative False Negative 66,67%
	Negative	False Positive 0%	True Negative 0%

Source: Author.

Table 13: Confusion matrix of prediction outputs using binary segmentation and the aircraft part in a cluttered scene

	Grasp Prediction		
Target	Positive	Positive True Positive 0%	Negative False Negative 80,0%
	Negative	False Positive 0%	True Negative 20,0%

Source: Author.

The Dex-Net's 4.0 dataset [1] contains objects from the 3D Net [42] database. The success of the prediction on unknown objects could be explained by comparing the shape of the objects in the training dataset, as mentioned in Subsection 3.3.3. Their similarities are one of the reasons that justify the capability of recognizing new objects.

As previously commented, the RGB-D sensor produces low-quality point clouds, and this causes failures. As can be noted, in some cases, there were failures in the prediction given only the point cloud, without the binary image. Therefore, it would be interesting to use other cameras for comparison in future works.

Another limitation is the gripper aperture diameter, which was configured before making predictions. It would be better to adopt a wider aperture, considering the size of the blisk and the diameter of the turbine support. The network was not trained for larger jaw openings, and its authors do not indicate test configurations different from the ones used by them. Therefore, to adopt a gripper with a larger aperture diameter, the net would have to be fully retrained, as well as the training dataset would have to be generated for that specific aperture diameter.

In order to achieve more True Positives, a default positioning of the objects can be established based on the poses that yielded True Positives for each object. There is no means to precisely correct the False Negatives, therefore transfer learning must be done using a dataset synthetically created with the IDEA project objects, whose generation

process is given for all possible positions and grasps for each object. Therefore, the fastest solution to be implemented is to adopt the positions that lead to True Positives for each object.

7 CONCLUSION AND OUTLOOK

This work aimed to analyse the capacity of a pre-trained neural network to generalise in novel objects to develop a system for robotic grasping of aeronautical parts in a production line to meet a requirement of the IDEA Project in the WZL institute. To this end, a data acquisition, processing, and analysis system were developed to determine grasping positions for the observed objects. In this study, the objects used were aeronautical parts, which have complex geometries and are distinct from objects used in everyday life.

A CNN called GQ-CNN was chosen among the available methods to determine grasp positions. Their pre-trained model was used to make predictions from the data obtained in this work. The neural network's output is a success metric, which predicts the probability that the robot will grasp the object for an antipodal point for the robot's gripper. Therefore, among the available sensors for data acquisition, the IntelRealsense D435 RGB-D camera was adopted, which is used in projects with similar characteristics. The RGB-D camera made it possible to acquire colour images and depth maps.

The data acquisition and processing were implemented using code in the Python programming language. It was stated that using filters such as spatial and temporal filtering makes it possible to improve the quality of the depth map acquired by the camera. Furthermore, foreground masking and border following filters allowed to obtain binary segmentation images from the colour images, thus facilitating the alignment phase for the prediction by the neural network and consequently increasing the probability of accuracy in the prediction of the grasp position.

A total of 164 predictions were performed, half of them made only with the point cloud obtained with the RGB-D camera and the other half also using the binary segmentation image. When comparing the results, there was a noticeable improvement in the object recognition when using binary segmentation, which was translated into an increase in the Q-value quality metric obtained and adequate positions for grasping the objects.

Therefore, a confusion matrix evaluation was made from the binary segmentation image predictions. This evaluation was done qualitatively, in other words, without performing physical grasping tests with the robot. From this analysis, despite the geometrical complexity of the objects used in the project, it was possible to obtain 41.46% of the pre-

dictions in suitable positions for grasping, and there were no false positives, which means that the network did not predict positions that would be inadequate for grasping the objects. True Positives corresponded to 14.63% of the results, indicating that the network was not able to generalize as well as defended by its authors. It can be attributed to the complexity of the tested objects, which differ from the classes seen during the networks' training.

8 FUTURE WORK

Considering the limitations regarding the camera used, another model of RGB-D sensor, with the same quality as the one used by the authors of the GQ-CNN project, could be tested in order to obtain better point clouds and consequently improve the grasp prediction.

In case the GQ-CNN still does not generalize well to the IDEA's objects, the author suggests using the method presented by Mahler et al. [16] in Dex-Net 1.0 in order to build a synthetic dataset with the IDEA project objects and use it during transfer learning with FC-GQ-CNN presented by Mahler et al. in [24]. Afterwards, the pre-trained model and the one trained with the IDEA dataset could be compared.

