TRAINING NEURAL NETWORKS FOR VISUAL SERVOING

Vadim Atlassov, Bachelor of Engineering

Submitted in fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering

NAZARBAYEV UNIVERSITY

School of Engineering and Digital Sciences Department of Electrical and Computer Engineering Nazarbayev University

Supervisors: Almas Shintemirov, Berdakh Abibullaev

May 2023

DECLARATION

I hereby, declare that this manuscript, entitled “Training Neural Networks for Visual Servoing”, is the result of my own work except for quotations and citations which have been duly acknowledged.
I also declare that, to the best of my knowledge and belief, it has not been previously or concurrently submitted, in whole or in part, for any other degree or diploma at Nazarbayev University or any other national or international institution.

[illegible]

Name: Vadim Atlassov
Date: 17.03.23

Abstract

Visual servoing is a technique which uses feedback from vision sensor to dynamically manipulate the joints of the robot for motion and predicting required posture. The classical visual servoing applies several cameras and computer vision techniques for coordinating the motions of the robot. Therefore, it heavily relies on algorithms of feature extraction and tracking of coordinates position, processing visual features of the environment. The initial attempts of applying revolution of the computer vision, deep learning and convolutional neural networks were used in 2018 and achieved great results in prediction of posture of the robot on the image. In this thesis project I propose potential models which can be applicable in visual servoing without support of direct and classical methods of visual servoing and trained on synthetic dataset, which could be useful in diminishing robot hours. The results have shown great adaptability and resilience for fluctuations in images. Although the training process requires protracted time, the final model of the CNN with regressor output can accurately predict the pose of the robot both in output value positions and in simulation.

Acknowledgments

I would like to express my deepest and sincere appreciation to my family and fiancée for their support in my intensive, laborious path of study. I am grateful and value their care, I owe them everything.

I am deeply indebted to Dr. Shintemirov, who provided me the chance to work with him and agreed to be my supervisor. I thank him for finding time to supervise me during his research visit abroad. Thank also to my cat, Luffy, who made me laugh and my stressful schedule easier and helped me to handle pressure from the study.

Finally, I want to sincerely thank Allah, who blessed me with opportunity to learn invaluable knowledge at Nazarbayev University, who provided me with strength and persistence to finish my master’s degree.

I hope that this small, coarse piece of scientific work will become starting point to the life journey of researcher.

Chapter 1 - Introduction

Visual servoing is a technique which uses feedback from vision sensor to dynamically manipulate the joints of the robot for motion and predicting required posture. The classical visual servoing applies several cameras and computer vision techniques for coordinating the motions of the robot. Therefore, it heavily relies on algorithms of feature extraction and tracking of coordinates position, processing visual features of the environment. The features represent points, lines, corners of the objects. In recent time there was a significant progress in computing input features, and closely related success of CNNs also impacted on robotics field of visual servoing. The new approach of direct visual servoing (DVS) was proposed by multiple sources [1], [2], [3]. The concept of direct visual servoing, as authors of [1] stated, uses direct signals of input almost without further preprocessing of signals. The direct visual servoing includes projective information for computation of control error by comparing two images, current and referenced. This method based on function of camera position and translation error between two frames. It is also useful that control error law will be isomorphic to the camera pose in point where equilibrium = 0. One of the important improvements of direct visual servoing that it generalizes control error for objects of different shapes. the main drawback of DVS was that it had small convergence domain compared to classical techniques, which is due to the high non-linearities of the cost function to be minimized.

Besides recent DVS, there are two well-known techniques for visual servoing, they differ on pose estimation and using space for object details extraction: position-based and image-based (PBVS) [4]. The position-based visual servoing aims to optimize the difference between two positions of the arm. The position-based visual servoing stabilize the robot pose through reconstruction of information from images. The input data include information about metric model of the required object, coordinates of robot and camera position. These inputs are significant in matter of accuracy as visual servoing error will be increased and lose focus on the object in case of bad calibration. The main issue of this method is lower stability and big difference between required position and computed position. To remedy the problem of stability usually one should be able to reconstruct the control error, build control system in form of error-response.

Alternative approach, image-based visual servoing applies 2D coordinates and input to the dynamic control system consists of image features. These features camera and program extracts from images of the target. Secondly system builds interaction matrix which can translate the space of the image to the system of robot and compute control law for motion. This method is complicated by choosing and exporting image features. There are constraints of insufficiency of image features. Additionally, construction of interaction matrix is also complex process in real-time experiments, as motions of the robot towards right position have many nonlinearities. Estimation of the interaction matrix for the pose demands time for calculation. Although stability of this method is similar to the former approach, and was measured by Lyapunov law, IBVS does not rely on intrinsic parameters of the camera [5]. Nevertheless, one of the novel approaches uses similar techniques as position-based methods, but accepts not coordinates of the object from the camera, but considers image space as a whole.

Convolutional neural networks (CNN) has great performance in object detection, classification and identification of the object on the picture, camera manipulation for surveillance. Recently, CNN also found its application in visual servoing systems to resolve problem with building interaction matrix and requirement of features, and problem with light perturbations. The uniqueness of the CNN approach that it can independently extract features from image dataset and adapt nonlinear features from 2D image space to coordinate space of the robot.

CNN is superior in comparison with other methods in terms of precision of the prediction of pose and had greater area of coverage. I think that CNN in visual servoing is a prospective novel combination of two different disciplines and requires research, experimentation. The latest pieces of research highlighted problems with image positioning due to light distortion and brightness fluctuation of the image. Therefore it is possible to compare different DNN models under this condition. I believe that this study can stimulate improvement of visual servoing techniques and evaluate the ability of the CNN. This thesis project will contribute to the research of application of CNN in visual servoing and also to the CNN domain for training networks with synthetic data.

1.1 - Deep learning

The deep learning field emerged in recent years as a subarea of machine learning with considerable ability of adaptation for patterns and potential for solving a variety of complex problems. Goodfellow et al. (2016) thoroughly argued how deep learning is capable of being shaped as a valuable tool for abstract problems and can remedy many problems of the modern epoch.

The one distinctive feature of deep learning is its flexibility to the input, since a common approach of machine learning requires several human hours for data cleaning, filtering, normalization and tuning the data to acquire features for training.

Deep learning surpassed this barrier, where scientists have to prepare data for classification, regression problems and some sources of data were limited, such as images, music, signals. Lecun indicates the specific type of representation learning, where features can be directly passed to the system. Deep learning is a set of such techniques, where the system can be shaped specifically for the needs of the input and it is made in a compact and smart structure. Additionally, the layering of functions was a key for adjusting the deep neural network for the difficult problems. Finally, Guo et al. (2016) noted that the ability of deep learning for feature extraction with little human intervention set deep learning as the superior technique for computer vision applications, chemistry and medicine. The multiple review of deep learning applications [8],[9],[10] confirm its adaptability.

The next chapter will consist of a comprehensive yet concise description of the deep neural network for better understanding of the reader. The question of machine learning and artificial intelligence is almost philosophical and discussed multiple times. Thus, Nillson [11] claimed that a program should have an approximate simulation of intellect for solving complex tasks, understanding abstraction and logic. Therefore, systems became more intelligent and adapted for solving tasks similar to humans, as it required a more "humanic" approach and comprehension. This issue was discussed in past decades and Kaplan et al. gave the definition of artificial intelligence, where it described the general ability to process data as the human brain and make conclusions based on learning process and previous experience in reaching objectives. Therefore AI is not one direction, but a class of categories of replication of human knowledge acquiring, experience, decision making and argumentation, reasoning. Deep learning takes a special place in those problems as it is accustomed to learn complex data. However machine learning perceives the task of prediction and reasoning as pattern recognition and attempts to automatically detect patterns in the stack of information to predict upcoming data [12].

The three pillars of learning stand for supervised, unsupervised and reinforcement learning [13]. The supervised learning as it says in the name requires supervision for learning of the agent. There are x and y instances in the space and training process consisting of correctly marking the instances of x with labels of y in the classification process. Later, the efficiency of the network will be checked under supervision in a test set of samples to evaluate the ability to predict. If there are a finite number of classes (labels) in the problem, it is a branch of supervised learning classification. Contrary to this statement, if labels are continuous, the problem lies in the regression. The training set is open in supervised learning, the algorithm has an access to compare and learn from ground truth labels, which are precisely correct. It helps the network to compare predictions with true labels. The next stage is analysis of predictor performance. As a special exam for the network, it will give only X samples and without prior knowledge of the Y (ground truth) label it will make predictions. Furthermore, the analysis function will do the accuracy calculation and evaluate performance. In unsupervised learning the problem becomes more aimed at recognition of the pattern and Y samples are not known or not accessible. The primary goal of this learning is to cluster the samples in groups and identify connections between instances of a large pool of data [13]. The last, reinforcement learning agent is important in action making problems. The concept of reward-penalty formulates the behavior of the algorithm for proper output, the networks attempts to maximize the reward and avoid penalty with possible ways. It is an alternative way of training the model and usually it does not include samples for training but it uses generations of models to make attempts in pursuit of the correct pattern of actions.

