Iterative Optimisation with an Innovation CNN

PhD Thesis Proposal Review

Gerard Kennedy
U5185867

Draft in progress

Updated On 15 July 2020

Student: Gerard Kennedy

Chair Supervisor: Prof. Robert Mahony

Associate supervisors: Dr. Xin Yu, Assoc. Prof. Nick Barnes

Abstract

State estimation is a fundamental problem in computer vision and robotic vision. Classical computer vision problems involve extracting all relevant information from a single image or set of images. Conversely in robotic vision, the images are captured in real-time and require filtering algorithms that track and update target information. In this report I present a deep learning framework that aims to be applicable in both these scenarios. I introduce a novel CNN architecture, termed an 'Innovation CNN', that estimates the innovation (difference between current state and desired/expected state) for a system. The Innovation CNN can be applied in a stochastic gradient descent framework to refine/update a state estimate using standard numerical optimisation techniques. The Innovation CNN can also be applied in a filter algorithm to update an evolving state estimate. In this report I discuss the literature for online and offline state estimation that is relevant to the Innovation CNN. The literature review particularly focuses on the initial target problem; object pose estimation. I also present the formulation, network architecture, and initial results for the application of the Innovation CNN to object pose estimation. Finally, I outline a research plan for the remainder of my candidature.

Contents

Publications to Date

Conference (Accepted)

G. Kennedy, V. Ila, R. Mahony, *A Perception Pipeline for Robotic Harvesting of Green Asparagus*, 6th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2019

Chapter 1

Introduction

1.1 State Estimation

In control theory, a *state* is an internal representation of a system that is sufficient to fully define the future evolution of all system variables. A *state estimator* is an algorithm that enables the extraction of information about features of a system that are not explicitly provided by the available data. The problem of obtaining a state estimate is relevant to applications within a variety of areas, from robotics and computer vision to economics and finance. For example, in robotics a state estimator could estimate the motion of the robot, given information from the robot's sensors. Likewise, many applications within computer vision involve estimating attributes of a scene, such as 3D structure, from a given image. These two examples highlight a key distinction between state estimator applications: those applied online and those applied offline. The types of algorithms used for state estimation are generally different for each of these two application categories. This report proposes the use of a deep neural network (DNN) as the update component of a state estimation algorithm. This state update term is often referred to as the *innovation*; the difference between the current state and the predicted state. It is expected that this concept could be incorporated into algorithms for either online or offline state estimation. This report therefore also provides an overview of state estimation problems in both these categories, limited to applications within robotics and computer vision. Particular emphasis is placed on the initial target application; offline object pose estimation from a single image.

1.2 Offline State Estimation

Offline state estimation is at the core of many problems in computer vision. Such problems typically receive image data as input and estimate features of the scene such as object type or pose, depth, or 3D structure. As the information comes in the form of images a particular algorithm architecture, the Convolutional Neural Network (CNN) is widely used in such applications. CNNs attempt to 'learn' a function that best approximates the underlying data distribution present in the provided image. This is a challenging proposition, as state spaces of most learning problems are very large, non-linear, and high-dimensional. This is challenging because the algorithm must learn to converge from a wide range of initial conditions to a very precise result. For this reason it is often helpful to separately learn a state update that can be applied to the initial estimate as a refinement (or innovation). In this way the search space is constrained by the training data to the region of the initial estimate. This approach has recently proved beneficial in research areas such as object detection and object pose estimation.

1.3 Online State Estimation

Online state estimation is fundamental in robotics, as robots generally have to interact with a temporally changing environment. State estimation problems in robotics include robotic manipulation, visual odometry, and simultaneous localisation and mapping (SLAM). Such problems involve continuously updating a state estimate as new information becomes available from the robot's sensors. Algorithms that have been developed for such problems are often referred to as 'filters' or 'observers'. Such algorithms typically have an innovation term, which is obtained from the previous state and current sensor measurements. The innovation is generally computed analytically rather than learnt, as the underlying state space is much more constrained.

1.4 Innovation CNN

This report presents a deep learning-based approach to refining a state estimate. The approach applies the generalisation capabilities of deep learning to a search space confined to the region of the initial estimate. The approach involves estimating the gradient (for fixed input applications), or the innovation for (for iterative applications). The gradient can be used along with standard optimisation algorithms such as gradient descent to obtain a refined state estimate. Likewise the innovation estimate can be used along with typical filtering algorithms such as visual odometry to predict the next state. While a learnt innovation is less optimal than one that is solved for analytically, it allows well established filtering and observer-based algorithms to be implemented even when such a solution is not available.

1.5 Report Structure

The remainder of this report is structured as follows: Chapter 2 presents a review of the relevant literature, including classical approaches to state updating and refinement, and state estimation as a supervised learning problem, with particular emphasis on the initial target problem of 6D object pose estimation from an RGB image. Chapter 3 outlines the current research and approach being undertaken to solve the initial target problem. Note that this chapter outlines two distinct research directions; robotic harvesting and innovation CNNs. Robotic harvesting was the focus of this project for the initial 6 months, before a number of circumstances led to a change of direction. Therefore, the work undertaken related to robotic harvesting is briefly summarised in this section, and the relevant publication is included in the Appendix. All other chapters in this report make no reference to this work on robotic harvesting, but it has been included to provide a complete account of work undertaken to date. Chapter 4 discusses immediate next steps and future target problems. Finally, Chapter 5 summarises expected project resource requirements and outputs.

Chapter 2

Literature Review

In the following review, the problem of state estimation has been divided into two broad categories: offline state estimation and online state estimation. Applications within these categories that are reviewed are limited to those that are relevant to robotics and/or computer vision. Offline estimation refers to problems involving a fixed input, typically an image in the relevant applications, from which information such as depth, camera or object pose estimation, or object detection can be estimated. Online state estimation, often also referred to as 'filtering' or 'observing', involves estimating the state based on the previous state and current sensor input in an temporal process. Applications in this category include visual odometry, robotic manipulation, optical flow estimation, and simultaneous localisation and mapping (SLAM). The following review is divided into two sections corresponding to these two categories, with particular attention payed to the initial target application: object pose estimation.

2.1 Offline State Estimation

Applications of offline state estimation are wide-ranging in computer vision. All applications involving static input information, such as an image, from which implicit information is extracted can be considered an application of offline state estimation. Such applications include object detection, depth estimation, deblurring, super-resolution, denoising, and object pose estimation, [1, 2]. A common approach to such problems is to formulate them as either classification or regression problems, which can then be solved in a single step using a DNN (often a CNN). Such an approach is an example of data-driven modelling, as the algorithms used to estimate the state rely on training data for their accuracy. Classification and regression via a DNN has become a very popular approach to offline state estimation in computer vision in the previous two decades. This revolution is largely due to the current abundance of training data, and the ever-increasing capacity of computational resources. An overview of some relevant applications is provided below.

Object detection involves locating and classifying objects in an image and labeling them with a bounding box indicating detection confidence [3]. The two typical approaches to object detection involve either a) generating region proposals, extracting high-level features, and classifying the regions [4, 5], or b) directly via either regression [6, 7] or classification [8] with a single network. Approach a) is viewed as the traditional method, while method b) has been introduced mainly to allow for real-time applications. It is reasonable to assume that a network formulated to estimate object pose via regression could be reformulated to estimate the gradient toward the correct pose.

Depth estimation involves regressing a per-pixel depth value from a given image. A common approach is to use an autoencoder network architecture involving a pretrained feature extractor paired with a novel decoder in a supervised learning framework [9, 10, 11]. Due to a lack of annotated data, there is also substantial interest in self-supervision using a rectified stereo image pair [12, 13]. Similarly to the previous application, a decoder trained to output a gradient toward the correct depth appears feasible.

Deblurring involves removing Gaussian blur and motion blur from an image [14]. Super-resolution involves restoring high-resolution images from one or more low resolution images [15]. Denoising involves restoring an image corrupted by Gaussian noise to its original form [16]. In each case, a widely used technique is to use a CNN to regress to the desired state in an end-to-end framework. Thus it appears feasible to reformulate these problems into the proposed iterative refinement framework.

Object pose estimation has been chosen as the initial target problem for this research, and as such is discussed in greater detail below.

2.1.1 Pose Estimation

The initial problem chosen to evaluate the proposed concept is 6D object pose estimation from a single RGB image. This problem is an example of offline state estimation typically performed in a single step with a CNN. Obtaining an accurate pose of a target object within an image has a range of real world applications, including robotics, scene understanding, augmented reality, and virtual reality. This problem is made easier if RGBD images are utilised, however, lack of reliability and environmental constraints mean that this is not always possible. Pose estimation from a single RGB image is therefore an important research problem. It is also a highly challenging problem due ambiguity inherent in the visual appearance of objects from different viewpoints. Due to symmetries objects often appear the same from a range of viewpoints. Research in object pose estimation can be sub-divided into four general areas since the deep learning revolution. These areas are: end-to-end regression from image to pose, pose classification via a discretised pose space, regression to an intermediate representation such as keypoints, and pose refinement. These areas are discussed in detail below.

2.1.1.1 End-to-End Regression

End-to-end pose regression from a single image is a highly challenging task due to the high dimensional, nonlinear space of rigid 3D rotations $SO(3)$. This is addressed in [17] by separating and carefully balancing translation and rotation terms in the loss function, although this is for camera pose estimation, a slightly different problem. More recently, [18] decouple or 'disentangle' translation and rotation terms by moving the centre of rotation to the centre of the object and representing translation in 2D image space. In practice, regression to 3D orientation has had limited success [19, 20]. This is partly due to such methods only covering the small subsection of pose space that is seen during training [21], which has in turn led to discretisation of the pose space as discussed below.

2.1.1.2 Pose Classification

Pose classification has typically involved discretising $SO(3)$, while the translation component is obtained via regression. However, the dimensionality of $SO(3)$ still provides difficulty, as even a coarse discretisation of $\sim 5^\circ$ leads to over 50,000 possible cases [20]. [21] approach this problem by decomposing 6D pose into viewpoint and discretised in-plane rotation. However, as noted in [20], this approach leads to ambiguous classifications as a change of viewpoint can be nearly equivalent to a change of in-plane rotation. Furthermore, such classification leads to coarse pose estimates that require further refinement [21, 22], as discussed below.

2.1.1.3 Pose via an Intermediate Representation

A recent, popular approach to pose estimation is to first regress to an intermediate representation such as keypoints, from which pose can be obtained via 2D-3D correspondences and a PnP algorithm [23, 24, 25, 26, 27]. Of these approaches, some, including [23, 24] regress to a set of bounding box corners. Such methods encounter difficulty when objects are occluded or truncated. Alternatively, [25] predicts pixel-wise unit-vectors that in turn indicate direction to keypoints. [26] implements these vector directions from [25] while also estimating object edge vectors and symmetry correspondences. Finally, [27] textures objects with a 2D image generated from spherical or cylindrical projections, and uses these to estimate correspondences.

2.1.1.4 Benchmarks

Popular datasets for object pose estimation include LINEMOD [28], Occlusion LINEMOD [29], and YCB-Video [18]. Popular metrics for this task include the average distance (ADD) metric proposed by [28] (or ADD(-S) for symmetric objects), and the 2D projection metric which computes the mean distance between 2D projections of 3D model points. A pose is considered 'correct' in terms of the ADD metric if it is less than 10% of the models diameter, or in terms of the 2D projection error if it is less than 5 pixels.

2.1.2 Numerical Optimisation

Offline state estimation does not necessarily need to be performed in a single step. For some applications, it is helpful to iteratively step towards the optimal solution, thus reducing the search space at each iteration. Such problems typically involve utilisation of first order optimisation algorithms such as gradient descent, or second order optimisation algorithms such as Newton's method. An application in this category involves determining the robotic joint angles that provide a desired configuration. This problem is known as inverse kinematics, and is typically solved via either the image Jacobian, or a numerical optimisation approach such as Newton's method [30]. Beyond robotic vision, applications of numerical optimisation are numerous, ranging from investment portfolio management to computing the optimal shape of mechanical components [31]. Due to the widespread applicability of these methods, substantial research has been devoted to developing them [31, 32].