Also, it would be interesting to develop a ROS integration between the camera, GQ-CNN and the robot and evaluate the results on a real robot. Finally, a wireless communication system could be implemented for the camera to improve robot mobility.

REFERENCES

[1] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, K. Goldberg, Learning ambidextrous robot grasping policies, Science Robotics 4 (26). doi:10.1126/scirobotics.aau4984.

[2] J. Mahler, R. Platt, A. Rodriguez, M. Ciocarlie, A. Dollar, R. Detry, M. A. Roa, H. Yanco, A. Norton, J. Falco, K. v. Wyk, E. Messina, J. , D. Morrison, M. Mason, O. Brock, L. Odhner, A. Kurenkov, M. Matl, K. Goldberg, Guest editorial open discussion of robot grasping benchmarks, protocols, and metrics, IEEE Transactions on Automation Science and Engineering 15 (4) (2018) 1440–1442. doi:10.1109/TASE.2018.2871354.

[3] J. Mahler, K. Goldberg, Learning deep policies for robot bin picking by simulating robust grasping sequences, in: Conference on robot learning, PMLR, 2017, pp. 515–524.

[4] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[5] L. Deng, D. Yu, Deep learning: Methods and applications, Found. Trends Signal Process. 7 (3–4) (2014) 197–387. doi:10.1561/2000000039.

[6] J. Teuwen, N. Moriakov, Chapter 20 - convolutional neural networks, in: S. K. Zhou, D. Rueckert, G. Fichtinger (Eds.), Handbook of Medical Image Computing and Computer Assisted Intervention, The Elsevier and MICCAI Society Book Series, Academic Press, 2020, pp. 481–501.

[7] C.-C. J. Kuo, Understanding convolutional neural networks with a mathematical model, Journal of Visual Communication and Image Representation 41 (2016) 406–413.

[8] B. P. Lathi, Linear Systems and Signals, 2nd Edition, Oxford University Press, Inc., USA, 2009.

[9] V. Dumoulin, F. Visin, A guide to convolution arithmetic for deep learning, arXiv preprint arXiv:1603.07285.

[10] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang, et al., Softmax regression, accessed: 2022-02-23.
URL http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/

[11] N. Silver, K. Simonson, The Signal and the Noise: Why So Many Predictions Fail—But Some Don't, Springer, 2013.

[12] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, K. Goldberg, Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics (2017). arXiv:1703.09312.

[13] L. Manuelli, W. Gao, P. Florence, R. Tedrake, kpam: Keypoint affordances for category-level robotic manipulation (2019). arXiv:1903.06684.

[14] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012.

[15] K. Akiyama, A. Alberdi, W. Alef, K. Asada, R. Azulay, A.-K. Baczko, D. Ball, M. Baloković, J. Barrett, D. Bintley, et al., First m87 event horizon telescope results. iv. imaging the central supermassive black hole, The Astrophysical Journal Letters 875 (1) (2019) L4.

[16] J. Mahler, F. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kroeger, J. Kuffner, K. Goldberg, Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a Multi-Armed Bandit model with correlated rewards, IEEE International Conference on Robotics and Automation (2016) 1957–1964.

[17] J. Mahler, M. . Matl, X. Liu, A. Li, D. Gealy, K. Goldberg, Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning, Proceedings of the IEEE (2018) 1–8.

[18] M. A. Roa, R. Suárez, Grasp quality measures: review and performance, Vol. 38, Springer, 2015, pp. 65–88.

[19] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, Y. Matsuoka, Physical human interactive guidance: Identifying grasping principles from human-planned grasps, IEEE Transactions on Robotics 28 (4) (2012) 899–910.

[20] S. Kumra, C. Kanan, Robotic grasp detection using deep convolutional neural networks, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2017, pp. 769–776.

[21] M. Gualtieri, A. ten Pas, R. Platt, Pick and Place without geometric object models, IEEE International Conference on Robotics and Automation (ICRA) (2018) 7433–7440.

[22] D. Morrison, P. Corke, J. Leitner, Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach, Robotics: Science and Systems XIV (2018) 1–10.

[23] J. Cook, V. Ramadas, When to consult precision-recall curves, The Stata Journal 20 (1) (2020) 131–148. doi:10.1177/1536867X20909693.