Another concept, similar to deep learning, proposes application of artificial neural networks. ANNs is a subfield of machine learning and adapted from brain neuron structure. Silva et al. (2017) states that it adapts by experience accumulation, learning to make conclusions after reviewing large sets of data. The artificial neuron unit is simple in comparison to the biological neuron. The perceptron model is not a new study as it was already discussed by Rosenblatt (1958). The model of perceptron is simple and brilliant, the input vector multiplied by weights, which initially set to be random but will be adjusted further in the backpropagation process. There is a threshold in the end of each neuron which is responsible for making decision, should it fire and transmit information to other neurons or stay inactive [14]. One of the most popular activation functions is the rectified linear unit, which also inherits the simplicity of perceptron. It is a piecewise function which rejects negative input and translates input otherwise, without changes [6]. The positive quality of the function is that it is not complicated output and adaptable for optimization, in addition to good performance of fitting to non-linear patterns. The multi-layer perceptron architecture adds new layers to the structure and helps to achieve the nonlinear nature of the model, resolving previous problem with XOR output. As empirical evidence has shown, the increase of parameters of the network has more potential in achieving abstraction and solving more sophisticated problems. A mathematical method for changing the output to required state is tuning the parameters and it can be achieved through error function minimization. The error minimization is implemented through gradient descent function. The gradient function has a direction to the point of the large fluctuations of function values, it is possible to follow the reverse path and reach the point of minima. The example of reaching global minima through opposite direction is also used in backpropagation to detect the best set of the parameters to decrease the training error, the prominent work in this function made by [16]. Further discovering the ability of deep learning the cognitron model was discussed, the first study was conducted by Hubel [17] where he found ability of cells to understand patterns and authors proposed LGN model for recognition of visuals. Researchers adapted activation of the neuron from a similar mechanism of cats and monkeys cells of the cortex. They lit on only when specific objects appeared in their view. Authors proposed connection types between simple cells to the complex which are responsible for action and decision making. Thus the biological cells were connected in a hierarchical manner for passing signals and creatures could make a conclusion from received information. The current trend of learning of image input, convolutional neural networks, resembles the analogous structure. Nevertheless, the similarity stops in this feature and backpropagation is exclusively a contribution of the research group of LeCun [18]. Overall, the built system of CNNs has presented a great leap in progress of detection of objects in image and learning patterns in feature maps. Further development of CNNs were launched by Krizhevsky [19] network success in 2012 and deep learning was completely engaged in building convolutional neural networks.

1.2 – Convolutional Neural Networks

The convolutional neural network is a specific type of neural network which has the ability to compute the output from multidimensional data with spatial representation, like pictures and images. Definitely, the main inspiration for CNN was the cognitron model. The convolution in the name of networks stands for matrix multiplication in layers of the network [6]. The convolution in convolutional neural networks is multiplication between two matrices, one of them is kernel and the other is kernel. The output of this multiplication is a map of features, which is called accordingly, feature map. There is a condition in convolution operation that two functions have non-zero elements. It is an important detail as tensors of parameters should be learned in the next steps. The CNN has two major layers, the convolutional layers for processing features and a fully connected layer to generate the output. The input is an image with 3 channels and it will be processed by a convolutional filter with the same depth. The output of these convolutions is a feature map. The convolutional decreases width and height of the image and increases the depth of the processed feature map. Multiple convolutions increase the capacity of the network to catch details of the picture and detect implicit patterns. This process should be conducted several times and in an improved version of convolutional neural network there are additional layers for increasing accuracy of the network. The last layer of convolution will translate the feature map to the flatten layer to change the dimension of the input for processing by a fully connected layer. The fully connected layer resembles the structure of a multi-layer perceptron. The concise description of CNN would be a set of multiplication of input and convolutional layers for extraction of important features which further will be used in fully connected layers for needs of the classification or regression problem. CNN also can apply different operations on image matrix for simplification of the convolution process. Likewise, the pooling layer decreases the size of the feature map and subsequently reduces the consumption of computational resources. It also reduces the probability of wrong classification due to the position of the image and prevents overfitting or memorization of the input. The position invariance is necessary where the feature detection is crucial [6].

The pooling layer also should be associated with input's depth and the operation of pooling implements on regions of even square. Similarly, often kinds of pooling is max pooling where the maximum value of one square region is preserved. The rectified linear unit is used for the activation function [20]. It remedies two main problems of deep learning, it is an accumulation of large input with a great number of layers and training as a rectified linear unit only copies the input, and also rejects negative numbers, by that decreasing chance of creating neurons with zero signal. The different type of operations on feature maps is padding. The padding has a unique, pragmatic function of constraining the size of the filter and dimension of the output. The zero-padding aims to not increase the size of the feature map and match the dimension of it with the upcoming filter. The implicit function of zero-padding is the possibility to use more convolutional layers without decreasing the size of the feature map. The trick of stacking padding layers has its drawbacks as [6] argues about its problem with eradication of edge features. Besides discussed layers there is a layer which is crucial for thorough training process, dropout layer. It has many advantages, as it reduces overfitting, mimics, resembles cross-validation technique in machine learning by omitting some neurons from networks with certain probability. Therefore, computational load drops down, some of the neurons will not pass the check and could be removed as not useful for training [6]. The batch normalization is also a useful layer of the CNN for training. Batch normalization function implemented for improving stability of the CNN and it applies statistical knowledge for normalization of input for every layer with characteristics of zero mean. It is profitable for the training process, because batch normalization accelerates eventual convergence time. Furthermore it reduces gradients values and gives and enables higher learning rate of the network.

The main feature of the training of CNN is the gradient descent algorithm. It propagates through all networks and was developed from Stochastic Gradient Descent. It is a method for optimization for objective function, which attempts to replace real gradient with its calculated values. It provides faster calculation of optimization, lowers computational cost but has drawback of slower convergence rate. The gradient descent considers all points in calculating the loss, whereas stochastic gradient descent uses a single point. The classical SGD has modification of adaptive moment estimation and nesterov accelerated moment estimation [21]. The problem of SGD as it could fluctuate around region local optima and can be stuck in the one region, thus did not reach the optimal point. By adding momentum to the SGD, and adding part of the previous vector to the operation. The momentum term will decrease if the gradient changes direction and enlarge if the previous dimension had the same direction as the current gradient point. Thus its convergence rate will accelerate.

The nesterov momentum has the name momentum with anticipation, as gradient calculated with thought of possible weight values of the next step. It avoids unnecessary steps where weights will change direction and not minimize error value. The adaptive moment estimation is a concept which calculates the learning rate for each weight. It stores the average of the previous iterations and also momentum values and using those values adapts the learning rate speed in favor of training. Adaptive moment estimation gives first-order as well as second-order moments expected values for weights and accelerates convergence.

Chapter 2 - Literature Review

2.1 Development of visual servoing

The first attempt to describe and approach the problem of visual servoing was published in 2006 by [22], where the research group of Chaumette et al. proposed optimization problem and its solution.

$$e(t) = s(m(t), a) - s^* \qquad (1)$$

The vector m(t) is used for image input data, coordinates, and corners, the center of the object. This vector helped compute the robot arm's visual orientation and measure the distance between the laying object and the arm. The standard tool for the implementation of visual servoing is a 6-DOF robot with a camera. The methods for manipulating robot in space deviated due to the placement of the camera, but in this thesis project, I describe eye-in-hand manipulation.

The crucial part of the robot manipulation was the controller's velocity, and to correctly manipulate it, the interaction matrix equation (2) was introduced.

$$\dot{s} = L_s v_c \qquad (2)$$

There are two different approaches for visual servoing, image-based and positionbased. The first approach uses a plane and image to find a set of points s. The interaction matrix solves the problem of translating of 2D coordinates of the image to the 3D parameters of the camera space. The vector m provides coordinates of the image point used as pixels. The significant points constitute the vector of the interaction matrix, it is a bare minimum for constructing a vector of Lx. One popular method for producing an interaction matrix is to use the pseudoinverse of the interaction matrix since the latter is difficult to calculate in real-world visual servoing problems. The approximation sums up two error estimations of interaction matrices and finds the average. The geometrical explanation of this formula vividly expresses the meaning of image-base visual servoing since it depends on the rotation around the axis of the camera to the required position. The application of the error estimation aims to decrease the error between two positions.

Nevertheless, there is still a problem with a small error that the manipulator tends to ignore since the error assessment is negligible for the system but visible for the camera and human observer. Additionally, the large discrepancy between the initial and final positions causes empty rotation errors. The remedy for this problem is to use an approximation of the error estimation as the initial position, stepwise, to reduce the significant difference between two positions or increase it in case of a small distance. Position-based visual servoing applies to the camera's position and considers the coordinates of another frame. The computing of the pose is more complex and uses intrinsic 3D parameters of the camera. This problem is well-known as a 3D-localization problem. The position-based visual servoing accepts three frames with positions, required, current, and transit-frame for calculating the distance from it. The interaction matrix with an error estimation in this method will use the translation vector first and theta coefficient to calculate the embedded interaction matrix. The rotation effect provides orientation for further movement to the required position. PBVS approach uses fading movements and decreases the velocity close to the necessary pose. There are alternatives to split each action of the PBVS controller by parts and separate rotational movement from transitional. It has the advantage of precisely aiming to the point by the cost of more extended distance calculation. The stability review demonstrates stable movements of the IBVS controller if the number of corners of the image on which the oriented camera is not higher than 6 (number of degrees of freedom). However, in the real-world problem, these criteria mostly will not be met, and the controller will tend to reach local minima with a gradual decrease of the velocity and changing trajectories from the farthest to the closest. Furthermore, the problem was not solved entirely, and the error still exists, but in relatively lower values. The PBVS stability feedback scheme was built to assure precision, and computationally it should perfectly fit the chosen position, the controller will move accurately in simulation. In real estimation, some calculations will be biased and obtain calibration errors. The problem with the positionbased approach of accumulating the external factors and values can bring erroneous behavior and miscalculation of the error and distance. Finally, there is no explicit answer to which method is better, and it depends on the field of application and possible drawbacks of the approach in the environment. The stability issues follow both strategies, IBVS needs 3-D parameters and PBVS requires pose estimation for correct manipulation of the robot-arm. The PBVS method was based on the movements of a closed-loop system in Cartesian coordinates, but it is difficult to ensure that all sensors will be used as 3-D sensors for these purposes. The IBVS applies only 2D estimations and provides relatively high precision in manipulation with limited input data. Although a plethora of papers is dedicated to PBVS and IBVS, the novel approach of deep learning and namely convolutional neural networks, was not popular in the research community for this direction. A few articles attempted to apply CNN [23] to the visual servoing. The work of Tokuda et al. [24] demonstrated the implementation of the backbone of DEFInet to estimate the next post and current pose of the robot-arm. The first block was dedicated to the feature extraction for the construction of both feature maps, and in the second layer, the authors proposed an interesting concept of merging both feature maps to process features. Despite the alternative method proposed in [23], the authors decided to save a fully connected layer for regression and changed it to the outputting 6-DOF translation action values and angle vector. Authors achieved great accuracy values in experiments with positioning and difference between output and y-labels of the image. Their method achieved an accuracy discrepancy only of 0.010 ~ 0.015, which is less than 1 mm. Nevertheless, the research of [23] was the main inspiration for this project, as the authors applied and adapted the AlexNet network for visual servoing. The article of Bateux [23] described the hypothesis of transferring from the convenient DVS technique to the CNN approach. The authors' concept was to combine control law for positioning the robot arm with acquiring the required values for the interaction matrix from the CNN predictor.