Recently, neural networks have been applied to problems typically solved with numerical optimisation, such as inverse kinematics [33], and 3D reconstruction [34]. Similarly, neural networks have been employed to refine a state estimate provided from a preceding algorithm, such as a different network. Applications that have made use of this approach include the initial target application, pose estimation, as discussed below.

2.1.2.1 Pose Refinement

Of the networks discussed above in Section 2.1.1, several have additional refinement steps that can be added to improve results. For example, [22] is presented as a refinement step for [18], [35] is presented as a refinement step for [21], and [27] is presented along with a refinement step. All these methods perform refinement using the additional input of a 3D model of the target object. In each case a synthetic image of the target object is rendered using the initial pose estimate and the 3D model. The key difference in these approaches is in the representation of the relative pose. [22] utilises an 'untangled representation' in which the centre of the object is the centre of rotation, and translation is estimated in pixel space. [35] represents rotation as a unit quaternion and translation as a vector in $\mathbb{R}^3$. [27] combines these approaches by estimating rotation relative to the object centre, and estimating translation as a vector in $\mathbb{R}^3$. These approaches therefore differ mainly in their loss functions, and all require annotated 3D object models. Figure 2.1 illustrates the approach used in [22]. Such networks are conceptually similar to the network proposed in this report, namely, they receive an input image and a pose estimate, and return an updated pose estimate. Unlike these approaches however, our method does not require an additional 3D object model, and can be applied more generally to a variety of applications not limited to pose estimation. It also iteratively improves the pose estimate based on established optimisation techniques rather than via visual similarity. It is therefore feasible to take the refined estimate produced by our network and refine it further using a pose refinement network which utilises the additional information of the 3D model.

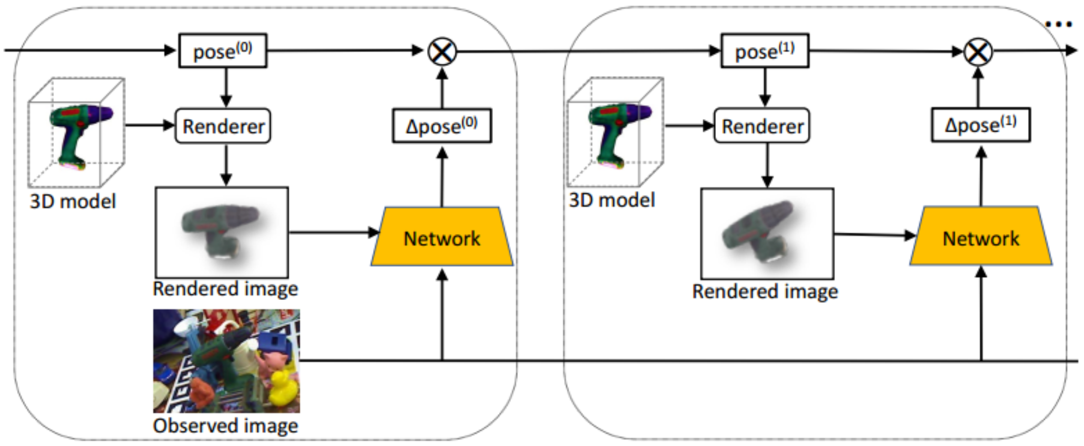


Figure 2.1: DeepIM [22] iterative matching architecture.

2.2 Online State Estimation

State estimators involving online data input are often referred to as filters or observers, which are examples of knowledge-driven modelling. Applications in this category include visual servoing, visual odometry (VO), optical flow estimation, and simultaneous localisation and mapping (SLAM).

Visual servoing involves using visual feedback to control a robot [36] for tasks such as manipulation and grasping. Such tasks are generally solved by minimising an associated error function, using similar approaches as those used for inverse kinematics. More recently, tasks such as robotic grasping have incorporated learning-based pose estimation, followed by classical control [37], or learning the control function itself [38].

Visual odometry (VO) involves estimating the egomotion of an agent using only the input from one or more cameras, and is updated iteratively as new image data becomes available [39]. VO is generally solved by computing relative camera pose via 3D-to-2D correspondences and a PnP algorithm. Currently popular approaches to VO include [40], which estimates relative camera pose by minimising photometric error across image patches, and [41], which minimises photometric error across image features, and jointly optimises for camera intrinsics, extrinsics and inverse depth. There has been recent interest in approaching VO from a learning perspective. Such work includes [42], which infers pose directly from a sequence of images for VO using a Recurrent Neural Network (RNN). Similarly, [43] constructs a VO algorithm by combining CNNs for depth and velocity estimation.

Optical flow (OF) estimation involves estimating per-pixel motion between consecutive frames in an image sequence. OF has traditionally been approached as an optimisation problem based on hand-crafted image features [44]. Recent work [45, 46] have cast OF as a learning problem, using iterative residual refinement and gated recurrent units respectively. [45] is particularly similar to our approach, as they learn the residual and then reapply the same network to iteratively improve the estimate. However, rather than taking our approach of learning the residual based on an initial estimate, [45] instead learn based on the combined performance over the total number of iterations.

The simultaneous localisation and mapping (SLAM) problem extends VO to include an estimation of the environment state. This requires a large state vector, of the order of the number of landmarks. Solutions to the SLAM problem typically involve a Bayesian filter such as the Extended Kalman Filter [47], or optimisation-based [48, 49]. Recently a new approach has emerged, formulating SLAM as a non-linear observer problem [50, 51]. The advantage of this is it removes the necessity to linearise the system equations. Like previous applications, there have also been recent efforts to apply learning to the SLAM problem. In some cases learning has been incorporated as front-end feature extraction [52, 53] for an optimisation-based back-end. Efforts have also been made to develop an entirely learned SLAM pipeline [54].

2.3 Summary & Common Challenges

Problems involving offline state estimation via regression or classification from a fixed-input are typically solved with a data-driven modelling approach such as a deep learning algorithm. Estimates provided by these algorithms can be refined further through the use of a separate neural network, as discussed above for pose refinement. This refinement is generally done without incorporating any knowledge-driven optimisation techniques. Conversely, problems involving online state estimation often utilise knowledge-driven modelling approaches involving numerical optimisation or filtering algorithms. Some recent work has explored the possible uses of deep learning within these problems, as discussed in Section 2.2. In general, such hybrid methods tend to replace some component of a classical algorithm, such as front-end feature extraction, with a neural network. There has been relatively little focus within these applications on the opposing problem, namely combining knowledge-driven techniques with data-driven techniques within offline estimation problems. This leaves room for the possibility that combining the capacity of learning with the rigour of optimisation algorithms may lead to improved results in such applications.

Recent work in this direction includes [55] and [56]. The goal of [55] is to learn an optimisation algorithm that can be used to train a network. The aim of learning an optimisation algorithm is similar to our own, although we manually choose the step size and explicitly learn the gradient with a CNN, whereas [55] learn a general update term in place of the step size and gradient with an LSTM. However, the application, an optimisation algorithm with which to train a network, is very different to our application, learning the gradient of an optimisation algorithm within which to apply a network iteratively.

The work by [56] involves learning a gradient descent scheme for iteratively applying a neural network to solve a problem. This scheme involves using a Recurrent Inference Machine [57] to learn the iterative framework. These works thus involve combining learning with optimisation strategies by attempting to learn the optimisation framework itself.

Most similar to our work is the work of [45] (Figure 2.2), which implements a residual-based refinement scheme for optical flow estimation. This work builds upon previous optical flow literature, in which it is common to implement a coarse-to-fine refinement scheme via pyramid levels within one network, or by chaining multiple networks. [45] extend this work by demonstrating that equivalent or better results can be obtained with equivalent or less parameters by substituting the network chain for a single network with a single set of weights. Key differences to our work include the use of a unit step size when updating the estimate using the residual. Furthermore, as mentioned above, while we supervise on a per-iteration basis, [45] combine the per-iteration loss into an overall loss.

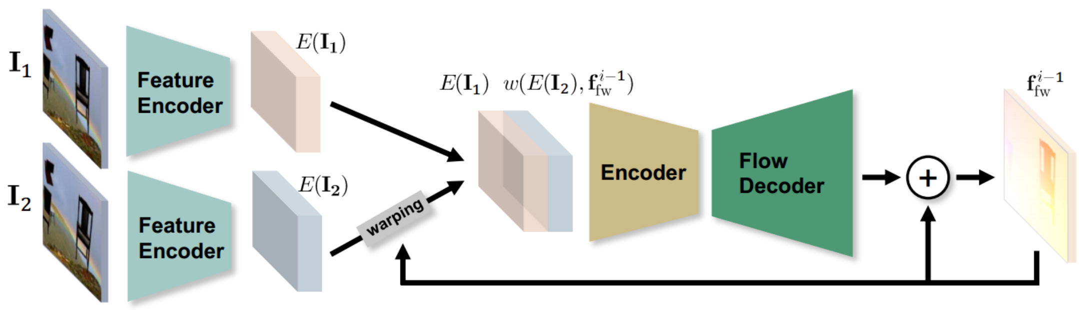


Figure 2.2: Iterative Residual Refinement of FlowNetS [45]

The work discussed in this section either aims to learn the residual [45], or a general update term of an optimisation algorithm [55, 56]. However, we believe there is merit in explicitly learning the state gradient, while using a separate step-size parameter to control the influence of the gradient on the iterative update. To the best of our knowledge, there is limited prior research that investigates the utility of a learning a gradient that can be combined with a scaling parameter to solve general estimation problems. Such an approach, if implemented correctly, could be expected to combine the benefits of data-driven learning algorithms and knowledge-driven optimisation algorithms.

Chapter 3

Research Report

This chapter outlines the planned research direction to be undertaken over the course of this project. The chapter is separated into four key sections. Section 3.1 provides a brief overview of the work undertaken in the development of the vision system for a robotic harvester of green asparagus. Section 3.2 outlines the ongoing work involving using a CNN in an iterative optimisation framework for object pose refinement.

3.1 Asparagus Harvesting

This project began with a focus on robotic harvesting of green asparagus in on-farm conditions. The aim was to develop a harvesting platform containing a Delta robot manipulator, and four monocular cameras. This involved developing a software architecture capable of detecting and localising asparagus spears in real-time from a multi-camera configuration. The outcome of this work was published in [58], and is available in the appendix.

A key problem identified during this work involved estimating the state of an asparagus spear (its position and height in this case), and then tracking this state over successive image frames. A wide range of filtering algorithms are available for problems such as this, and the Kalman Filter [59] was chosen for this application. Such algorithms receive an initial state estimate, and compute an innovation term which is then used to update the state in an iterative framework. An alternative to such algorithms is to simply re-estimate the state at each new time step. This is the approach generally taken when deep learning frameworks such as Convolutional Neural Networks (CNNs) are used for the state estimation, as such networks are typically not designed to receive an initial estimate of the target state as input. Other deep learning frameworks such as Recurrent Neural Networks (RNNs) are able to refine a state estimate based on temporal information. However, these networks learn how to refine the state based on the input data, with no theoretical justification for the resultant output. For this reason, this project has changed focus toward a new type of CNN, an 'Innovation CNN', which directly learns the innovation term of a filtering algorithm. This allows for the state to be refined using the standard techniques of numerical optimisation. Innovation CNNs are the current focus of this project, and are presented below.