[24] V. Satish, J. Mahler, K. Goldberg, On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks, IEEE Robotics and Automation Letters 4 (2) (2019) 1357–1364. doi:10.1109/LRA.2019.2895878.

[25] J. Wu, Introduction to convolutional neural networks, National Key Lab for Novel Software Technology. Nanjing University. China 5 (23) (2017) 495.

[26] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network (2015). arXiv:1505.00853.

[27] S. Theodoridis, Chapter 18 - neural networks and deep learning, in: S. Theodoridis (Ed.), Machine Learning (Second Edition), second edition Edition, Academic Press, 2020, pp. 901–1038.

[28] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034. doi:10.1109/ICCV.2015.123.

[29] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, D. Quillen, Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, The International Journal of Robotics Research 37 (4-5) (2018) 421–436. doi:10.1177/0278364917710318.

[30] M. Schwarz, A. Milan, A. S. Periyasamy, S. Behnke, Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter, The International Journal of Robotics Research 37 (4-5) (2017) 437–451. doi:10.1177/0278364917713117.

[31] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[32] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, K. Goldberg, Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 7283–7290.

[33] M. Badamchizadeh, A. Aghagolzadeh, Comparative study of unsharp masking methods for image enhancement, in: Third International Conference on Image and Graphics (ICIG'04), 2004, pp. 27–30. doi:10.1109/ICIG.2004.50.

[34] C.-F. W. Ron Kikinis, H. Knutsson, Adaptive image filtering, in: I. N. BANKMAN (Ed.), Handbook of Medical Imaging, Biomedical Engineering, Academic Press, San Diego, 2000, pp. 19–31.

[35] S. Suzuki, K. be, Topological structural analysis of digitized binary images by border following, Computer Vision, Graphics, and Image Processing 30 (1) (1985) 32–46.

[36] A. Singh, J. Sha, K. S. Narayan, T. Achim, P. Abbeel, Bigbird: A large-scale 3d database of object instances, in: 2014 IEEE international conference on robotics and automation (ICRA), IEEE, 2014, pp. 509–516.

[37] Depth camera d435, https://www.intelrealsense.com/depth-camera-d435/, accessed: 2022-02-23.

[38] Nep 1 — a simple file format for numpy arrays kernel description, https://bit.ly/2O5xiN8, accessed: 2019-11-11.

[39] M. Guo, D. V. Gealy, J. Liang, J. Mahler, A. Goncalves, S. McKinley, J. A. Ojea, K. Goldberg, Design of parallel-jaw gripper tip surfaces for robust grasping, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 2831–2838. doi:10.1109/ICRA.2017.7989330.

[40] Zhengyou Zhang, Flexible camera calibration by viewing a plane from unknown orientations, in: Proceedings of the Seventh IEEE International Conference on Computer Vision, Vol. 1, 1999, pp. 666–673 vol.1. doi:10.1109/ICCV.1999.791289.

[41] A. Grunnet-Jepsen, D. Tong, Depth post-processing for intel® realsense™ d400 depth cameras, https://dev.intelrealsense.com/docs/depth-post-processing, accessed: 2022-02-23.

[42] W. Wohlkinger, A. Aldoma, R. B. Rusu, M. Vincze, 3dnet: Large-scale object class recognition from cad models, in: 2012 IEEE international conference on robotics and automation, IEEE, 2012, pp. 5384–5391.

A CONFIGURATION FILE FOR RGB-D DATA ACQUISITION AND PRE-PROCESSING

```yaml
processing_parameters:
  depth_thresh_low: 0.3 # everything between 0.3 and 0 should be noisy, therefore is avoided
  depth_thresh_upper: 0.6 # distance between workspace and camera
  spatial_alpha: 0.6
  spatial_delta: 8
  temp_alpha: 0.5
  temp_delta: 20
  depth_scale: 2
  decimation_magnitude: 1.0
  spatial_magnitude: 2.0

high_resolution:
  height: 720   # high resolution
  width: 1280
  fps: 30
  camera_intrinsics: '../../model/calib/realsense/realsense_depth_intrinsics_high.intr'

low_resolution:
  height: 480   # high resolution
  width: 640
  fps: 30
  camera_intrinsics: '../../model/calib/realsense/realsense_depth_intrinsics_low.intr'

save: 1
frame: 'realsense_overhead'
output_dir_raw: '../../data/raw'
output_dir_processed: '../../data/processed'
json_file: '../../model/calib/realsense/realsense_viewer.json'
```