2.2 – Convolutional Neural Networks in Visual Servoing.

Authors of comprehensive study of CNNs usage in visual servoing explored the requirements for application of neural networks for robotics manipulation. Scholars considered optimization problems and built a framework with an end-effector for robot arm manipulation. Likewise, authors reduced the need for extracting and filtering image features. As CNN-method implies, it is also not necessary to calculate camera pose and geometry coordinates of the scene. Interestingly, their prototype has shown good performance in orientation of synthetics samples and lab testing. As authors claim, their CNN model can manipulate robots with different conditions of light, angle of location without prior understanding of the scene and image features. The dataset for training was a 7-scene datascene with a variety of scenes and different positions of spectator. Evaluation process included 5 attempts of comparison of generated image and output with similar camera angle and real world scene. Researchers used simulation for testing real values of moving robot hands from computed and measured error. Proposed method achieved negligible value of the error and motion of the camera between two poses were close to the real movement of real-world test and position. Scholars also found a relation between two frames, that input information was enough to compute approximate 6 values of degrees of freedom. Alternatively, some works highlight using technique of ego-motion, which uses feature training of the model. The coupled siamese networks use 2 inputs as current and referenced image. Moreover, authors use regression in two image removing classification layer. The optimization process also differs, since authors implement loss and optimization functions closer to visual servoing task. There is another study which used optical flow between two images. However, the authors did not apply optical flow for computer vision problems and attempted to only generalize similar features of both images and also find connections between two conditions of generated image and output of tests in the laboratory.

Different study by Fischer et al. [25] used computer vision techniques and supervised learning for optical flow approach. This method used segmentation of the image to multiple pixels and calculation of distance as an input for model FlowNet, it was the one solution for problem with visual odometry of camera. Authors decided to develop a model which will calculate optical flow and can correctly translate the position of the camera and image, using pixel location. The FlowNet's goal was to correctly calculate optical flow through input of two images and values of flow as ground truth. Architecture of the model used stacked images of 6 channels which will be processed through convolution layers and rectified linear units for adaption of nonlinearity of data. Furthermore, CNN used decreasing the size of output feature maps for accelerating training process and decreasing computational power. The concept of prediction of the feature map of optical flow involves the requirement of high resolution of the picture as coordinates should be located in the same point. Authors also found solutions for correct output of per-pixel predictions. Blurry images were taken for upsampling to increase the scale of the picture. Commonly, in DNNs upconvolution is followed by unpooling, the process of upscaling with bilinearity, researchers used this layer after upsampling. This method was referenced by [26], where authors recovered corrupted feature maps and retrieved information from them. The layer of upconvolution was set in different scales, it will produce feature maps 25% smaller than original input. Finally, the authors discarded the loss stage of the original FlowNet. Instead of this, the authors used a fully-connected layer and added dropout function. It should be noted that scholars also separated regression layers.

Authors of [27] used deep learning for real-time visual servoing and manipulation of robotic arm. The first application of deep learning for grasp detection was used by [28].

Another research modified existing AlexNet architecture input with CGD and replaced blue channel in picture with depth information. Grasp rectangles were the output of the neural network. Moreover, authors developed different learning technique, which used calculation of rectangle per region for input.

Therefore, they introduced multigrasping learning, and AlexNet learned how to associate rectangle per object, and detecting multiple objects.

Siamese network used in work of [29] who used a couple of ResNet which will simultaneously calculate the output, one using information of RGB and other gets values of depth from CGD. The pipeline model was proposed by [30] as authors used detection and classification algorithms in two stages. The second stage classified objects and marked them with rectangles. The problem of that approach was slow reaction of grasp detection, and the input from cameras went through a protracted process of detection, classification and retrieving information to the robot. Further modification [31] of this network calculated only the angle of the rectangle.

2.3 – State of the art models for visual servoing.

One of the prominent works of [32] was the model which did not require preliminary knowledge of the environment and scene. It was achieved by training the model developed on foundation FlowNet and a large dataset [33]. Authors used a dataset of multiple pictures together with transformation of camera position. It used the relation between two images, desired image with position and current image. The network predicted transformation which will match the camera pose. It is important that the authors used UAV for calculating the position of the quadcopter. Both tests inside the facility and outside have shown great numbers of precision. Bateux et al.[23] used CNN model for predicting the transformation of a camera through the same method, using two images. The main difference in their approach was different architecture as Bateux et al. [23] applied AlexNet and VGG models and also model of robot and synthetically developed dataset. The input and output of the model operates on values of 6 Degrees of Freedom (DoF). Prominent detail there was usage of homography techniques for creating artificial dataset for the training. The work of [34] used four models for evaluation of their potential for visual servoing tasks, mainly for predicting capability. The networks used only images with required position and current image to make a regression of manipulation signal values. Therefore the author's work used a control system for predicting velocity, but not the pose of the robot. CNN was trained for grasp detection on a variety of objects (up to 200 objects). The controller prevented one-shot grasping and should address the issue with change of environment and erroneous signal. Authors were concerned about good generalization quality and trained controllers of visual servoing with a long process of optimization and validation of data. As scholars stated, it is possible to create a real-time tool for the grasping task as they use a light version of the network with faster reaction time.

The visual servoing function of the controller was used for manipulation of the robot, as it made a sequence of comparison of camera and referenced image, their positions and view of the object. Therefore, the visual servoing task should resolve the difficulties of the camera for tracking and grasping the objects. Consequently, the authors used CNN models, trained them, tested them with new, test dataset. The last stage gathered results of accuracy, speed of tested models. Last stage compared testing results of datasets with real-time grasping of the controller for four models. The system required proper values for the desired image of the object. The controller uses L1-norm for signal controlling if it surpasses border values. The system couples input (referenced) image and camera image (current position). The concept of siamese structure of two networks, which will predict the object in real-time helped the robot arm to grasp and find objects in real time. Additionally, visual servoing network predicted raw velocity signal and it was multiplied by coefficient λ which will be related to the camera. The joint velocities were calculated automatically by the controller of the robot and it required only position values from the detector of the object. After every iteration, the image is updated with the robot's position and if the signal is sufficient for translation of the movement, the next iteration will start. Accordingly, if the condition inside the loop is met, the predictions were close to the threshold of coordinates and the prediction task was successfully performed. Later, the robot will calculate the kinematics to move to the required position and grip the object. Grasping area applies only for x and y coordinates of region, as robot uses only single grasping of the object. Authors claim that multiple view of the grasping were not the objective of their research and can be discussed as future work.

Deep learning and visual servoing with data augmentation were discussed in the research work by [4]. Their model uses image input and pose values of the robot arm as an output. The training dataset consists of images taken with a built-in camera; ground truth labels use data of 6 degrees of freedom values for each image. Authors' method used a virtual camera, which can capture images and also provide different positions. For this task they applied OpenCV vision library to develop perspective vision in a virtual camera. They used two networks for positioning and pointing tasks. Network uses a small input image and produces two feature maps of two small networks and calculates the 6 DOF values of the camera itself for the input image. The first network resembles the architecture of AlexNet but uses the principle of regression output for the image. The importance of using a regressor in the last layer with output of 6 degrees of freedom values allows the robot arm freely in the space. To control this system authors added the PointNet. The PointNet network learns point features in input samples, the PointNet learns relationships, non-linearities of picture and space. It takes the feature points in the image and learns connections of the points to the pose in space. PointNet has a pipeline of two stages, where the first part of the model processes feature points in the picture. In the next stage values are translated to the grid with coordinates in the picture. The second stage includes deconvolution from feature vector to the feature map with the same scale.

Combination of the feature maps of two networks gives the 6 floats of freedom after a fully connected layer. Activation function of two networks uses a rectified linear unit and a leaky rectified linear unit for PointerNet for connecting more features to the space. The system of embedded camera in the robot hand, some of the points of the object can be not visible for the system. Therefore in calculation part points will be indicated as negative, as they leave the field of view. This part of the robot arm manipulation also could be trained in the network. The remedy for this problem is mixing the knowledge of the location of feature points in the environment which was introduced before to the model. The key of the training of two networks include separate training of PrNet and PtNet and after combination of them for the optimization process. Subsequently, the training requires addition of the labels for the both networks, first would be the state of the controller and values of position with feature points in the image. The controller network will be a combined CNN of the image input and referenced pose and desired pose. Authors also indicated that initial pose prediction will include random values of the output as a starting point, and in difficult cases of prediction, where error will be small, control law output fluctuations will be greater than usual. This case implies the desired position is closer to the robot arm and in this case the condition of eliminating calculating of the pose will be activated. It is created for prevention of unnecessary fluctuations. Researchers argue that in test scenarios no uncommon oscillations were not noticed.