3.2 Innovation CNN

3.2.1 General Problem Formulation

As outlined in Section 1.1, a state is an internal representation of a system that is sufficient to fully define the future evolution of all system variables. Let the state be constructed from some given input information $\mathbf{I}$, for which there is a state estimator that computes the state representation $\boldsymbol{X}_t$ from this information. Let $\mathbf{y}_t$ be the set of information that is of interest for a given problem. Then there is a mapping $h : \boldsymbol{X}_t \to \mathbf{y}_t$ (from the state to the desired information) that captures this information. Given this mapping, a state estimator can be formulated:

$$\begin{aligned} \dot{\mathbf{X}}_t &= f(\mathbf{X}_t, \mathbf{I}_t) - \Delta \\ \mathbf{y}_t &= h(\mathbf{X}_t) \end{aligned}$$

where $f(.)$ is the state model, and Δ is the innovation term. In typical problems in computer vision, the input data from which the state is constructed is in the form of an image or a sequence of images. These image(s) are obtained prior to any estimation, and the state is then estimated in an offline manner, typically via a deep learning framework. There is therefore an assumption that there is enough information in the given image data to reconstruct the state $\mathbf{X}_t$. In computer vision the desired representation $\mathbf{z}_t$ may represent scene depth, 3D structure, human or object pose, etc. Figure 3.1 provides a general example of the offline problem of object pose estimation as it is typically approached. In the context of the above description of a state estimator, such problems in computer vision can be described by

$$\hat{\mathbf{X}} = h^{-1}(\mathbf{I}_t)$$

where h^{-1} is typically estimated with a deep learning framework. Conversely, in robotic vision the input data consists of image information plus other data, such as IMU measurements, which are provided at each time step. The information of interest is then commonly obtained via a filtering or observing algorithm, which is iteratively updated in an online manner. There is therefore an assumption that the input information has a low enough dimensionality to allow the state to be estimated in an online framework. However, unlike common computer vision applications there is not an assumption that a single image can provide sufficient information to estimate a state. In contrast, it is more common to use a sequence of images, and/or additional sensor information. In robotic vision the desired representation is often the pose of the robot. For example, a state estimator for a visual odometry application could be formulated via

$$\begin{aligned} \dot{\hat{\mathbf{X}}}_t &= f(\hat{\mathbf{X}}_t, \mathbf{I}_t) - \Delta \\ \hat{\mathbf{y}}_t &= h(\hat{\mathbf{X}}_t) \\ \text{where} \\ \Delta &= K(\hat{\mathbf{y}}_t - \mathbf{y}_t) \\ \mathbf{y} &= [p_1, p_2, ..., p_N] \end{aligned}$$

where K is a gain, and $p_1, ..., p_N$ represent robot and landmark pose vectors.

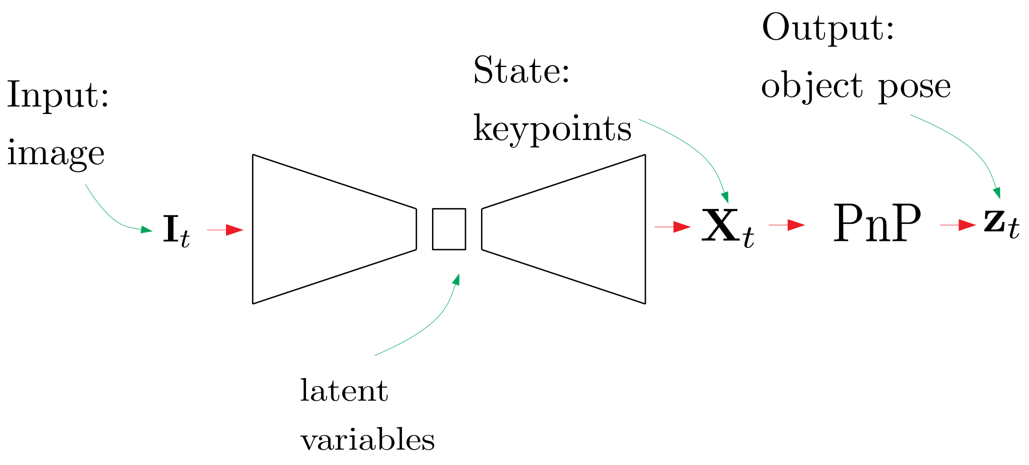


Figure 3.1: The standard approach to object pose estimation. In this problem the state is often represented by keypoints, which can be mapped the desired output with a PnP algorithm. The difference between the concepts of state and latent variables are also made clear in this context.

The remainder of this chapter focuses on the application of the Innovation CNN to the problem of object pose estimation.

3.2.2 Initial Application: Object Pose Estimation

As object pose estimation is an offline problem, we assume that

$$f(\hat{\mathbf{X}}_t, \mathbf{I}_t) = \mathbf{X}_{t-1}$$

Therefore

$$\hat{\mathbf{X}}_t = \hat{\mathbf{X}}_{t-1} - \Delta$$

The state can be estimated via stochastic gradient descent if the innovation is defined as so:

$$\text{let: } \Delta = \alpha \nabla_{\hat{\mathbf{X}}_t}$$
$$\text{then: } \hat{\mathbf{X}}_t = \hat{\mathbf{X}}_{t-1} - \alpha \nabla_{\hat{\mathbf{X}}_t}$$

where ∇_X represents the gradient with respect to X, and α is a step size. Therefore, for the offline problem of object pose estimation, the problem involves estimating the gradient of the underlying state representation. To obtain an initial estimate for the gradient descent framework, an off-the-shelf object pose estimation network is used. The network chosen is PVNet [25]. The output of this network provides the initial estimate for the Innovation CNN, and the network also provides the base architecture from which the Innovation CNN for object pose estimation is constructed.

3.2.2.1 Notation

The aim of the network is to generate an update/innovation that can be applied in an iterative framework to refine a state estimate. While this framework can be applied to a range of applications, the notation outlined here is specific to 6DOF object pose refinement from a single RGB image. The state estimate is a unit vector field with vectors defined by

$$\boldsymbol{\eta}_{ij}^k = \hat{\boldsymbol{\xi}}^k - \boldsymbol{\xi}_{ij}$$

where $\boldsymbol{\xi}_{ij}$ denotes a 2D pixel with coordinates (i, j) within an image $\mathcal{I}$ with dimensions $M \times N$, and $\hat{\boldsymbol{\xi}}^k$ represents the pixel location of keypoint k. The set of target keypoints points is denoted by $\mathcal{K}$ with individual points k. The unit vector field is thus

$$\hat{\boldsymbol{X}}_{ij}^{k} = \frac{\boldsymbol{\eta}_{ij}^{k}}{\max(1, ||\boldsymbol{\eta}_{ij}^{k}||_2)} \in \mathbb{R}^{2\times\mathcal{K}\times M\times N}$$

The initial state $\hat{\mathbf{X}}_{ij}^{k}(0)$ is the vector field estimate obtained from PVNet.

Our network learns the gradient of the original loss function Φ, which in this case is

$$\Phi = \frac{1}{2}||\mathbf{X}_{ij}^{k} - \hat{\mathbf{X}}_{ij}^{k}||_1^2$$

Therefore the gradient of the loss is

$$\begin{aligned} \nabla_{\Phi} &= \frac{1}{2}\nabla_{\hat{\mathbf{X}}_{ij}^{k}}||\mathbf{X}_{ij}^{k} - \hat{\mathbf{X}}_{ij}^{k}||_1^2 \\ &= -|\mathbf{X}_{ij}^{k} - \hat{\mathbf{X}}_{ij}^{k}|_1 \end{aligned} \tag{3.1}$$

where $\mathbf{X}_{ij}^{k}$ is the ground truth vector field.

The state estimate can then updated via a first order optimisation framework

$$\hat{\boldsymbol{X}}_{ij}^{k}(t_E + 1) = \hat{\boldsymbol{X}}_{ij}^{k}(t_E) + \delta\widehat{\nabla_{\Phi}}(t_E) \tag{3.2}$$

where $0 < \delta \leq 1$ is the step size used during evaluation, and $t_E \in [1, 2, ..., T_E]$ denotes the current step during evalulation. For clarity, there is an equivalent update equation during training:

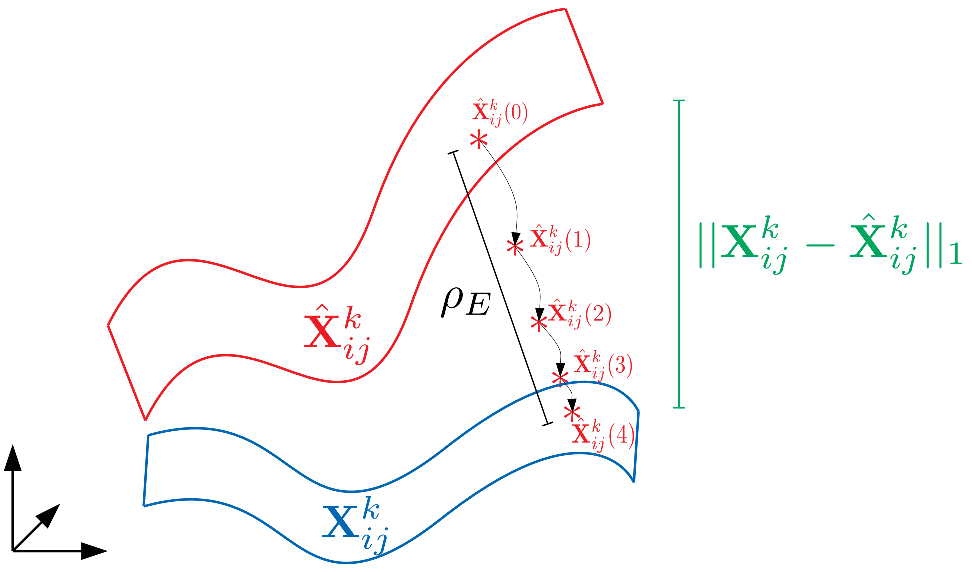
$$\hat{\boldsymbol{X}}_{ij}^{k}(t_T + 1) = \hat{\boldsymbol{X}}_{ij}^{k}(t_T) + \sigma\widehat{\nabla_{\Phi}}(t_T) \tag{3.3}$$

where $0 < \sigma \leq 1$ is the step size used during training, and $t_T \in [1, 2, ..., T_T]$ denotes the current step during training.

As iteration is employed during training via Equation 3.3, and during evaluation via Equation 3.2, the following equivalence holds

$$\rho_E = T_E\delta \triangleq T_T\sigma + \epsilon = \rho_T$$

where ρ_T is the 'training horizon' (the distance from the original data manifold that successive iterations have achieved), ρ_E is the 'evaluation horizon', and ϵ is the generalisation error associated with the network. The interpolation distance is the distance from the PVNet manifold, as illustrated in Figure 3.2.



$\hat{\mathbf{X}}_{ij}^k(0)$
$\hat{\mathbf{X}}_{ij}^k(1)$
$\hat{\mathbf{X}}_{ij}^k(2)$
$\hat{\mathbf{X}}_{ij}^k(3)$
$\hat{\mathbf{X}}_{ij}^k(4)$
$\hat{\mathbf{X}}_{ij}^k$
ρ_E
$||\mathbf{X}_{ij}^k - \hat{\mathbf{X}}_{ij}^k||_1$
$\mathbf{X}_{ij}^k$

Figure 3.2: Iterative refinement from the initial PVNet estimate manifold ($\hat{\boldsymbol{X}}_{ij}^k$) toward the ground truth manifold $\mathbf{X}_{ij}^k$ during an example forward pass with 4 iterations. The difference ($||\mathbf{X}_{ij}^k - \hat{\mathbf{X}}_{ij}^k||_1$) between input and output states is refined over the interpolation distance ρ_E via Eq 3.2.

Pseudo-code outlining how the iterative framework might be implemented is presented in Algorithm 1.