B CODE FOR RGB-D DATA ACQUISITION AND PRE-PROCESSING

```
"""
Author: @ltorquato4
Class to interact with ONE Intel RealSense camera D400 series
"""

import logging
import time
import cv2
import numpy as np
import pyrealsense2 as rs
from perception import CameraIntrinsics
from autolab_core import YamlConfig


class RealSense:

    def __init__(self, cfg=None, filter_depth=True, resolution='high'):

        self._running = None

        if cfg is None:
            cfg = 'cfg/realsense.yaml'

        # read cfg file
        self._config = YamlConfig(cfg)

        ids = self.discover_cams()
        self.id = ids[0]
        self.json_file = self._config['json_file']
        self._resolution = resolution
        if self._resolution == 'low':
            resolution = self._config['low_resolution']
            self._height = resolution['height']
            self._width = resolution['width']
            self._fps = resolution['fps']
            self._camera_intrinsics = resolution['camera_intrinsics']
        else:
            resolution = self._config['high_resolution']
            self._height = resolution['height']
            self._width = resolution['width']
            self._fps = resolution['fps']
            self._camera_intrinsics = resolution['camera_intrinsics']

        self._output_dir_raw = self._config['output_dir_raw']
        self._output_dir_processed = self._config['output_dir_processed']
```

```
46
47          self._filter_depth = filter_depth
48
49          self._frame = self._config['frame']
50
51          if self._frame is None:
52              self._frame = 'realsense_overhead'
53          self._color_frames = '%s_color' % self._frame
54
55          # real sense objects
56          self._pipe = rs.pipeline()
57          self._cfg = rs.config()
58          self._align = rs.align(rs.stream.color)
59
60          # camera parameters
61          self._intrinsics = np.eye(3)
62
63          # save data with day and hour
64          self._save = self._config['save']
65          self._filename = time.strftime("%d%m%Y-%H%M%S")
66
67      @staticmethod
68      def discover_cams():
69          """Returns a list of the ids of all cameras connected via USB"""
70          ctx = rs.context()
71          ctx_devs = list(ctx.query_devices())
72          ids_ = []
73          for i in range(ctx.devices.size()):
74              ids_.append(ctx_devs[i].get_info(rs.camera_info.serial_number))
75          assert ids_, "[!] No camera detected."
76          return ids_
77
78      def _config_pipe(self):
79          """Configures the pipeline to stream color and depth.
80          """
81          self._cfg.enable_device(self.id)
82
83          self._cfg.enable_stream(
84              rs.stream.color,
85              self._width,
86              self._height,
87              rs.format.bgr8,
88              self._fps
89          )
90
91          self._cfg.enable_stream(
92              rs.stream.depth,
93              self._width,
```

```
94              self._height,
95              rs.format.z16,
96              self._fps
97          )
98
99      def _set_intrinsics(self):
100         """Read the intrinsics matrix from the depth stream.
101         """
102         self._profile = self._pipe.get_active_profile()
103         strm = self._profile.get_stream(rs.stream.depth)
104         depth_profile = rs.video_stream_profile(strm)
105         obj = depth_profile.get_intrinsics()
106         self._intrinsics[0, 0] = obj.fx
107         self._intrinsics[1, 1] = obj.fy
108         self._intrinsics[0, 2] = obj.ppx
109         self._intrinsics[1, 2] = obj.ppy
110
111     def _set_depth_scale(self):
112         """Retrieve the scale of the depth sensor.
113         """
114         depth_sensor = self._profile.get_device().first_depth_sensor()
115         self._depth_scale = depth_sensor.get_depth_scale()
116
117     @property
118     def intrinsics(self):
119         """:obj:`CameraIntrinsics` : The camera intrinsics for the RealSense color camera.
120         """
121         return CameraIntrinsics(
122             self._frame,
123             round(self._intrinsics[0, 0], 1),
124             round(self._intrinsics[1, 1], 1),
125             round(self._intrinsics[0, 2], 1),
126             round(self._intrinsics[1, 2], 1),
127             skew=0.0,
128             height=self._height,
129             width=self._width
130         )
131
132     def __del__(self):
133         """Automatically stop the sensor for safety.
134         """
135         if self.is_running:
136             self.stop()
137
138     @property
139     def is_running(self):
140         """bool : True if the stream is running, or false otherwise.
141         """
```

```
142         return self._running
143
144     @property
145     def frame(self):
146         """:obj:`str` : The reference frame of the sensor.
147         """
148         return self._frame
149
150     @property
151     def color_frame(self):
152         """:obj:`str` : The reference frame name of the color sensor.
153         """
154         return self._color_frames
155
156     def start(self):
157         """Start the sensor.
158         """
159         try:
160             self._config_pipe()
161             self._pipe.start(self._cfg)
162             # store intrinsics and depth scale
163             self._set_intrinsics()
164             self._set_depth_scale()
165             self._load_settings_json(self.json_file)
166
167             # skip few frames to give auto-exposure a chance to settle