Liu et al. (2020) [30] have achieved full control by decreasing both angles θT, θR close to 0. Movements of other joints are also coordinated by critical thresholds. This system of limits and constraints developed for avoiding fluctuations in stability of the arm and incorrect positioning. The float values for each joint were computed independently and complete control of each joint were accomplished in sequential manner. After the model was trained on a large dataset of tuples of image-positions, it applied for visual servoing in a real robot-arm. For this task researchers created a simulation of a robot with a built-in camera and two images were passed to the input of the visual servoing system. Both images had discrepancy in positions and the model learned how to manipulate the regressor coefficients to achieve required pose. Authors used different datasets during experiments and compared results, impact of input change on two networks.

The experiments with dataset #3 used only 2D objects which can lay on the surface, and with this input manipulator had small fluctuations, but in most cases has shown great stability and accuracy. Similarly, authors decided to move the border value of the angle to 0. In dataset #4 unexpected instability has negatively impacted on precision, the key point of zero angle worsened predictors ability to compute values. Thus, Liu et al. increased stop value for 2mm and 1.5 degrees for this input.

As the authors stated, the second network experienced difficulties in learning patterns of translation from 2D picture to the space only with input data of images.

Another interesting work by Raj et al. (2020) [34] discussed two paths of visual servoing and combining it with modern CNN techniques. As authors claim, currently there are two methods of visual servoing, one of which a manipulator learns by image and reference coordinates. The other method, where a robot is learning a path to get the nearest point possible to the object, usually it is an image of the object. The basic concept of visual servoing includes long training of the model, with optimal number of samples of 10,000 instances. However it is a difficult task to gather 10,000 pictures with coordinates. Therefore, Raj et al. (2020) [34] discussed methods of not fixing desired image position and using pretrained models. Authors' first attempt of training did not succeed as the model could not achieve sufficient convergence rate. Therefore, they used a direct visual servoing method. As they stated, although deep learning, specifically CNN, achieved considerable success in performing visual servoing task, it still requires support of DVS method and is not distinct without it. Authors attempted to find a border value where the robot arm can get to the object and reproduce positions which were provided with image. Additionally there were others attempts to evaluate performance of deep learning algorithms for learning manipulator actions. Thus, RNNs were tested. In similar work [35] they combined two methods of orientation within a closed environment. The algorithm changed directions depending on conditions. Robot calculated the range from the received pictures of the camera. Therefore, to move close to the objects network trained on the zoomed images and for objects which were on the distance it used focus computations. However in their work, focused objects training included only images with small scale, identical to closer objects.

Furthermore, the authors decided to improve the existing model of swap algorithm for changing CNN and added a meta-learning model. Their method outperformed common techniques of changing the model. The idea of swapping the model came from the similar concept of robot with fruit grabbing. It was trained to ignore the fruits if the range of the detection was too high, it worsened accuracy, but improved focus on closest fruits near the robot. Consequently, if the robot got to the target, the target changed and accuracy was the priority.

In the authors model the encoder has great importance, as they combined the popular ResNet model together with ImageNet but removed the classification goal of the networks. Instead of classification they used a pose regressor. The top layers did not change as they filter features. Two images were passed to the CNN layers, independently. Consequently, features after convolutional layers were connected and passed to the other convolutional layers to filter only important details for manipulation of the arm. As CNN network structure implies, these features will pass batch normalization with a rectified linear unit to further adapt the model for non-linearity. The pooling layer will maximize them and remove unnecessary values for computation. Total size of the feature map will be 1024x7 and 7 channels. The average pooling will change it to 7x1024. Additionally, authors indicated that the data generation process will be challenging for doing it in the real camera of the robot as [36] research has shown that it could take 700 robot hours. Therefore authors used simulation with artificial objective and other additional objects for reproducing scenes from real camera. Scholars applied a free camera model up to the robot and it set to the right corner from the manipulator. The camera was moved with different translation and rotation behavior within limits of physical analogue. Each scene used a different location of objects. Researchers highlighted that images and pose of the robot were saved and automatically labeled with position values. The experimental part used only a small fraction of the training dataset, approximately 100 instances with values.

Yu et al.[29] has proposed siamese architecture of the network, which is used for calculation two translations between two poses of the camera. Each network independently processed features of the images. Alternative method used connection of two pictures within channel dimension and perform extraction of lines, dots from image as a whole. Authors used two neural networks for convolutional operations instead of stitching images and process them in one network as used in [37]. As Yu et al.[29] explained, it will be important for the next stage of the pipeline to locate the pose for the manipulator. They used CaffeNet for the feature extraction layer in both networks. The number of features were limited by reducing the dimension of the output feature map to the 96 with a small kernel. Firstly, the authors used a sum of two feature maps and it did not bring satisfying output, therefore they decided to flatten the output and feed it to the classification layer. Furthermore, additional fully connected layers were set in the end of the network for processing features of concatenated input. The output had two directions, to calculate the manipulations of the robot arm. As [29] states, it did not work and coupling of two feature maps worsened proper computation for translation and rotation as it separated them. Authors decided to measure relative pose transformation between two points of the camera by training a model on samples of image and label of TD2E. As scholars mentioned, the translation parameter with other parameters were balanced with an additional weight of 0.99. For the optimization authors used root mean square error with parameters of 3 and 4. Authors used a quaternion system for better encoding of axis angles.

Chapter 3 – Methodology

3.1 – Background and context research.

The research of using CNN in visual servoing had a narrow and limited number of sources, therefore for the deeper understanding of the field, I highlight multiple sources [38][39][40] where deep learning applied for visual servoing tasks. Likewise, authors of the first article [38] described an alternative approach to the CNN — autoencoders. As they state, they trained autoencoders for minimizing the reconstruction error. ResNet was proposed as both encoder and decoder, and authors emphasize that there are more advanced architectures, which I tested in my thesis project. The process of decoding involved translation of standard networks with exchange downsampling layers to upsampling. Similarly to the [38] I did not apply old weights and new weights were initialized. In the authors' case, they split the model to two networks with separate weights. The authors' technique of weights normalization instead of using common batch normalization inspired me to experiment with AlexNet architecture to find the best trade-off between complexity of the model and its performance, thus I added several DropOut layers. Additionally, I selected models with less number of batch normalization layers and attempted to avoid it as they state it interferes with the LI values and precision significantly drops. One of the interesting parts of their study, exchanging average pooling to the grouped convolution. Similar concepts exist in EfficientNet with convolutional layers and width parameters.

The second work [39] increased complexity of visual servoing and although the topic discussed did not closely relate to the thesis project, as I focus on visual servoing with 2D objects, images from the cameras, authors use CNN pose-estimation architecture for understanding the depth of 3D objects. Authors use ConvNet and deep pose estimation models for synthesis of 2D belief maps, and it is applicable for the predicted values of the regression network, with 6 degrees of freedom. This publication helped me understand how a model will perceive the set of acquired features after processing it through the consequence of filters. Their receptive field is larger than my developed model as it attempts to create a “belief map” together with a feature map of the image taken from the robot-hand. Additionally, authors highlighted the importance of VGG-19 model for features extractor and this transfer learning also suggested to me the principle to use smaller models and combine/alter their hyperparameter, modify for visual servoing problem.

Third study [40] discussed the ability of the robot-manipulator to recognize and properly move clothes and in the chapter of contrastive prediction, authors make disclaimer about a fully observable environment, as I had in simulation. It will be the most desirable, clean conditions for visual servoing of my developed model as CNN trained on the dataset samples made in such conditions. Authors suggest application of direct visual servoing mode where they used learning pixel-by-pixel and use regression for tuples of normalized values of the image. The model was learned in this manner and their first step was similar to my attempt to learn a model for moving towards a 2D object (image). They also emphasized the difficulty of learning in DVS approach as the pixel values cannot always symbolize distance to the object. For explanation they give an example of the ball manipulation task, as moving the ball will invoke problems with pixel mixing and image and feature maps will be distorted for end-effector calculations [40]. Additionally authors used specific loss functions and gave me motivation to reconsider choice of the loss function and optimization algorithm for model specifically for visual servoing, where authors used InfoNCE contrastive loss [41].

Crucial study for background research of the deep neural networks in visual servoing was the ablation study by [24] as they compared performance of the model with different configurations, such as size, depth of the model and architecture. After 5 runs in ResNet encoder for multiple models they reported about the importance of the network size, and convergence rate changes similarly. Furthermore, the size of the dataset impacted on convergence rate, especially with configuration of CNN+DVS.

3.2 – Visual servoing optimization problem.

The concept of application of deep learning in visual servoing unites two ideas, the first abovementioned problem of optimization in robotics and deep learning approach of finding patterns, connections between input images and provided correct positions. Therefore, the first attempts supposed to create hybrid CNN model for processing referenced frames of desired positions and final layer will be regressor with multiple output of 6 positions, according to 6 degrees of freedom. Basic model of CNN consists of 5 pillars: convolutional layer with filters to extract the most important features of the object, the pooling layers to decrease the size of created feature maps, it helps in decreasing computations and generalizes output, “squeezing” the maximum of them. The activation function, one of the crucial parts of network, uses mathematical function to transmit significant values to the final fully connected layers, which will gather and connect all information from layers. Furthermore, after protracted training of three stages and measuring results, 9 models were tested. The three stages training and measuring helped to show which optimization function will have faster convergence rate.