Algorithm 1 Iterative Optimisation with Innovation CNN

```
1: Choose $0 < \alpha < 1$
2: Choose $T > 0$                                    ▷ Maximum #iterations
3: $\hat{\boldsymbol{X}}_{ij}^k(0) \leftarrow$ PVNet
4: for $t = 1 \rightarrow T$ do
5:     $\widehat{\nabla_\Phi} \leftarrow$ Innovation CNN
6:     $\hat{\boldsymbol{X}}_{ij}^k(t) = \hat{\boldsymbol{X}}_{ij}^k(t-1) + \alpha\widehat{\nabla_\Phi}(t)$
7: pose = PnP($\hat{\boldsymbol{X}}_{ij}^k(T)$)
```

3.2.2.2 Network Architecture

The network consists of two encoder-decoder blocks operating in parallel (see figure 3.3). The top block in figure 3.3 is labeled the 'Gradient Estimator', and the bottom block is labeled the 'Estimate Autoencoder'. The Gradient Estimator in our application is PVNet [25] with a new loss function. The Gradient Estimator encoder consists of a pretrained ResNet-18 [60] architecture. The Gradient Estimator decoder consists of successive convolution and upsampling operations. See figure 3.4 for a breakdown of the individual components of the network. The Estimate Autoencoder is also obtained from PVNet, with the input dimensionality modified to accept a vector field state estimate instead of an RGB image. The Gradient Estimator receives an image and returns an estimate of the gradient of the vector field state. The Estimate Autoencoder receives a vector field state and returns an estimate of the vector field state. The Estimate Autoencoder is therefore an auxiliary task aimed at allowing information from the initial state estimate to be injected into the network. This auxiliary task is expected to lead to improvement on the main task by leveraging domain specific information via multi-task learning [61]. The network is trained with the loss functions described below in Section 3.2.2.4.

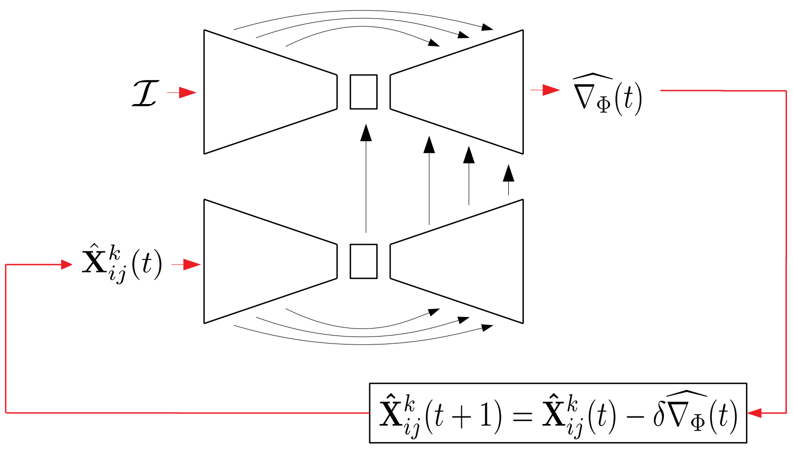

$\mathcal{I}$
$\widehat{\nabla_\Phi}(t)$
$\hat{\mathbf{X}}_{ij}^k(t)$
$\hat{\mathbf{X}}_{ij}^k(t+1) = \hat{\mathbf{X}}_{ij}^k(t) - \delta\widehat{\nabla_\Phi}(t)$

Figure 3.3: Network Architecture: Gradient Estimator (top block), and Estimate Autoencoder (bottom block). Skip connections are indicated with black arrows, and information flow is indicated with red arrows. $\mathcal{I}$ is an input image, $\widehat{\nabla_\Phi}$ is the estimate of the gradient of the state, $\hat{X}_{ij}^k$ is the state estimate, and $\hat{X}_{ij}^k$ is the encoded state estimate.

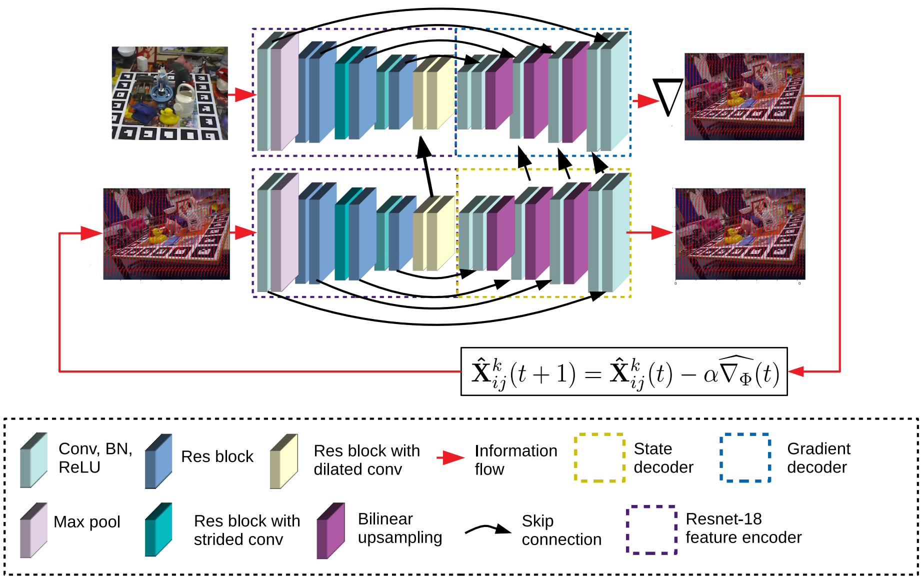

∇
$\hat{\mathbf{X}}_{ij}^k(t+1) = \hat{\mathbf{X}}_{ij}^k(t) - \alpha\widehat{\nabla_\Phi}(t)$
Conv, BN, ReLU
Res block
Res block with dilated conv
Information flow
State decoder
Gradient decoder
Max pool
Res block with strided conv
Bilinear upsampling
Skip connection
Resnet-18 feature encoder

Figure 3.4: Network Architecture: individual network components.

3.2.2.3 Training Loss

The loss function used to train the network is

$$\mathcal{L} = \mathcal{L}_{\nabla_\Phi} + \gamma \mathcal{L}_{\boldsymbol{X}}$$

where ∇_Φ is the gradient of the state loss on the Gradient Estimator, $\mathcal{L}_{\boldsymbol{X}}$ is the state loss on the Estimate Autoencoder, and γ is a scaling factor. These functions are defined below.

The state loss is applied to the Esimate Autoencoder, and is defined by

$$\mathcal{L}_{\boldsymbol{X}} = \sum_{k=1}^{\mathcal{K}} \sum_{(i,j)\in\mathcal{S}} ||\hat{\boldsymbol{X}}_{ij}^k - \bar{\boldsymbol{X}}_{ij}^k||_1^2$$

where $(i,j) \in \mathcal{S}$ indicates that the pixel is within the segmentation mask, and the $l1$ norm is the 'smooth l1 norm' for unit vectors proposed by [62].

The state gradient loss is applied to the Gradient Estimator, and is defined by

$$\begin{aligned}\mathcal{L}_{\nabla_\Phi} &= \sum_{k=1}^{\mathcal{K}} \sum_{(i,j)\in\mathcal{S}} ||\widehat{\nabla_\Phi} - \nabla_\Phi||_1 \\ &= \sum_{k=1}^{\mathcal{K}} \sum_{(i,j)\in\mathcal{S}} ||\widehat{\nabla_\Phi} - (\hat{\boldsymbol{X}}_{ij}^k - \boldsymbol{X}_{ij}^k)||_1.\end{aligned}$$

3.2.2.4 Evaluation Metrics

The following three metrics are used to evaluate network performance. Of these metrics, the first two are the standard metrics for evaluating pose estimation networks, while the third is designed to further illustrate the outcome of iteratively applying Eq. 3.2.

Given the ground truth rotation $\mathbf{R}$ and translation $\mathbf{T}$ and the estimated rotation $\hat{\mathbf{R}}$ and translation $\hat{\mathbf{T}}$, the average distance ADD metric computes the mean of the pairwise distances between the 3D model points transformed according to the ground truth pose and the estimated pose:

$$\text{ADD} = \frac{1}{m} \sum_{\mathbf{x}\in\mathcal{M}} ||(\mathbf{R}\mathbf{x} + \mathbf{T}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{T}})||_2 \tag{3.4}$$

where $\mathcal{M}$ denotes the set of 3D model points and m is the number of points. The pose is considered correct if the average distance is smaller than a 10% of the 3D model diameter. The metric reported is the percentage of posses considered correct.

Projecting the provided object model into the image plane using the ground truth pose $\boldsymbol{X}_{ij}^k$ and the estimated pose $\hat{\boldsymbol{X}}_{ij}^k$ allows the 2D projection error to be computed. The estimated pose is considered correct if the average error is less than 5 pixels.

$$\text{2d Proj} = \frac{1}{|V|} \sum_{\mathbf{v}\in V} ||\mathbf{P}\mathbf{X}_{ij}^k\mathbf{v} - \mathbf{P}\hat{\mathbf{X}}_{ij}^k\mathbf{v}||_2 \tag{3.5}$$

where V is the set of all object model vertices, and $\mathbf{P}$ is the camera matrix.

Taking the square of the Frobenius Norm of the difference between the estimated and ground truth vector fields provides an indication of whether the state estimate

$\hat{\boldsymbol{X}}_{ij}^k$ is converging to or diverging from the true state $\boldsymbol{X}_{ij}^k$. This metric is defined by

$$\text{Norm}(\hat{\mathbf{X}}_{ij}^k - \mathbf{X}_{ij}^k) = \frac{1}{N}\frac{1}{||\mathcal{S}||}\sum_{n=0}^{N}\sum_{(i,j)\in\mathcal{S}} ||\hat{\mathbf{X}}_{ij,n}^k - \mathbf{X}_{ij,n}^k||_2^2. \tag{3.6}$$

3.2.2.5 Choosing the Step Size

A key benefit of our formulation is the ability to choose and vary the step size using knowledge-driven techniques. In the above formulation two step size parameters have been defined. The step size σ is applied to iteration during training, while δ is applied to iteration during evaluation. For convenience, α is used throughout this section to represent a generic step size.