168             for _ in range(30):
169                 self._pipe.wait_for_frames()
170
171             self._running = True
172         except RuntimeError as e:
173             print(e)
174
175     def stop(self):
176         """Stop the sensor.
177         """
178         if not self._running:
179             logging.warning('Realsense not running.')
180             return False
181
182         self._pipe.stop()
183         self._running = False
184         return True
185
186     @staticmethod
187     def _to_numpy(frame, dtype):
188         arr = np.asanyarray(frame.get_data(), dtype=dtype)
189         return arr
```

```
190
191     def _load_settings_json(self, json_file):
192         """
193         Load the settings stored in the JSON file
194
195         """
196         with open(json_file, 'r') as file:
197             json_text = file.read().strip()
198
199         device = self._profile.get_device()
200         advanced_mode = rs.rs400_advanced_mode(device)
201         advanced_mode.load_json(json_text)
202
203     def _filter_depth_frame(self, depth_frame):
204         """
205         Applies post-processing filters to the depth frame received
206         :param depth_frame: frame to post-process
207         :return: depth_filtered
208         """
209
210         processing_parameters = self._config['processing_parameters']
211         depth_thresh_low = processing_parameters['depth_thresh_low']
212         depth_thresh_upper = processing_parameters['depth_thresh_upper']
213         spatial_alpha = processing_parameters['spatial_alpha']
214         spatial_delta = processing_parameters['spatial_delta']
215         temporal_alpha = processing_parameters['temp_alpha']
216         temporal_delta = processing_parameters['temp_delta']
217         decimation_magnitude = processing_parameters['decimation_magnitude']
218         spatial_magnitude = processing_parameters['spatial_magnitude']
219
220         decimation_filter = rs.decimation_filter()
221         threshold_filter = rs.threshold_filter()
222         temporal_filter = rs.temporal_filter()
223         hole_filling = rs.hole_filling_filter()
224         spatial_filter = rs.spatial_filter()
225
226         filter_magnitude = rs.option.filter_magnitude
227         filter_smooth_alpha = rs.option.filter_smooth_alpha
228         filter_smooth_delta = rs.option.filter_smooth_delta
229
230         min_distance = rs.option.min_distance
231         max_distance = rs.option.max_distance
232
233         # define post-processing filters
234         decimation_filter.set_option(filter_magnitude, decimation_magnitude)
235         spatial_filter.set_option(filter_magnitude, spatial_magnitude)
236         spatial_filter.set_option(filter_smooth_alpha, spatial_alpha)
237         spatial_filter.set_option(filter_smooth_delta, spatial_delta)
```

```
238             temporal_filter.set_option(filter_smooth_alpha, temporal_alpha)
239             temporal_filter.set_option(filter_smooth_delta, temporal_delta)
240             threshold_filter.set_option(min_distance, depth_thresh_low)
241             threshold_filter.set_option(max_distance, depth_thresh_upper)
242 
243             out = decimation_filter.process(depth_frame)
244             out = threshold_filter.process(out)
245             out = spatial_filter.process(out)
246             out = temporal_filter.process(out)
247             depth_filtered = hole_filling.process(out)
248             return depth_filtered
249 
250         def _read_color_and_depth_image(self):
251             """
252             Read a color and depth image from the device.
253             :return: color_image, depth_image
254             """
255             frames = self._pipe.wait_for_frames()
256 
257             frames = self._align.process(frames)
258 
259             depth_frame = frames.get_depth_frame()
260             color_frame = frames.get_color_frame()
261 
262             if not depth_frame or not color_frame:
263                 logging.warning('Could not retrieve frames.')
264                 return None, None
265 
266             if self._filter_depth:
267                 depth_frame = self._filter_depth_frame(depth_frame)
268 
269             # convert to numpy arrays
270             depth_image = self._to_numpy(depth_frame, dtype=np.float32)
271             color_image = self._to_numpy(color_frame, dtype=np.uint8)
272 
273             # convert depth to meters
274             depth_image = depth_image * self._depth_scale
275 
276             return color_image, depth_image
277 
278         def frames(self):
279             """Retrieve a new frame from the RealSense and save a depth image,
280             color image and depth intrinsics
281 
282             Returns
283             -------
284             : color image, depth image and the current frame for reference.
285 
```

```
286         Raises
287         ------
288         RuntimeError
289             If the RealSense stream is not running.
290         """
291         raw_color_im, raw_depth_im = self._read_color_and_depth_image()
292 
293         if self._save:
294             np.save('%s/%s_depth_image.npy' % (self._output_dir_raw, self._filename),
295                                                raw_depth_im)
296             cv2.imwrite('%s/%s_color_image.png' % (self._output_dir_raw, self._filename),
297                                                    raw_color_im)
298         self.intrinsics.save(self._camera_intrinsics)
299 
300         return raw_color_im, raw_depth_im
```