As discussed in the literature review, visual servoing presents an optimization problem, and the robot manipulator aims to minimize positioning errors between two frames. Considering the equation from [23]:

$$\hat{r} = \arg\min \rho\,(\mathbf{r}, \mathbf{r}^{*}) \qquad (3)$$

This problem can be formulated as the cost function of the Euclidian norm of two different vectors:

$$\hat{r} = \arg\min \, || s(r) - s^{*} || \qquad (4)$$

The most distinct feature here is that the vector can be used as a set of 2D or 3D features. There are many shortcomings with the extraction of these features by both approaches, and in [23], there was proposed direct visual servoing (DVS), which was further improved with deep learning and CNNs. A similar method processes the image entirely as a set of features. The problem with this method is that it does not suit the concept of optimization by interaction matrix due to the high nonlinearity of the image data. In my term project, I revisit and modify a novel approach to applying the CNN deep learning field to the DVS of robotic manipulation.

The network will estimate the pose between two images and output 6-DOF coordinates for the robotic arm. As in direct visual servoing, the network will compare referenced frame with the required frame. The CNN firstly will calculate the equation for the distance between two poses by homogeneous matrices and obtain the value. The cost function of this network represents Euclidean distance and tends to minimize the difference between two frames. This method is similar to the PBVS for converging initial and end poses. From the pose difference calculated by CNN it is possible to compute camera velocity by control law. Finally, the velocity value defines how to manipulate the robot arm.

3.3 – CNN models for visual servoing.

Developing CNN model requires a specific dataset with large enough instances for training, validation, and testing. The rational solution here will be the application of the pretrained model. Applying the pre-trained model will decrease the dataset's size and significantly reduce training time. Pre-training or transfer learning process is a well-known image classification technique, but it will also be useful for regression problems. Although the network input is images, the last layer replacement is required. As the first candidate for this position was chosen a simple, classical model VGG-16, the neural network which previously trained on 1.2 million images with 1000 classes. The ability to fast classify images and acquire features, edges and lines will be crucial in the regression task of visual servoing. The choice of loss function was also important as it served to redirect the function training to the region of at least, local minima. Certainly, the global minima were the desired goal for the training process. I developed custom of Euclidian distance function (3) for loading the array of 6 floats which were taken from dataframe as ground truth labels and compared to the generated floats of the last layer of linear activation. It summed the squared difference of true values and predicted floats (5), and the sum was taken under the square root.

$$\sqrt{\ }[\ (y_{1\ true} - y_{1\ pred})^2 + (y_{2\ true} - y_{2\ pred})^2 + (y_{n\ true} - y_{n\ pred})^2] \qquad (5)$$

Furthermore, I modified architecture of the CNNs. Commonly, CNN applied to the classification task, object detection. In case of visual servoing, the classification task was replaced by multiple output regression problem. To do this, in the first stage I acquired the most famous architectures of CNNs and changed final, fully connected layer with softmax activation and output of 1000 classes (AlexNet case) to the linear regression of 6 degrees of freedom. As referenced in Bateux et al. [23] the loss function was also modified in favor of mean squared error.

Consequently, the problem of visual servoing is optimization of positioning of the robot, not predicting the correct class of the object. Therefore, mean squared error used for every predicted float of joint. The mean squared error is well-known function for machine learning regression problems and it widely used by robotics scholars in applying CNN [24][25]. Thus, I applied MSE error as a loss for training and testing models. In comparison with MSE error I tested adaptive moment estimation optimization function (Adam) and extension of popular stochastic gradient descent, root mean square propagation. This optimization function is underrepresented and has narrow usage in neural networks, as it is handful in situations where gradient can be too small or too large. In simple demonstration on figure 1, it is shown the speed of finding optimal parameters for used function. The red line expresses RMSProp function and as it visible, it takes less steps and fluctuations for reaching required point.

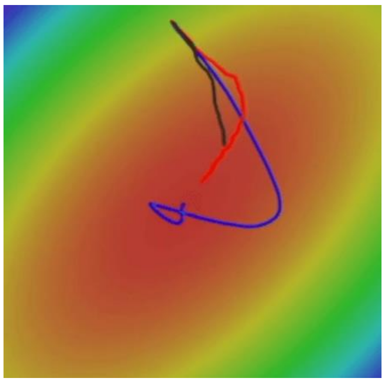

Figure 1. The search of minima by SGD, Adam and RMSprop optimizers

The RMSProp gradient function multiplies the sum of squared gradients and use coefficient of decay rate with current gradient. The RMSProp divides used gradient by sum of squared gradients to speed up the search of minima and update weights across one dimension.

For minimizing necessary research time for training, I used pre-trained models, the research of [29] has shown that it is possible to fine-tune the network without training the model from the empty, random weights. I decided to remove the last layer with softmax to the layer with only 6-DOF output floats. The network was trained by inputting tuples of image - coordinates. The cost function of this problem represents the Euclidian cost function. The network was initially adapted for visual servoing and later trained on a dataset of 8000 images with coordinates.

The training and testing process included several models (VGG-16, AlexNet, AgeNet, AlexNet modified with Dropout, Xception, EfficientNet, DenseNet, ResNetv1/v2 and different structure of 50 and 152 layers). Accordingly, all models have shown different results. The continuous training of the models was useful for achieving satisfactory conversion rate.

Table 1. The comparison of used architectures

Architecture	**Structure**	**Description**
VGG-16	13 convolutional layers 3 Fully Connected	CNN with 16 layers, size of 3x3 convolutional layers with stride 1 and padding 1, together with max pooling layers. It has a compact architecture.
VGG-19	13 Conv.layers 3 Fully connected layers	CNN with 19 layers, similar to VGG-16 but with additional convolutional layers. Extended version of VGG-19 with more parameters and higher computational requirements.
AlexNet	5 convolutional layers 3 fully connected layers	Powerful CNN with 8 layers, consisting of 5 convolutional layers, max pooling layers, and 3 fully connected layers.
AgeNet	8 convolutional layers 3 fully connected layers	CNN for age estimation from facial images with 8 convolutional layers.
AlexNet with Dropout	5 convolutional layers 2 fully connected layers Dropout (0.5)	Modified version of AlexNet that includes Dropout.
Xception	36 separable Conv.layers	CNN with 36 depthwise separable convolutional layers. Possible to scale and diminish the structure without loss of accuracy.
EfficientNet	MBConv blocks, 19 convolutional layers	CNN with the objective of minimizing computation power, decreasing FLOPS and compound scaling technique which scales the depth, width, and resolution.

DenseNet	Dense blocks with transition layers	CNN with dense blocks which enhance translation of information to other blocks. Each layer is connected to every other layer, transition layers that downsample the feature maps between dense blocks.
ResNetv1/v2	Residual blocks	CNN with multiple residual blocks, every block consists of several convolutional layers, residual connection that skip the block. Tests included 50 and 152 layers architecture

Later, the test in simulation of roboticstoolbox package from matlab which was transferred to the python environment. The goal of the simulation test was to evaluate the potential performance of visual servoing predictor in lab conditions. It is important to check will the robot arm get to the right point. The actions of the robot arm were closer to the predicted point with negligible error of 1~2 cm.

3.4 - Dataset

One of the significant parts of the thesis project was the generation of the dataset. Firstly, it required different positions and angles of images and subsequent coordinates to collect enough heterogeneous instances. Secondly, the synthetic dataset was needed because it is impossible to create many images within the laboratory. In this section, I describe how synthetic data was generated in simulation. The first dataset was developed only from one image, and here I applied computer vision techniques to simulate different camera viewpoints on the same image. The referenced image was used for transfer technique and deep homography. Thanks to the work of authors [23], which introduced a valuable approach for creating images with coordinates by applying homography. The stages of the generation of the synthetic dataset included multiple steps. First, provide the first referenced image with the initial pose. Secondly, generate 8,000 images for training and testing purposes by the gaussian method and perturbations around the camera by calculating the 6 DOF floats.

The multiple configurations were trained on the dataset of 8000 images which were generated through gaussian draw within simulation and camera, and from camera’s point of view I set the script to automatically make a screenshot and save the position of the camera as set of the joints distances and angles (6 degrees of freedom). Author prolonged synthesis of data with randomized coordinates, the dataset was ready for pre-processing.

I split the dataset in 3 parts with 80% of the dataset (5600 instances) for training, 1600 for validation and a small partition I saved for test evaluation. This is a common technique for deep neural network training as it helps optimize parameters before the final test of CNN performance.

The weights for training were reinitialized and previous weights discarded as they are not useful for the visual servoing task. The images were resized according to the input of the model where their initial size was decreased to the 224x224 and RGB channel, for memory saving and faster training. The total time for training of one configuration took almost 22 hours for 1000 epochs in the cloud-based environment of Google colab with acceleration of matrix multiplications by GPU, which was Tesla T4. The distance of the camera in the simulation scene was 20 cm of the viewpoint and points were distributed by gaussian draw according to the 6 joints positions, 1 cm shift for three distances and 1 degree for angles per iteration. Additionally, the dataset was fed to the datagenerator module for justified distribution of the instances and shuffled before the training with default random seed of 42.

Chapter 4 - Results

The experimental stage used the pipeline of gradual testing of multiple CNN architectures, all of them were highlighted as the best for image classification and majority of CNN models achieved the highest precision in ImageNet competition. Additionally, I experimented with loss, accuracy metrics and optimization functions for tested models. The initial loss was decreased with the increase of number of epochs, I set the model for preserving the best weights and stopping training to avoid overfitting. The VGG-16 model has achieved the loss of 10 with percentage of 91.1 prediction correct, although the yaw (rotation around the vertical axis of the robot-hand manipulator) predicted erroneous value, adding too many degrees in comparison with ground truth number. The first experiment with adaptive moment estimation optimizer achieved visible results in comparison with RMSProp.