In the numerical optimisation literature, a popular approach to choosing an appropriate step size is by imposing the Wolfe Conditions [31]. To apply these conditions we begin by defining the objective function

$$\begin{aligned} f_{t+1} = f(\hat{\mathbf{X}}_{ij}^k(t) + \alpha\widehat{\nabla_\Phi}(t)) &= \frac{1}{2}||\mathbf{X}_{ij}^k - (\hat{\mathbf{X}}_{ij}^k(t) + \alpha\widehat{\nabla_\Phi}(t))||_1^2 \\ &= \frac{1}{2}||\mathbf{X}_{ij}^k - \hat{\mathbf{X}}_{ij}^k(t+1)||_1^2 \end{aligned}$$

The gradient of the objective function with respect to the state estimate is thus

$$\begin{aligned} \nabla_{\hat{\mathbf{X}}_{ij}^k(t+1)} f_{t+1} &= -||\mathbf{X}_{ij}^k - \hat{\mathbf{X}}_{ij}^k(t+1)||_1 \\ &\triangleq \widehat{\nabla_\Phi}(t+1) \end{aligned}$$

using Eq 3.1. As the descent direction typically has the form

$$\widehat{\nabla_\Phi}(t) = -B^{-1}(t)\nabla_{\hat{\mathbf{X}}_{ij}^k(t)} f_t$$

where $B(t)$ is a symmetric and nonsingular matrix, it can be seen that we have implemented the method of steepest descent, for which $B(t)$ is the identity matrix [31]. Likewise,

$$\widehat{\nabla_\Phi}(t+1) = -B^{-1}(t+1)\nabla_{\hat{\mathbf{X}}_{ij}^k(t+1)} f_{t+1}.$$

When the descent direction is in this form we have

$$\begin{aligned}(\widehat{\nabla_\Phi}(t))^T \nabla_{\hat{\mathbf{X}}_{ij}^k(t)} f_t &= \nabla_{\hat{\mathbf{X}}_{ij}^k(t)} f_t^T \widehat{\nabla_\Phi}(t) \\ &= -(\nabla_{\hat{\mathbf{X}}_{ij}^k(t)} f_t)^T \nabla_{\hat{\mathbf{X}}_{ij}^k(t)} f_t \\ &= -||\nabla_{\hat{\mathbf{X}}_{ij}^k(t)} f_t||_2^2 \\ &= -||\widehat{\nabla_\Phi}(t)||_2^2 < 0\end{aligned}$$

which shows that $\widehat{\nabla_\Phi}$ must be a descent direction. Likewise,

$$(\widehat{\nabla_\Phi}(t+1))^T \nabla_{\hat{\mathbf{X}}_{ij}^k(t+1)} f_t = -||\widehat{\nabla_\Phi}(t+1)||_2^2 < 0.$$

Using this formulation, the following inexact line search condition can be applied to ensure sufficient decrease of the objective function

$$\begin{aligned}f(\hat{\mathbf{X}}_{ij}^k(t) + \alpha\widehat{\nabla_\Phi}(t)) &\leq f(\hat{\mathbf{X}}_{ij}^k(t)) + c_1\alpha\nabla_{\hat{\mathbf{X}}_{ij}^k(t)} f^T \widehat{\nabla_\Phi}(t) \\ \frac{1}{2}||\mathbf{X}_{ij}^k - (\hat{\mathbf{X}}_{ij}^k(t) + \alpha\widehat{\nabla_\Phi}(t))||_1^2 &\leq \frac{1}{2}||\mathbf{X}_{ij}^k - \hat{\mathbf{X}}_{ij}^k(t)||_1^2 - c_1\alpha||\widehat{\nabla_\Phi}(t)||_2^2 \qquad (3.7)\end{aligned}$$

for $c_1 \in (0,1)$. This inequality is the first Wolfe condition, also sometimes referred to as the Armijo condition.

To rule out unacceptably short steps, the second Wolfe Condition (or the Curvature Condition) requires that α satisfies

$$\begin{aligned}\nabla_{\hat{\mathbf{X}}_{ij}^k(t+1)} f(\hat{\mathbf{X}}_{ij}^k(t) + \alpha\widehat{\nabla_\Phi}(t))^T \hat{\mathbf{q}}_{ij}^k(t+1) &\geq c_2 \nabla_{\hat{\mathbf{X}}_{ij}^k(t)} f_t^T \hat{\mathbf{q}}_{ij}^k(t) \\ \nabla_{\hat{\mathbf{X}}_{ij}^k(t+1)}(\frac{1}{2}||\mathbf{X}_{ij}^k - \hat{\mathbf{X}}_{ij}^k(t+1)||_1^2)^T ||\mathbf{X}_{ij}^k - \hat{\mathbf{X}}_{ij}^k(t+1)||_1 &\geq c_2||\mathbf{X}_{ij}^k - \hat{\mathbf{X}}_{ij}^k(t)||_1^2 \\ ||\mathbf{X}_{ij}^k - (\hat{\mathbf{X}}_{ij}^k(t) + \alpha\widehat{\nabla_\Phi}(t))||_1^2 &\geq c_2||\mathbf{X}_{ij}^k - \hat{\mathbf{X}}_{ij}^k(t)||_1^2 \qquad (3.8)\end{aligned}$$

for $c_2 \in (c_1, 1)$.

To apply Equations 3.7 and 3.8 during training and evaluation, an appropriate step size α must be found at each iteration. This is done via the backtracking line search outlined in Algorithm 2.

Using Algorithm 2, Equations 3.7 and 3.8 are applied at each iteration t during training and evaluation to determine appropriate values for σ and δ (substituted for α) respectively.

Algorithm 2 Backtracking Line Search

```
1:  Choose $0 < \alpha < 1$
2:  Choose $T > 0$                                    ▷ Maximum #iterations
3:  $\eta = 1 - (1/T)$
4:  $t = 1$
5:  while True do
6:      if (Eq 3.7 and Eq 3.8) = True then
7:          break
8:      if $t \geq T$ then
9:          break
10:     $\alpha = \eta\alpha$
11:     $t+ = 1$
```

3.2.2.6 Initial Results

[illegible]

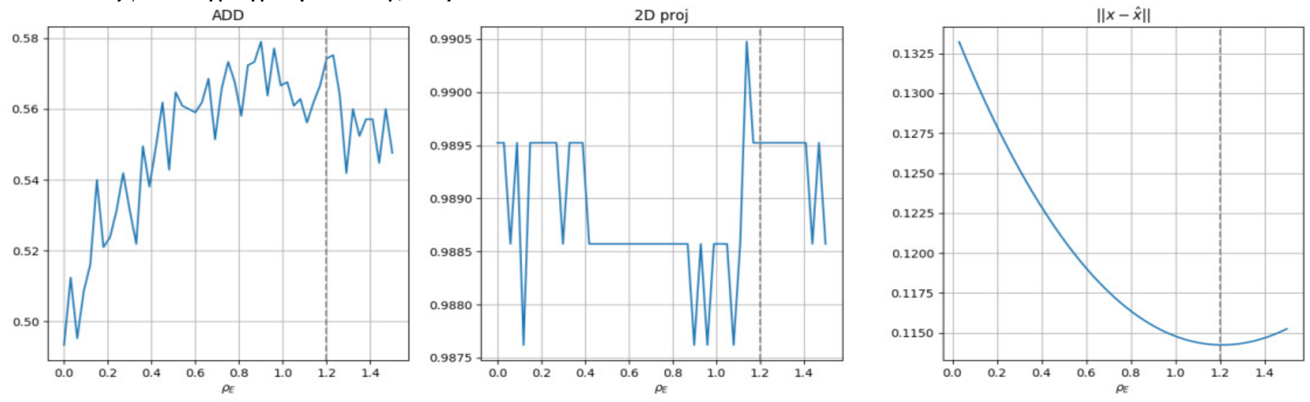


Figure 3.5: Qualitative Performance on the Ape class of the Linemod dataset [28]. The left panel illustrates iterative increase of the ADD metric. The central panel illustrates the iterative fluctuations of the 2D projection metric. The right panel illustrates the iterative decrease in the distance between the estimated and ground truth vector fields. In each panel, the x-axis is $\rho_E = \delta T_E$, the 'interpolation distance' during evaluation. In each panel, the grey dotted line illustrated ρ_T, the interpolation distance during training.

The results shown in Figure 3.5 are typical when an optimal set of training and evaluation parameters are chosen. When this is the case it can be seen that the ADD metric (reported as a percentage) increases logarithmically until the ρ_T threshold (marked by the vertical grey dashed line) is reached. As the network has only experienced training examples that are a distance of $\leq \rho_T$ from the manifold of the initial PVNet estimate, it is expected that performance begins to degrade beyond this point. From the centre panel of Figure 3.5 it can be seen that the 2D projection error is largely unchanged by successive iterations. This is likely due to the already very high value obtained by the initial estimate. Finally, the right-hand panel of Figure 3.5 indicates that the distance between the estimated and ground truth vector fields decreases consistently until the training interpolation distance is reached.

When the Wolfe conditions are applied during evaluation to the iterative update process, such as that shown in Figure 3.5, results such as those shown in Figure 3.6 are obtained.

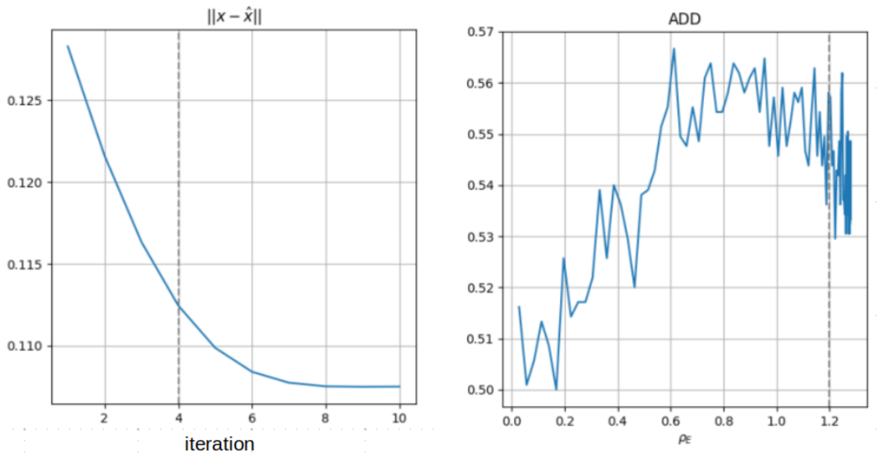


Figure 3.6: Application of Wolfe Conditions (Equations 3.7 and 3.8) to an example of iterative evaluation on the Ape dataset.

From Figure 3.6 it appears that the Wolfe conditions successfully stop the state estimate from diverging once the iterative framework extends past the training interpolation distance. It also appears that the Wolfe conditions assist in allowing the ADD metric to plateau at later iterations, as hoped.

Initial experiments have indicated that the performance of the network is sensitive to a range of training parameters. Of these, key parameters are σ (the step size during training), δ (the step size during evaluation), ρ_T (the interpolation distance during training), and ρ_E (the interpolation distance during evaluation). For this reason, a study was undertaken comparing network performance as these parameters were varied. The results of this experiment are presented in Table 3.1.

From Table 3.1 it can be seen that there is a large fluctuation in the mean increase in the ADD metric. It can also be seen that there is not always a strong correlation between an increase on the ADD metric and a decrease in the distance between ground

σ	ρ_T	δ	ρ_E	**mean(ADD)**	**mean(% increase ADD)**	**mean(% decrease norm(X-X^))**	**mean(ADD) T/P values**
1	1	1	2	0.424 +/- 0.04	-9.81 +/- 8.47	9.29 +/- 1.37	-2.12/0.05
1	2	1	4	0.520 +/- 0.03	10.33 +/- 6.70	6.75 +/- 1.33	3.76/0.001
0.9	1.8	0.9	3.6	0.518 +/- 0.03	9.48 +/- 7.17	7.56 +/- 1.02	3.61/0.001
0.6	1.2	0.6	2.4	**0.539 +/- 0.03**	**14.50 +/- 6.56**	**17.99 +/- 0.89**	**5.26/0.001**
0.6	2.4	0.6	4.8	0.464 +/- 0.01	-1.83 +/- 3.66	2.11 +/- 0.34	-5.16/0.001
0.3	1.2	0.3	2.4	0.471 +/- 0.01	0.44 +/- 2.79	1.91 +/- 0.34	-0.65/0.2+

Table 3.1: Study of Iteration Parameters. All mean and standard deviation values were computed from the last 20 epochs of training, from a total of 50 epochs. T-value is obtained from Welch's T-test of the mean(ADD) compared to the original PVNet distribution: 0.472+/-0.0067. P-value is obtained from corresponding probability that the two means come from separate distributions.

truth and estimated vector fields. This is likely due to the back-end, RANSAC and PnP-based computation that generates object poses given the vector field provided by the network. However, these initial results indicate that a step size of $0.5 \leq \delta = \sigma \leq 1$ leads to optimal performance.

Aside from the iteration parameters, a range of other design choices have been tested in a similar fashion. Tables 3.2 and 3.3 present the results of these tests. In Table 3.2, the design choices being investigated are

- GE/+SA: Gradient Estimator or Gradient Estimator and State Autoencoder. This comparison is designed to determine if including the State Autoencoder in the network architecture improves results.
- $\nabla_\Phi / \widehat{\nabla_\Phi}$. This comparison tests whether it is best to update the state with the ground truth gradient or the gradient estimated by the network during training.
- L_{∇_Φ}. This comparison tests whether it is best to scale (by $\frac{1}{||\nabla_\Phi||}$) this loss term.
- initial est. This comparison tests whether it is better to use the output of the baseline PVNet as the initial estimate during training, or whether it is preferable to use a perturbed ground truth.