C CODE TO START REALSENSE

```
"""
Can be executed from the global command line
"""

from .realsense_sensor import RealSense

if __name__ == '__main__':

    # setup sensor
    sensor = RealSense()
    sensor.start()
    color, depth = sensor.frames()
    intrinsics = sensor.intrinsics
    sensor.stop()
```

D CONFIGURATION FILE FOR BINARY SEGMENTATION

```
segmask:
  low_gray: 100
  upper_gray: 250
  distance_thresh: 1000
  area_thresh: 300
  depth_thresh_low: 0.3
  depth_thresh_upper: 0.6
  output_dir_processed: 'data/processed'
```

E CODE FOR BINARY SEGMENTATION

```
"""
Author: @ltorquato
Class to segment color images from the RealSense D400 series.
The output can be used as --segmask input for the GQCNN model.
"""

from autolab_core import YamlConfig
from perception import ColorImage, DepthImage
import cv2
import numpy as np
import argparse
import time


class BinarySegmentation:

    def __init__(self, cfg=None):

        if cfg is None:
            cfg = 'cfg/segmask.yaml'
        # read cfg file
        self._config = YamlConfig(cfg)

        self._segmask = self._config['segmask']
        self._low_gray = self._segmask['low_gray']
        self._upper_gray = self._segmask['upper_gray']
        self._distance_thresh = self._segmask['distance_thresh']
        self._area_thresh = self._segmask['area_thresh']
        self._output_dir_processed = self._segmask['output_dir_processed']
        self._filename = time.strftime("%d%m%Y-%H%M%S")

    @staticmethod
    def _to_numpy(frame, dtype):
        arr = np.asanyarray(frame.get_data(), dtype=dtype)
        return arr

    def color_mask(self, path_to_image, path_to_background_image):
        """
        Takes a color image and a background image to subtract from the image
        and convert it to a binary image.
        Save the binary image in the same dir as the input
        :param: path_to_image, path_to_background_image
        :return: color_mask
        """

```

```
46          background_image = cv2.imread(path_to_background_image)
47          color_image = cv2.imread(path_to_image)
48
49          color_image = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)
50          background_image = cv2.cvtColor(background_image, cv2.COLOR_BGR2GRAY)
51
52          subtract = color_image - background_image
53          subtract_image_filtered = cv2.inRange(subtract, self._low_gray, self._upper_gray)
54          subtract_image_filtered = self._to_numpy(subtract_image_filtered, dtype=np.uint8)
55
56          subtract_image_filtered = ColorImage(subtract_image_filtered)
57          binary = subtract_image_filtered.to_binary()
58          color_mask = binary.prune_contours(area_thresh=self._area_thresh,
59                                             dist_thresh=self._distance_thresh)
60
61          cv2.imwrite('%s/%s_color_mask.png' % (self._output_dir_processed, self._filename),
62                                             color_mask._image_data())
63
64          return color_mask
65
66
67  if __name__ == '__main__':
68
69      parser = argparse.ArgumentParser(
70          description="Run segmentation mask")
71      parser.add_argument("--color_image",
72                          type=str,
73                          help="path to color image")
74      parser.add_argument("--background",
75                          type=str,
76                          help="path to background image")
77      args = parser.parse_args()
78      color_image = args.color_image
79      background = args.background
80
81      bs = BinarySegmentation()
82
83      color_segmask = bs.color_mask(color_image, background)
```