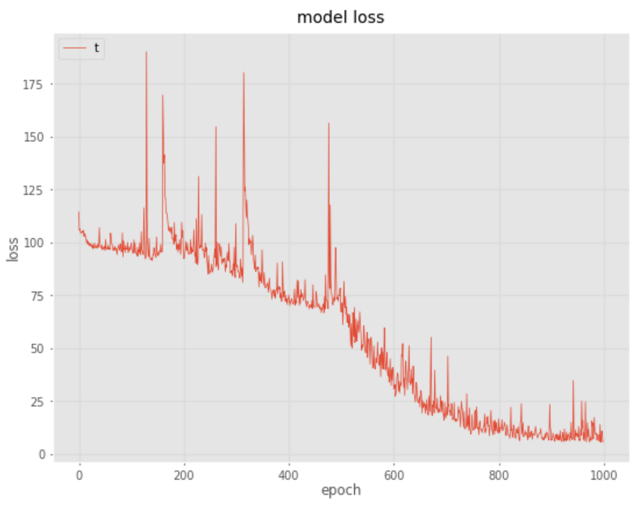

Figure 2. The VGG train loss

The first attempt of training models together with metrics has shown visible results, where loss of the function did not converge in the first 100 epochs and slightly declined towards value of 150, however all metrics greatly declined towards value of 20. The AlexNet training loss and validation loss started to decrease after the 50th epoch and decreased significantly after 400 epoch. The ResNet-152v2 training process indicates great fluctuation due to heterogeneous input and the model complexity worsens its performance as it does not converge properly both in error metrics as well as in training with validation loss.

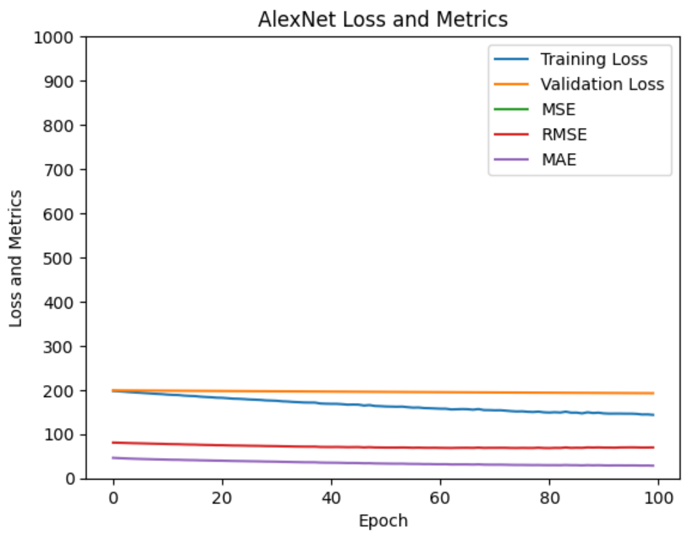

Figure 3. AlexNet Loss and Accuracy metrics after 100 epochs

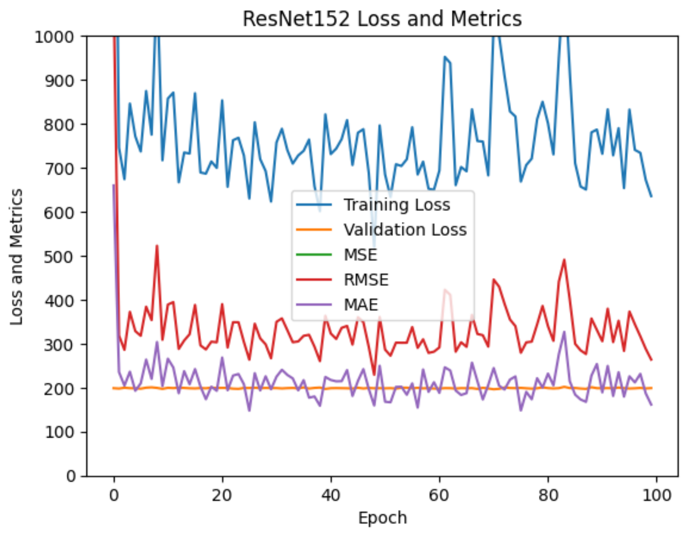

Figure 4. ResNetv2-152 Loss and Accuracy metrics after 100 epochs

XceptionNet resulted in good convergence of the training loss but in the validation dataset it did not improve. The depthwise convolution operations were efficient in classification of images but regression tasks with unpredictable input did not give considerable result of loss.

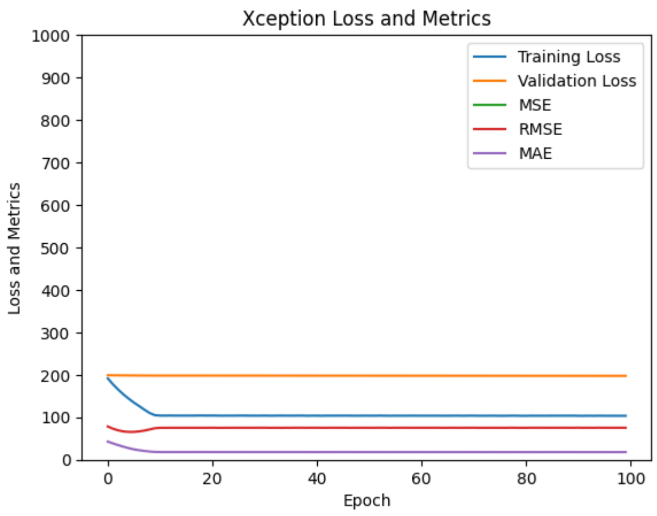

Figure 5. Xception Loss and Accuracy metrics after 100 epochs

Although all models have indicated high accuracy and both MAE and MSE metrics prove that in the training stage, I considered root mean squared error as the main factor of model performance, for the models with low performance it did not decrease.

Definitely, all models did not converge as 100 epochs was not enough for a full training process, but it was important to show intermediate, preliminary results after some time of the model training. Consequently, the training of ResNet model stopped at approximate loss position of 45, with further fluctuations in test dataset, as its accuracy were not stable and only 3 positions of 6 were predicted correctly, and majority of labels in dataset did not align with predicted values.

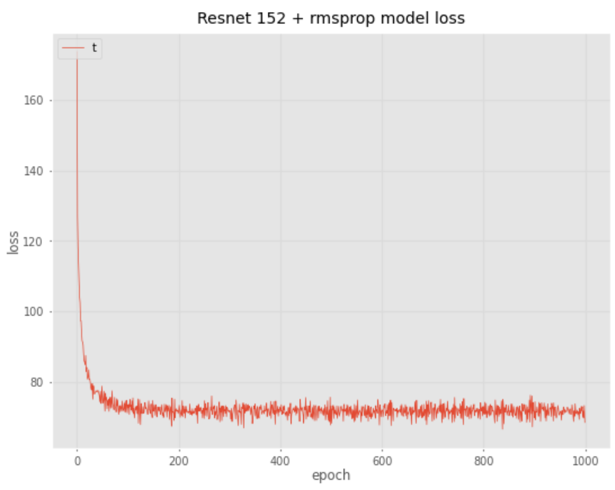

Figure 6. The ResNet152 train loss with RMSprop

The interesting result were achieved by XceptionNet, as it used two kinds of convolutions, depthwise and pointwise. The decrease of the time also correlated with loss in the first epochs, but later result did not change, and model did not converge in validation dataset.

The RMSprop accelerated the learning process of the model and loss stopped at position of 55. This network was close to the training simplier models, as VGG-16/19.

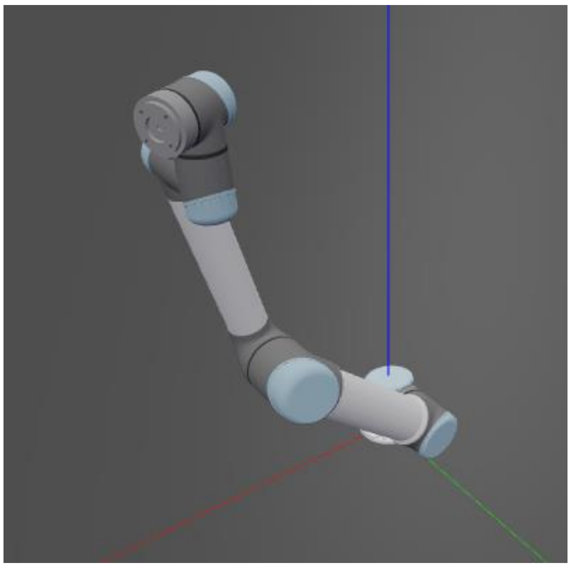

Figure 7. Robot arm model positioning in Swift simulation

Multiple models were discarded as they indicated less efficient results both in training and validation slices of dataset. Likewise, the EfficientNet versions with Adam optimizer did not converge and significant loss of 197 highlighted inaccurate predictions of model, consequently, the accuracy of the model was unacceptable for further experiments.

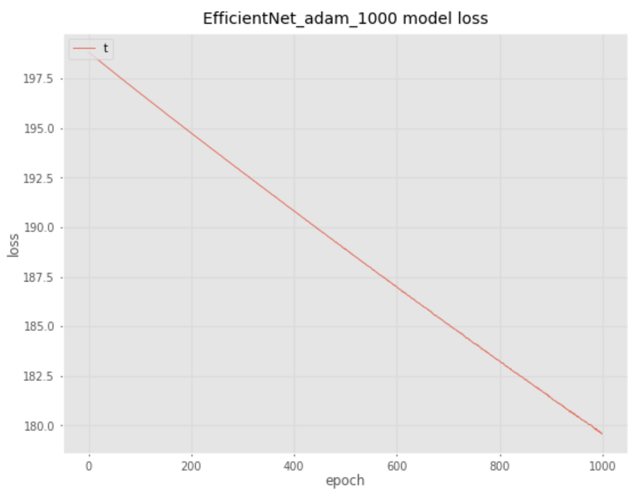

Figure 8. The EfficientNet model training process, problems with convergence rate

Similarly, the improved version of ResNet, ResNetv2 with a smaller number of layers, 50, despite prototype of residual blocks with pre-activation converged only to 250 values of loss.