In Table 3.3, the design choices being investigated are

- IRR: loss accumulates each iteration and is backpropogated after T_T iterations
- Innovation CNN: loss backpropogated after each iteration

GE/+SA	$\nabla_\Phi / \widehat{\nabla_\Phi}$	L_{∇_Φ}	initial est	mean(ADD)	mean(% increase ADD)	mean(% decrease norm(X-X^))	mean(ADD) T/P values	GE/+SA	$\nabla_\Phi / \widehat{\nabla_\Phi}$	L_{∇_Φ}	initial est
GE+SA	∇_Φ	**scaled**	**PVNet**	0.511 +/- 0.036	8.29 +/- 7.82	12.91 +/- 2.55	2.13/0.05		🟧		
GE+SA	$\widehat{\nabla_\Phi}$	**scaled**	**PVNet**	**0.508 +/- 0.043**	**7.51 +/- 9.12**	12.50 +/- 3.25	**1.38/0.2**		🟧	🟩	
GE+SA	$\widehat{\nabla_\Phi}$	**unscaled**	**PVNet**	0.470 +/- 0.066	-0.16 +/- 13.67	**13.60 +/- 2.55**	0.01/0.5+			🟩	
GE	∇_Φ	**unscaled**	**PVNet**	0.506 +/- 0.034	7.40 +/- 8.06	14.78 +/- 1.07	2.08/0.05	🟥			
GE+SA	∇_Φ	**unscaled**	**PVNet**	0.539 +/- 0.030	14.50 +/- 6.56	17.99 +/- 0.89	4.79/0.001	🟥			🟦
GE+SA	∇_Φ	**unscaled**	**GT +/- 10%**	0.460 +/- 0.008	-2.81 +/- 2.61	2.04 +/- 0.20	-10.85/0.001				🟦
GE+SA	∇_Φ	**unscaled**	**GT +/- 1%**	0.460 +/- 0.007	-2.74 +/- 2.07	1.84 +/- 0.11	-13.83/0.001				🟦

Table 3.2: Training Parameters. Experiments are colour-coded based on which design choices are being compared. The results of the experiment that performed best for a given pair of design choices is highlighted with the corresponding colour. T-value is obtained from Welch's T-test of the mean(ADD) compared to the original PVNet distribution: 0.472+/-0.0067. P-value is obtained from corresponding probability that the two means come from separate distributions.

Loss	$\sigma, \rho_T, \delta, \rho_E$	mean(ADD)	mean(% increase ADD)	mean(% decrease norm(X-X^))	mean(ADD) T/P values
IRR	1,2,1,4	0.401 +/- 0.04	-15.04 +/- 8.51	4.79 +/- 1.14	3.14/0.01
	1,4,1,4	0.457 +/- 0.03	-3.12 +/- 6.83	1.41 +/- 1.01	
Innovation CNN	1,2,1,4	**0.520 +/- 0.03**	**10.33 +/- 6.70**	**6.75 +/- 1.33**	**3.76/0.001**
	1,4,1,4	0.406 +/- 0.03	-13.95 +/- 7.72	3.21 +/- 1.46	

Table 3.3: IRR: Backprop after all iterations. Innovation CNN: Backprop each iteration. Both experiments use: $\sigma = 1$, $\rho_T = 2$, $\delta = 1$, $\rho_E = 4$. T-value is obtained from Welch's T-test of the mean(ADD) compared to the original PVNet distribution: 0.472+/-0.0067. P-value is obtained from corresponding probability that the two means come from separate distributions.

From Table 3.2 it appears that

- performance is relatively improved when the State Autoencoder is included in the network architecture
- performance is relatively improved when the ground truth state gradient is used to perform the iterative update during training
- performance is relatively improved when the L_{∇_Φ} loss is unscaled
- performance is relatively improved when the output of the original PVNet is used

Note: It is expected that a less naive method of perturbing the ground truth might improve network performance when using the ground truth as an initial estimate. Therefore further investigation is required to conclude that the best available initial estimate to use during training is the output of the original PVNet.

From Table 3.3 it appears preferable to backpropogate after each iteration.

From Tables 3.1, 3.2, 3.3 it can be seen that there is a consistent, large variation in the mean ADD increase between epochs. It can also be seen that a decrease in distance between the estimated and ground truth vector fields does not consistently correspond to a proportional increase in ADD value. As discussed above, this is likely due to the stochastic back-end which computes pose from the estimated vector field via RANSAC and PnP. This lack of consistency makes drawing concrete conclusions difficult. However, it is still possible to confirm the magnitude of the impact of a particular design choice by viewing the overall effect the choice has on the three metrics presented in these tables.

Chapter 4

Future Directions

In very general terms, my research involves looking for an answer to the following question, and studying the ramifications of the answer.

Can you train a neural network to learn the direction (gradient) from a given state to a desired state?

Why this question?

If the answer to this question is yes, then for any problem currently solved by a neural network it should be possible to devise a new network that predicts the direction from the output of the original network to a better output. Furthermore, if you then updated your original output using this direction and got a better result, you could then take this result and use the same network to once again predict the direction to an even better result. And so on. This strategy of iterative refinement given an initial estimate and a desired direction is known as numerical optimisation, and is a well-studied area of applied mathematics. This approach, then, provides an opportunity to leverage both the data-driven advantages of neural networks, and the knowledge-driven advantages of numerical optimisation within a single framework.

What is the utility?

A general network architecture capable of predicting a state gradient has widespread potential applications in both offline and online state estimation. Within offline state estimation, this architecture can potentially be applied to any problem currently solved with neural networks. Given the output of a neural network trained on the specific problem, such as PVNet for object pose estimation, the proposed network could predict a direction in which this output could be improved. Numerical optimisation techniques, such as the gradient descent algorithm can then be employed to iteratively refine the initial estimate. Within online state estimation, the aim is often to refine a state, given a previous state and some new measurements. The difference between the previous state and the state predicted by the new measurements is known as the *innovation*. To compute the innovation it is necessary to have a mathematical model of the state as a function of the given measurements. This is often referred to as the 'observation model', and a key task when implementing an algorithm for online state estimation involves formalising this model. There is potential for our proposed network to replace this model. In this case, the input to the proposed network would be the previous state and the current measurements, and the output would be the new state. Therefore, the proposed network architecture has widespread potential applications in both online and offline state estimation.

How best to demonstrate this utility?

The initial demonstration of utility is provided in Chapter 3, wherein the proposed network architecture is used to iteratively refine the initial output of PVNet, a network designed for the offline problem of object pose estimation. Beyond this initial demonstration, the goals outlined in the following sections are proposed.

4.1 0-6 months: Finalise Pose Refinement Work

Immediate next steps will involve finalising the current work on object pose refinement, and preparing for publication of this work. This will include adapting the network training regime to utilise a less naive perturbed ground truth state as the initial estimate, rather than the output of PVNet. The network will also need to be trained and tested on the Linemod, Linemod Occlusion, and YCB datasets, with full image resolution. These results will then be compared to other networks for pose estimation. There is also potential for multiple publication avenues for this work. A possible approach to publication might involve an initial submission demonstrating that Innovation CNNs can improve the performance of a chosen network. Work could then be undertaken to attempt to reach state of the art performance on the pose estimation problem, after which a second publication would be possible.

4.2 6-12 months: Depth Refinement with an Innovation CNN

The next proposed major piece of work involves applying the Innovation CNN concept to a second offline estimate problem; depth estimation from a single RGB image. By applying our concept to a second application I hope that we will be able to demonstrate it's general applicability to offline state estimation problems. This will aid in emphasising a key claim that the Innovation CNN is a general framework that can be applied to refine an initial estimate for a wide range of offline estimation problems. It will also serve to bring our work to the attention of another related research area. This work will also provide further insight into the ease with which our framework can be applied to a new problem and a new network architecture. The network architecture to be used for this work has tentatively been chosen to be FastDepth [10]. The motivation for this choice is the relative simplicity of the network architecture compared to recent, more sophisticated networks such as [9]. The simplicity of the architecture makes parts of this task suitable for an undergraduate student project. Such a project is already underway.

Aside from simply applying the concept to a new application, I am interested in potential ways to improve the Innovation CNN architecture and optimisation framework. Potential avenues for exploration include

- learning the step size and/or number of iterations
- learning a per-pixel step size (attention)
- varying the step size during training to enable better generalisation
- Using perturbed ground truth as the initial estimate during training to remove bias toward the initial estimate provided by the base network

4.3 12-24 months: State Filtering with an Innovation CNN

The applications of the Innovation CNN proposed above both involve offline pose estimation. Applying this concept to an online application involving state filtering or observing would therefore greatly expand the demonstrated utility of the concept. This is again likely to increase awareness for our concept within a wider research community, as it will demonstrate potential applicability with a range of popular state estimation algorithms, such as SLAM and VO. It would also further emphasis novelty of this concept, as one that is applicable to two largely separate problem areas of offline and online state estimation. A potential candidate algorithm that this concept could be incorporated into is the nonlinear observer of [51], as this would enable collaboration within the research group.

4.4 Long term: General Theory of Innovation CNNs

Finally, I would like to summarise and generalise the concept of Innovation CNNs for publication as a journal article. This work would provide an overview of the tested and expected applications within both online and offline estimation problems.

4.5 Timeline

Figure 4.1 provides a rough outline of a timeline for the plans discussed above.

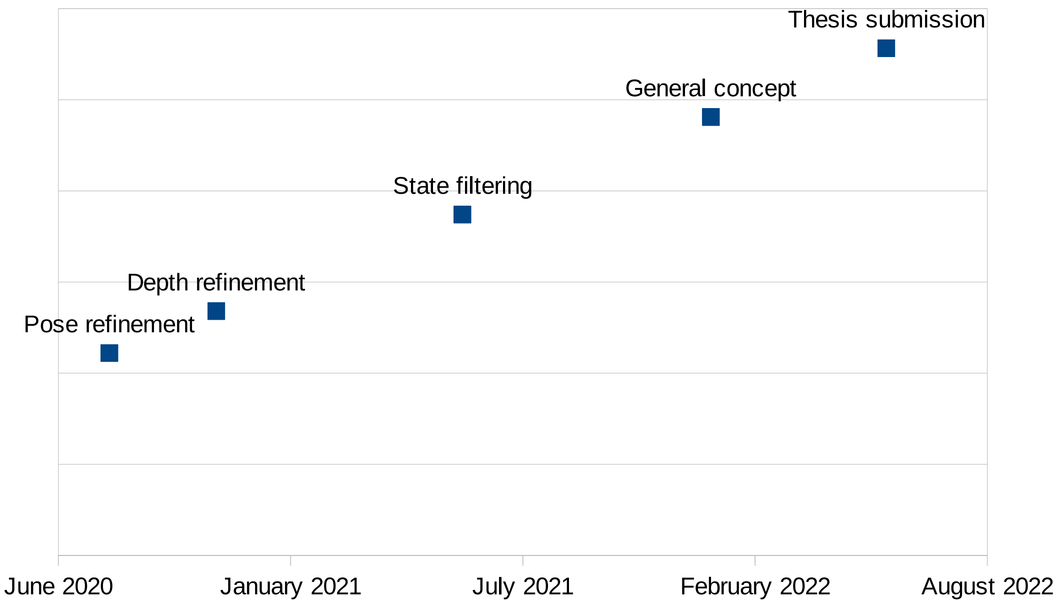


Figure 4.1: Proposed timeline

Chapter 5

Resource and Plans

5.1 Proposed Output and Dates

- Pose refinement work to WACV2021 and/or CVPR2021. Submission: 26 Aug 2020 and/or 9 Nov 2020
- Depth refinement work to CVPR2021/ICCV2021. Submission: 9 Nov 2020/ 1 Jan 2021
- State filtering work to ICRA2022. Submission $\sim$ 30 Jun 2021
- Overview of Innovation CNNs to ICCV2022. Submission: $\sim$1 Jan 2022
- Thesis submission: $\sim$ May-Jul 2022

5.2 Required Resources

5.2.1 Compute

I expect to continue to require access to the Quad-core GPU machine for the remainder of my program. I don't expect to regularly require the full capacity of this machine, though this would likely be helpful closer to submission dates.