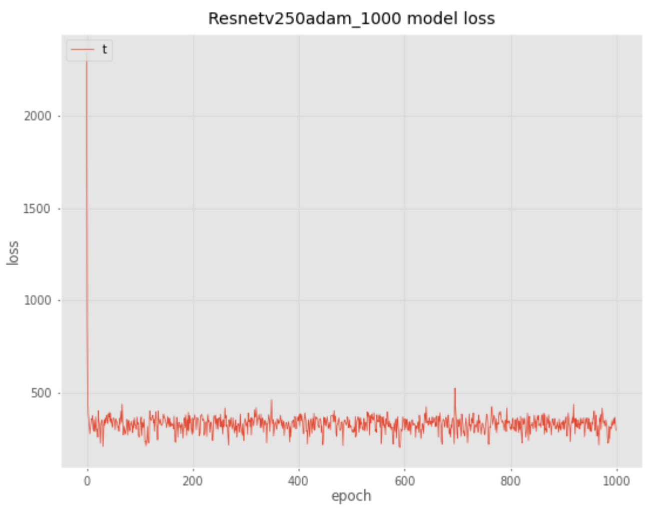

Figure 9. The ResNetv2 50-layer model training process, problems with convergence rate

The second stage with preliminary interference of 100 epochs together with test dataset and undiscovered slice of dataset has shown real performance of the models. The first attempt of AgeNet revelated that the test loss did not decrease and even grew, where training dataset indicated loss of 60.

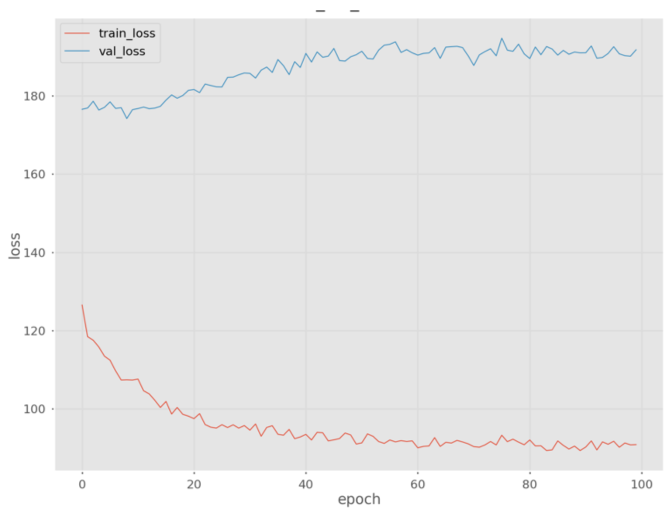

Figure 10. The AgeNet model training and testing process

The second attempt of applying DenseNet indicated potential improvement of the test loss together with training loss. The purple line indicates significant drop of the training loss, validation loss has shown moderate decrease from 200 to 180.

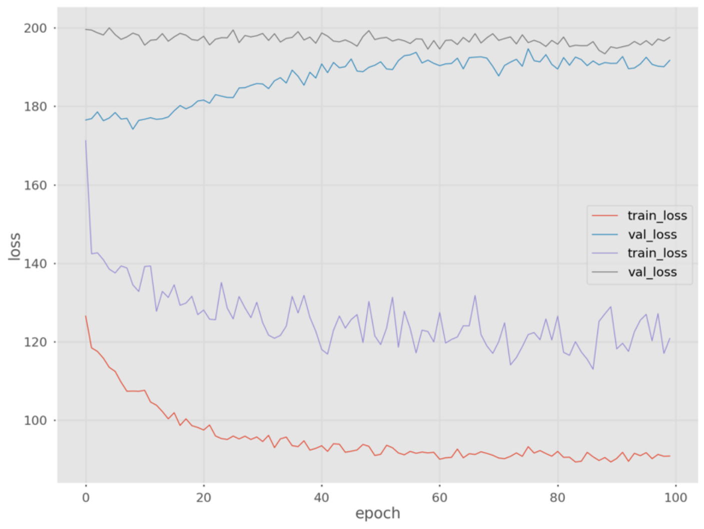

Figure 11. The DenseNet model training and testing process

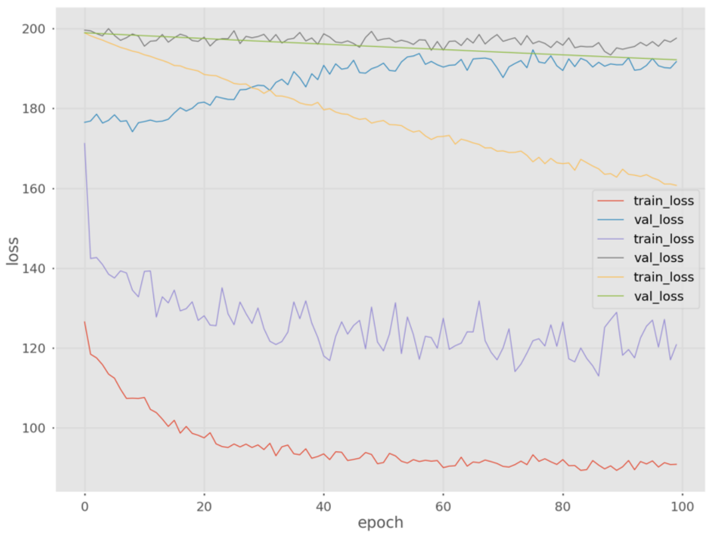

Figure 12. The AlexNet model training and testing process

Third attempt applied AlexNet with Dropout layers, regularization, and dropout rate 0.5, and it further improved validation loss, and steep decrease of it. Further improvement of the AlexNet through increase of epochs for training achieved considerate value of test loss.

The results of the AlexNet evaluation accounted of 1.5 cm deviation from provided ground truth. The positioning error achieved an accuracy increase with a proportional decrease in SSD error, and final floats accounted for a deviation of 1cm, which was worse than what was published in the original paper. Adding gaussian noise to the pictures and blurring the lines also worsened the results of accuracy.

The result which achieved by my proposed model were 3e+08 SSD distance with deviation after 700 iterations if training. The positioning error constituted 1.5~1.75 cm for XYZ vectors, translational errors were 1~0.95 cm in comparison with ground truth. The final precision achieved value of 1.2 cm – 1.5 cm of discrepancy. Further analysis of the results has shown erroneous predictions in yaw positioning angle. In each model predictions deviated. This problem was fixed with further adjusting and more interestingly, the AlexNet architecture with modified layer improved predictions and minimized error to 2 degrees as I added Dropout layers to the network with value of 0.5. The tilt of the manipulator also disappeared. The Figure 13 illustrates improvements of moving the head of the robot towards possible image coordinates.

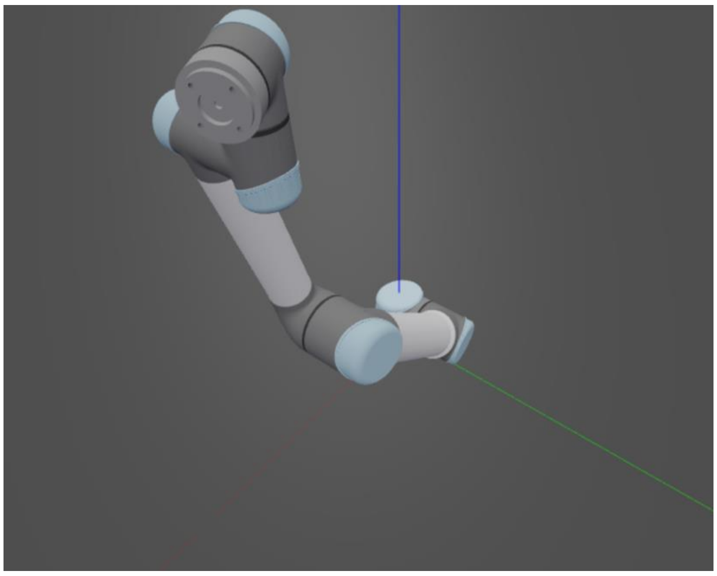

Figure 13. 3D model of the UR-5 Robot in Swift simulation of roboticstoolbox package

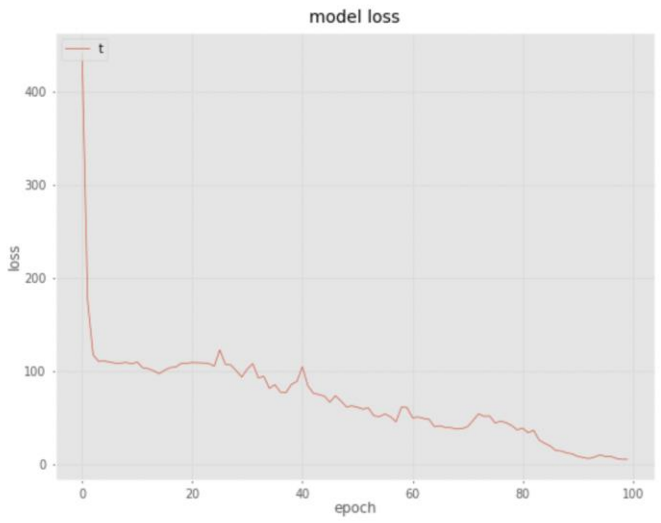

Figure 14. The AlexNet test loss of the modified and pretrained model with serial run of Adam and RMSprop optimizer

CHAPTER 5 – FUTURE WORK

The future work of this thesis project includes positioning of the robot under distortion of the lighting and training the model after data augmentation of images with rotation, translation image not only for the enhancement of the training process, but also for preparing the model for unexpected conditions and reducing preparation time of the model. The bright and darkened lighting can be useful in input images, the shift of the dataset pictures positions can make the model more flexible, instead of rigid values of the joints, which although were generated for training, but did not completely. imitate real experience of hand manipulation. Another interesting topic of the research is positioning of the robot on similar objects and pictures with different depth and shape, it is possible to train the model for slight change of the movements after processing unusual input which differs from the majority of the dataset. The combination of the positioning of the different lighting and on new objects can also open new directions for CNN training for visual servoing. The new word in the world of image classification is the attention models and their ability to focus only on “hot" regions of the image which can be encoded. Its distinctive feature can be also applied in visual servoing, especially on focus on close or far positioned objects.

CHAPTER 6 – CONCLUSION

In this thesis project I have shown potential of CNN models and refined them for visual servoing task. Transfer learning of weights of the network was efficient in terms of reducing the time of training. The initial goal of this project was to prove the possibility of repurposing CNN and Deep Learning for the problems of robotics and evaluate the performance of the network compared with referenced parts of the research. Subsequent objectives were to design ultimate solution from existing CNN models and modify them, repurpose for visual servoing task. Thirdly, it was required to generate synthetic dataset and prove feasibility of the training of the model. According to the reported results, all three key objectives were achieved through several attempts described in methodology. The method of application of CNN has viability and requires future investigation. The AlexNet modified network can regress to the required float value, and the difference between prediction and ground truth accounts for 1-1,5 cm and less than 2 degrees in terms of angle joints. The network training and adding data augmentation to the model had positively impacted accuracy. Although this network adapts the PBVS behavior of manipulating robot-arm in space, it is also possible to use it for other tasks and improve the network for assessing the position of 3D objects. Another significant contribution of this project was simulation learning since methods of homography synthesized part of the data used for the training. The final accuracy result confirms that training without real-world data in visual servoing is efficient and could be helpful for research in a different direction of robotics.

Bibliography/References

[1] Silveira, G. and Malis, E., 2012, “Direct Visual Servoing: Vision-Based Estimation and Control Using Only Nonmetric Information,” IEEE Trans. Robotics, **28**(4), pp. 974-980.

[2] Ourak, M., Brahim, T., Lehmann, O., and Andreff, N., 2019, “Direct Visual Servoing Using Wavelet Coefficients,” IEEE/ASME Trans. Mechatronics, pp. 1–11.

[3] Caron, G. and Yoshiyasu, Y., 2022, "Direct visual servoing in the non-linear scale space of camera pose,” *26th Int. Conf. on Pattern Recognition (ICPR),* Montreal, QC, Canada, pp. 4154-4160.

[4] Chaumette, F. and Hutchinson, S., 2006, “Visual servo control. I. Basic approaches,” IEEE Robotics & Automation Magazine, **13**(4), pp. 82-90.

[5] Chaumette, F. and Hutchinson, S., 2007, “Visual servo control. II. Advanced approaches [Tutorial],” IEEE Robotics & Automation Magazine, **14(**1), pp. 109-118.

[6] Goodfellow, I., Bengio, Y., and Courville, A., 2016, *Deep Learning*, MIT Press, Cambridge, MA.

[7] Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., and Lew, M.S., 2016, “Deep learning for visual understanding: A review,” Neurocomputing, **187**(C), pp. 27–48.

[8] Goh, G.B., Hodas, N.O., and Vishnu, A., 2017, “Deep Learning for Computational Chemistry,” arXiv.

[9] Chai, J., Zeng, H., Li, A., and Ngai, E., 2021, “Deep learning in computer vision: A critical review of emerging techniques and application scenarios,” Mach. Learn. Appl., **6**, p. 100134.

[10] Yang, S., Zhu, F., Ling, X., Liu, Q., and Zhao, P., 2021, “Intelligent Health Care: Applications of Deep Learning in Computational Medicine,” Front. Genet., **12**, p. 607471.

[11] Nilsson, N.J., 1980, *Principles of Artificial Intelligence,* Morgan Kaufmann Publishers Inc., San Francisco, CA.

[12] Hastie, T., Tibshirani, R., and Friedman, J., 2001, *The Elements of Statistical Learning*, Springer New York Inc., New York, NY.

[13] Bishop, C. M., 2006, *Pattern Recognition and Machine Learning (Information Science and Statistics),* Springer-Verlag, Berlin, Heidelberg.

[14] Silva, I., Spatti, D., Flauzino, R. A., Bartocci Liboni, L., and Reis Alves, S., 2017, *Artificial Neural Networks*, Springer Int. Publ., Switzerland.

[15] Rosenblatt, F., 1958, “The perceptron: a probabilistic model for information storage and organization in the brain,” Psychological review, **65**(6), pp. 386–408.

[16] Fukushima, K., 1975, “Cognitron: A Self-Organizing Multilayer Neural Network,” Biol. Cybernetics, **20**, pp. 121–136.

[17] Hubel, D. H., 1961, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” The Journal of physiology, **160**(1), pp. 106–54.

[18] Lecun, Y, Bottou, L., Bengio, Y., and Haffner, P., 1998, "Gradient-based learning applied to document recognition," Proc. IEEE, **86**(11), pp. 2278-2324.

[19] Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012, “ImageNet Classification with Deep Convolutional Neural Networks,” *Proc. of the 25th Int. Conf. Neural Inf. Process Systems,* **1**, pp. 1097–1105.

[20] Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y., 2009, “What is the best multi-stage architecture for object recognition?,” *2009 IEEE 12th Int. Conf. Computer Vision,* Kyoto, Japan, pp. 2146-2153.

[21] Kingma, D.P. and Lei Ba, J., 2015, “Adam: A Method for Stochastic Optimization,” *ICLR,* San Diego, CA.

[22] Chaumette, F. and Hutchinson, S., 2006, “Visual servo control. I. Basic Approaches,” IEEE Robotics & Automation Magazine, **13**(4), pp. 82–90.

[23] Bateux, Q., Marchand, E., Leitner, J., Chaumette, F., and Corke, P., 2018, “Training Deep Neural Networks for visual servoing,” *2018 IEEE Int. Conf. Robotics and Automation (ICRA).*

[24] Tokuda, F., Arai, S., and Kosuge, K., 2021, “Convolutional Neural Network-Based Visual Servoing for Eye-to-Hand Manipulator,” IEEE Access, **9**, pp. 91820-91835.

[25] Fischer, P., et al., 2015, “FlowNet: Learning Optical Flow with Convolutional Networks,” from https://www.arxiv-vanity.com/papers/1504.06852/.

[26] Gudovskiy, D. A., Hodgkinson, A., and Rigazio, L., 2018, “DNN Feature Map Compression using Learned Representation over GF(2),” from https://arxiv.org/abs/1808.05285.

[27] Ribeiro, E. G., de Queiroz Mendes, R., and Grassi, V., 2021, “Real-time deep learning approach to visual servo control and grasp detection for autonomous robotic manipulation,” Rob. Auton. Syst., **139**, p. 103757.

[28] Lenz, I., Lee, H., and Saxena, A., 2013, “Deep Learning for Detecting Robotic Grasps,” from https://arxiv.org/abs/1301.3592.

[29] Yu, C., Cai, Z., Pham, H., and Pham, Q.-C., 2019, “Siamese Convolutional Neural Network for Sub-millimeter-accurate Camera Pose Estimation and Visual Servoing,” from https://arxiv.org/pdf/1903.04713.pdf.

[30] Liu, J. and Li, Y., 2020, “Visual Servoing with Deep Learning and Data Augmentation for Robotic Manipulation,” JACIII, **24**, pp. 953–962.

[31] Zhou, X., Lan, X., Zhang, H., Tian, Z., Zhang, and Y., Zheng, N., 2018, “Fully convolutional grasp detection network with oriented anchor box,” *2018 IEEE/RSJ Int. Conf. Intelligent Rob. Syst. (IROS)*, IEEE, Madrid, Spain, pp. 7223–7230.

[32] Saxena, A., Pandya, H., Kumar, G., Gaud, A., Krishna, K. M., 2017, “Exploring convolutional networks for end-to-end visual servoing,” *2017 IEEE/RSJ Int. Conf. Intelligent Rob. Autom. (ICRA)*, IEEE, Marina Bay Sands, Singapore, pp. 3817–3823.

[33] Glocker, B., Izadi, S., Shotton, J., Criminisi, A., 2013, “Real-time rgbd camera relocalization,” *2013 Int. Symposium on Mixed and Augmented Reality (ISMAR)*, IEEE, Adelaide, Australia, pp. 173–179.

[34] Raj, P., Namboodiri, V. P., and Behera, L., 2020, “Learning to Switch CNNs with Model Agnostic Meta Learning for Fine Precision Visual Servoing” from https://arxiv.org/pdf/2007.04645.pdf.

[35] Sadeghi, F., 2019, “Divis: Domain invariant visual servoing for collision-free goal reaching,” from https://arxiv.org/abs/1902.05947.

[36] Pinto, L. and Gupta, A., 2016, “Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours,” *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 3406–3413.

[37] Dosovitskiy, A., Fischer, P., Ilg, E., Husser, P., Hazirbas, C., Golkov, V., v. d. Smagt, P., Cremers, D., and Brox, T., 2015, “Flownet: Learning optical flow with convolutional networks,” *2015 IEEE Int. Conf. Computer Vision (ICCV)*, pp. 2758–2766.

[38] S. Felton, P. Brault, E. Fromont and E. Marchand, "Visual Servoing in Autoencoder Latent Space," in IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 3234-3241, April 2022, doi: 10.1109/LRA.2022.3144490.

[39] A. Al-Shanoon, Y. Wang, and H. Lang, ‘DeepNet-Based 3D Visual Servoing Robotic Manipulation’, *Journal of Sensors*, vol. 2022, pp. 1–13, 03 2022.

[40] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, ‘Learning Predictive Representations for Deformable Objects Using Contrastive Estimation’, arXiv [cs.LG]. 2020.

[41] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint, 2018.