5.2.2 Students

We currently have an honours student helping to begin the work on depth refinement. This student is likely to continue this work until November 2020. It may prove useful to engage a second honours student to assist with comparing optimisation algorithms, or with the state filtering work. However, a clearer outline of how this student might contribute is yet to be determined.

5.2.3 Finances

I hope to submit papers to the conferences listed above, and am therefore likely to require travel funding if these submissions are successful.

5.2.4 Collaboration & Assistance

When I am in need of assistance on a given problem, or if I am looking to collaborate with others, there are a range of people that I hope to turn to depending on the problem area. These people include, but are not limited to those who are a part of my supervision panel. A brief overview of those who I hope to be able to approach for these reasons is provided below.

For problems in systems theory I will turn to my primary supervisor; Rob Mahony. If for some reason this is not possible/necessary, I also hope to utilise the skills of Pieter Van Goor and/or Yon Ng in this area. For problems related to deep learning I will turn to my supervisors; Xin Yu, Nick Barnes, and/or Hongdong Li. If this is not possible/necessary, I hope to continue working with Zheyu Zhuang while he remains a PhD candidate.

I believe it would be highly beneficial for me to continue weekly, or bi-weekly meetings that include Xin Yu, beyond the duration of our current work in pose estimation, so that I can continue benefiting from his experience in deep learning. I also hope that I will be able to continue regular meetings with Zheyu while our work on pose estimation and depth estimation continues. I expect that I will also require some regular meetings with either Yon Ng or Pieter Van Goor when this project moves toward online state estimation problems.

5.2.5 National & International Travel

I am open to and interested in the possibility of spending extended time at another university. A possible option is to spend time at UTS collaborating with Xin Yu. However, I am unsure where else it might be beneficial for me to look at attending. At this stage I have no firm plans in this area.

Chapter 6

Appendix

Submitted Paper

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-30 (2019) 288–293

A Perception Pipeline for Robotic Harvesting of Green Asparagus

Gerard Kennedy * **Viorela Ila** ** **Robert Mahony** *

* *Australian National University, Canberra, Australia (e-mail: firstname.lastname@anu.edu.au).*
** *University of Sydney, Sydney Australia (e-mail: firstname.lastname@sydney.edu.au).*

Abstract: The global population is expected to pass 9 billion by 2050 requiring ongoing improvements in food production methods. Robotic harvesting offers part of the solution to this challenge and has spurred research in the use of agricultural robots (AgBots) for harvesting horticultural crops over the past several decades. While there has been significant progress in automation of harvest of many crops, robotic systems for crops that require selective harvesting remain far from mature. This paper presents the first steps toward a perception pipeline for a selective green asparagus harvesting robot. We show that a novel single-view representation of information from a multi-camera system can be combined with simple temporal filtering to reliably localise asparagus spears in real-time in lab experiments and difficult outdoor conditions.

Keywords: robotics, vision, automation, harvesting, agriculture

1. INTRODUCTION

An increasing global population and a changing climate are among several factors that are expected to apply pressure to the global food chain in coming decades. The United Nations has predicted that the global population will increase to over 9 billion by 2050, and that food production will increase by 70% in response (FAO (2009)). Furthermore, it has been observed that current agricultural practices have limited scalability due to factors including the lack of arable land, water shortages, under-investment, environmental considerations, and lack of available labour caused in part by an aging global population (ACRV (2018); Duckett et al. (2018)). In short, we face a challenge to produce more with less resources.

Agricultural robots (AgBots) have been identified as part of the solution to this impending global crisis. AgBots are expected to fit into a broader picture wherein crop breeding, planting, and harvesting practices are refined to allow greater levels of automation. Increased levels of robotics in agriculture is predicted to improve efficiency throughout the food production chain.

While mechanisation of broad field farming is highly developed and robotic automation of these processes is progressing as expected, progress in selective harvesting (only harvest a crop if criteria are met) robotic system has been slow. There are still relatively few examples of successful prototypes and almost no commercial examples of selective harvesting AgBots. The key reason for this is that most applications of selective harvesting AgBots require advanced sensor data processing to identify target crops for harvest amongst the unripe crops and surrounding plant matter. Achieving reliable identification of crops remains a challenge in difficult on-farm conditions where speed and accuracy are paramount and the environment is highly variable (ACRV (2018)). In order for robotic technology to be successfully integrated into existing farm operations and logistics they need to be able to operate at speeds and efficiencies comparable or superior to human labour. High speed harvest requires real-time (in the order of tens of milliseconds) perception and cognition as well as high speed actuators and robust mechanical designs.

In this paper we present a prototype perception pipeline for a green asparagus harvesting robot. Green asparagus grows irregularly in beds with the harvest period covering eight to ten weeks in Australia. Spears grow straight out of the ground with no obscuring foliage. The asparagus spears are harvested when they reach between 20 to 25 cm tall with taller asparagus to be removed and discarded. Since asparagus spears can grow up to 5cm per day, each row must be harvested each day during the peak growing period, and only those asparagus in the correct height range should be harvested. No commercial automated mechanical systems for asparagus harvesting exist and more than half the cost of producing asparagus is associated with the labour costs.

The proposed perception pipeline has been optimised for speed and accuracy through the use of multiple monocular cameras, a distributed compute architecture, and simple algorithms. We expect this approach to allow our harvester to reach an optional balance of speed and accuracy. The perception system is targeted to provide accurate information to a mechanical harvesting system (commercial in

⋆ G.Kennedy and R. Mahony, are with the Australian Centre for Robotic Vision, Australian National University, Australia. V. Ila is with the Australian Centre for Field Robotics, Sydney University, Australia. This research was supported in part by the Australian Government Research Training Program Scholarship and in part by the Australian Research Council through the "Australian Centre of Excellence for Robotic Vision" CE140100016.

confidence) capable of a cycle time of 300ms per asparagus spear. The target is for the robotic harvest system to travel at 1-2 m/s, roughly an order of magnitude faster than existing systems and comparable or superior to human labour. Asparagus spears will be visible to the perception system for 500-1000ms depending on the configuration of cameras. Results are presented that demonstrate the performance of the pipeline to varying crop layout in-lab, and difficult, on-farm conditions. The key contributions of this paper are:

- Algorithm architecture capable of calibrating a multi-camera system, and detecting and localising asparagus spears in outdoor conditions
- A novel single-image representation of a scene captured by a multi-camera system
- On-farm and in-lab trials of the perception pipeline

The remainder of this paper is structured as follows: Section 2 provides a brief history of robotic harvesters, with particular focus on robotic harvesters for green asparagus. Section 3 breaks down our proposed algorithm architecture. On-farm and in-lab experimental results are presented in Section 4, before concluding remarks in Section 5.

2. RELATED WORK

The first example of considering computer vision to crop detection involved utilising light reflectivity differences to detect citrus fruit (C Schertz (1968)). The first applied system was developed in 1977 for apple detection, using a black and white camera with red optical filter along with intensity thresholding and morphological operations (A. Parrish et al. (1977)). Early applications of computer vision to crop detection utilised mainly spectral, intensity or range cameras, and pixel-wise or shape-based crop detection methods (Jimnez et al. (2000)).

Robotic harvesters are currently being developed for a range of crops that require selective harvesting, with oranges, tomatoes, apples, and asparagus being among the most frequently targeted (Bac et al. (2014)). Recent robotic harvesters often employ sensing systems including single RGB (Lehnert et al. (2017)) or RGBD (Leu et al. (2017)) camera systems, combined with 3D reconstruction and 2D or 3D segmentation algorithms. A survey of robotic harvesters conducted in 2013 of 50 distinct projects over the previous 30 years found that average harvesting success rate was 66%, and average cycle time between harvests was 33 seconds (Bac et al. (2014)).

Previous asparagus harvesting robots have typically employed either LIDAR or single camera sensors. These sensing modalities have led to perception systems that only provide information about asparagus height (LIDAR) or else rely on computationally expensive algorithms such as dense 3D reconstruction. The first green asparagus harvester, CAMIA, was developed in the 1990's using laser sensors to detect asparagus height. While CAMIA had promising initial results, unreliability of the sensors caused many asparagus to be missed and damage caused to the bed (Arndt et al. (1997)). Similar to CAMIA, the Kim Haws harvester (Haws (2019)) and Geiger-Lund harvester (Geiger-Lund (2019)) currently being developed also employ laser sensors to detect crop height, and have also been noted to miss a significant proportion of ripe asparagus (Leu et al. (2017)). MANTIS (K. Carpenter (2019)), utilises a LIDAR sensor suite and can detect asparagus at a rate of 1 every 6 seconds (K. Carpenter (2019)). Finally, the GARotics harvester employs an RGBD camera and 3D reconstruction to harvest asparagus and reported a 90% harvest success rate of detected spears, although their detection success rate is not included (Leu et al. (2017)). At the time of writing, there are no examples of commercially successful selective harvesting robots for green asparagus. We believe that key reasons for this include poor detection success rates and slow harvest cycles. These insights form the motivation for this work.

3. ALGORITHM ARCHITECTURE

A system of three PointGrey cameras is used to capture images of asparagus at a frame rate of ∼55Hz. To keep lighting conditions consistent in the field our perception system will be housed within a darkened enclosure with its own internal lighting. This will enable day-time and night-time harvesting. The algorithm architecture is split into two components that operate in parallel. These components are titled Extrinsic Camera Calibration, and Perception Pipeline, and are discussed in Sections 3.1 and 3.2 respectively.

3.1 Extrinsic Camera Calibration

Extrinsic camera calibration is computed online as a separate process from the perception pipeline. It is computed at a low frequency and used to update camera pose and ground plane estimation periodically. This ensures that the cameras remain well calibrated throughout long periods of on farm use even in rough conditions. Extrinsic calibration involves computing a camera pose matrix $T \in \mathbf{SE}(3)$ for each camera. The calibration is implemented in the open-source OpenMVG framework via a Structure from Motion (SfM) algorithm computed from sparse feature points that are matched between frames (Moulon et al. (2017)) [1].

Given that the vast majority of matched feature points occur on the ground plane, this plane can be found via simple 3D plane fitting. The equation for the ground plane is defined as

$$ax + by + c = z$$

for 3D points $(x, y, z) \in \mathbb{R}^3$. The constants a, b, c are solved for using

$$\begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ & \vdots & \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_n \end{bmatrix},$$

which can be represented as an over determined linear system $Ax = B$ and solved via the left pseudo inverse

$$x = (A^T A)^{-1} A^T B.$$

This ground plane provides a convenient 2D representation of the region of interest. The output of the extrinsic calibration is a matrix T representing the pose of each camera, and a ground plane G, all with respect to a common inertial frame.

[1] Code available at: https://github.com/kennege/openMVG/tree/develop_multi_camera_calibration

3.2 Perception Pipeline

The perception pipeline incorporates the software for receiving images continuously from the multi-camera system, and information from the extrinsic calibration, and returning locations of asparagus to be harvested. Broadly, the procedures involved in the pipeline for each time step are: generate a 2D probability map for each image, project these maps to a common ground plane computed from the extrinsic calibration, and detect potential asparagus 'hypothesis' on this ground plane. Finally, the hypothesis are filtered temporally across time steps to improve accuracy. The four **procedures** are discussed further in the following four subsections respectively.

Algorithm 1 Perception Pipeline

```
1:  Input images from time $t = 0 : end$
2:  Initialise Kalman Filter
3:  for ($\mathbf{t} = 0 : end$) do // for images each time step
4:      for ($\mathbf{I}_{i=0:2}$) do // for images this timestep
5:          procedure 1. 2D PROBABILITY MAP($\mathbf{I}_i$)
6:              $\mathbf{I}_i$ = yCBCR($\mathbf{I}_i$) // colour space transform
7:              $\mathbf{I}_i = \gamma(\mathbf{I}_i)$ // perform Gamma correction
8:              $\mathbf{p}_{(0,1,2),i}$ = PCA($\mathbf{I}_i$)
9:          procedure 2. GROUND PROJECTION($\mathbf{p}_{1,i}$)
10:             $\mathbf{P}_i = \mathbf{K}_i \mathbf{T}_G \mathbf{T}_i^{-1}$ // projection matrix
11:             $\mathbf{H}_i = \mathbf{P}_i$[:][0,1,3]
12:             $\mathbf{G}_i$ = warp($\mathbf{p}_{1,i}$,$\mathbf{H}_i$) // backward projection
13:     procedure 3. FIND HYPOTHESIS($\mathbf{G}_{i=0:2}$)
14:         $\mathbf{G} = \sum_{i=0}^{2} \mathbf{G}_i$ // combine images
15:         $\mathbf{G}$ = thresh($\mathbf{G}$)
16:         $\mathbf{G}$ = removeClutter($\mathbf{G}$)
17:         $\mathbf{L}_{1:n}$ = fitLines($\mathbf{G}$) // fit lines 0:$n$
18:         $\mu_{1:m}, \Sigma_{1:m}$ = hypothesis($\mathbf{L}_{i=1:n}$,$\mathbf{L}_{j=1:n}$,$\mathbf{L}_{k=1:n}$)
19:     procedure 4. FILTER HYPOTHESIS($\mu_{1:m}, \Sigma_{1:m}$)
20:         $\mu_{1:m}, \Sigma_{1:m}$ = KalmanFilter($\mu_{1:m}, \Sigma_{1:m}$)
21:         for $h$=0:m do // for each hypothesis $h$
22:             if ($m_h$ > thresh & 20cm < $l_h$ < 25cm) then
23:                 Mark $h$ for harvesting
```

2D Probability Map: For each timestep images from each camera are first processed to produce a representation of probability of asparagus. Our approach involves colour space conversion from RGB to yCBCR, gamma correction, and Principle Component Analysis (PCA). The yCBCR colour space has been chosen empirically based on the relative contrast of plant matter to background. Gamma correction was computed with $\gamma = 1.5$. The first three principal axis are generated, and the second axis is used as the output probability map.

Ground Plane Projection: The known camera poses and ground plane provided by the extrinsic calibration are used to project each image plane I to the common ground plane, labeled G (see Figure 2). This is done by first computing the camera projection matrix $P \in \mathbb{R}^{3\times4}$ using

$$P = KT_GT^{-1},$$

where K is the intrinsic camera matrix computed offline via the Matlab Camera Calibration app (Matlab (2019)). The camera pose $T \in \mathbf{SE}(3)$ is obtained from the extrinsic calibration, and $T_G \in \mathbf{SE}(3)$ is chosen to transform the

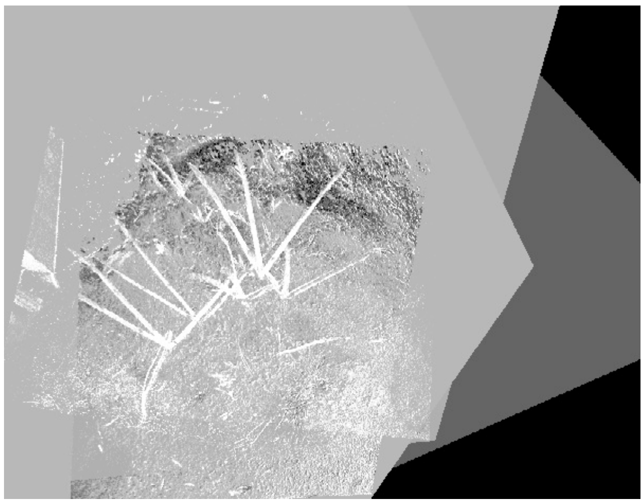


Fig. 1. Ground Plane image obtained by projecting the 2D probability map from each camera view to a common ground plane.

inertial frame of the extrinsic calibration such that it is aligned with the z-axis orthogonal to G. Let $P = [p_1, p_2, p_3, p_4]$ with each $p_i \in \mathbb{R}^3$ representing a column of P. The first three columns p_1, p_2, p_3 represent the vanishing points in G with respect to the x, y, and z axis respectively, in homogeneous image coordinates. As we have defined T_G such that the z-axis of the new inertial frame is orthogonal to the ground plane then $p_3 \approx (0, 0, 1)^\top$, where $\approx$ denotes equality up to scale. Set $H = [p_1, p_2, p_4] \in \mathbb{R}^{3\times3}$. This is an homography between G and I. Points in each image frame I can thus be mapped to G using a backward homography projection,

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \approx H \begin{bmatrix} x_g \\ y_g \\ 1 \end{bmatrix}$$

for $i \in I$ and $g \in G$.

An example ground plane image is provided in Figure 1. This image is created using the scale 1 pixel = 1mm. In this figure, each asparagus is represented by a 'chicken-foot' object corresponding to the same asparagus projected from three different views. This is a novel method of encoding multi-camera information in a single image plane. We believe that the extra information encoded in this image (three views of each object in one image) has the potential to allow machine learning algorithms to extract high quality estimates of object location and pose, compared to typical single image-based algorithms that perform estimates from single RGB images. While the goal is to eventually apply such algorithms for 3D localisation, this step is currently implemented via the algorithm outlined below.

Hypothesis Generation: The ground plane image is thresholded and morphological operations (closing with the OpenCV line structuring element (Bradski (2000))) are applied to remove 'clutter'. Lines are fit to each remaining morphological 'blob' via a least-squares approach. Area A, centroid c, and eccentricity e are also computed for each blob. All line intersections are computed, and a hypothesis of an asparagus 2D location within the ground plane is represented by every set of three intersections.

The hypothesis mean μ_h is computed as the centroid of the triangle spanned by the three intersections,

$$\mu_h = \frac{1}{3}(a+b+c) \tag{1}$$

where $a, b, c \in \mathbb{R}^2$ represent the three intersections of the given hypothesis. The hypothesis covariance Σ_h is represented by the Steiner circumellipse of the triangle (Wolfram (2019)) and the score of each triangle edge, via

$$\Sigma_h = \begin{bmatrix} \frac{e}{3}\sum_{i=0}^{2} S_i & 0 \\ 0 & \frac{f}{3}\sum_{i=0}^{2} S_i \end{bmatrix} \tag{2}$$

where e and f refer to the major and minor axis of the Steiner circumellipse respectively and S_i refers to a score of triangle side i. We propose a likelihood score S to be

$$S = \left|1 - \frac{e + \frac{A}{\mu_A}}{2}\right|,$$

where A and e are the area and eccentricity of the current asparagus 'blob' respectively and μ_A is found empirically to be the area of a typical asparagus 'blob' in pixels. This score is designed to provide a normalised measure of how well the eccentricity and area of the detected blob matches a typical asparagus. A hypothesis is added to the system state if it satisfies the condition

$$\left(\frac{1}{3}\sum_{i=0}^{2} S_i < 0.6\right) \wedge \Big(t < 1\Big) \wedge \Big(\mu_h \in I_b\Big) \wedge \Big(||\Sigma_h|| < 1\Big),$$

where $t = |\frac{\bar{l}-\mu_l}{\sigma_l/\sqrt{3}}|$, $||x||_2$ is the L2-norm of x, and

$$\bar{l} = \sum_{i=0}^{2} 2||\mu_h - c_i||_2 \tag{3}$$

is the average length of the three asparagus 'blobs' in the hypothesis (in pixels), μ_l and σ_l are found empirically to be the mean and standard deviation in asparagus 'blob' length, c is the centroid of the current asparagus 'blob', and $x \in I_b$ indicates that x lies within the ground plane image, on an asparagus 'blob'. This condition was chosen empirically based on a range of tests run on farm and in-lab data. Asparagus hypothesis height in meters ($l \in \mathbb{R}$) can be obtained using Equations (1) and (3) and

$$l = \frac{1}{3}\sum_{i=0}^{2} \frac{\frac{l_i}{1000} z_i}{\sqrt{\frac{\mu_h^2}{1000}} + \sqrt{x_i^2 + y_i^2} + \frac{l_i}{1000}} \tag{4}$$

where x_i, y_i, z_i represent the position of camera i with respect to G, and the scale factor of $\frac{1}{1000}$ is used due to the imposed scale of 1mm = 1 pixel in the ground plane image.

Using this formulation, the probability encoded in the 2D map is propagated to morphological blobs, then to lines via the line scores S, then to 2D location hypothesis via μ_h and Σ_h. Finally, these hypothesis are filtered temporally with a Kalman Filter.

3.3 Kalman Filtering

A 2D Kalman Filter (KF) is used to track hypothesis temporally through successive ground plane images. Given our expected harvester speed we expect each asparagus spear to be in view for 30-50 frames. Temporal filtering allows hypothesis to be refined and pruned over this period.

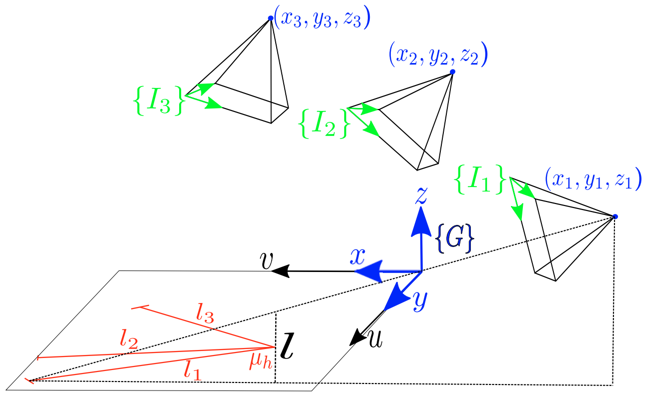


Fig. 2. Relationship between image frames I_i and ground plane G. Where l_i is the 2D length of the asparagus in G projected from image plane I_i, and μ_h is the 2D position estimate and l is the height estimate.

The KF is applied to the following process model:

$$X_k = X_{k-1} + U_k + W_k$$

with observations of the form:

$$Z_k = X_k + V_k$$

where

$$X_k = \begin{pmatrix} x_k \\ y_k \\ l_k \end{pmatrix}, \; U_k = \begin{pmatrix} u_k \\ v_k \\ 0 \end{pmatrix},$$

$$W_k \sim \mathcal{N}(0, Q_k), \; V_k \sim \mathcal{N}(0, R_k),$$

$$Q_k = \begin{bmatrix} \Sigma_{u_k} & 0 & 0 \\ 0 & \Sigma_{v_k} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; R_k = \begin{bmatrix} \Sigma_{h,k} & 0 \\ 0 & \Sigma_{l,k} \end{bmatrix}$$

and k denotes timestep, x, y represent the 2D location of an hypothesis in ground plane coordinates, with associated measurements provided by μ_h from Equation (1), l represents the height of the asparagus hypothesis obtained in pixels using Equation (3) and converted to meters using Equation (4). The covariance Σ_h is found using Equation (2), and Σ_l represents the covariance of the height estimate. The covariances Q_k and R_k represent process and measurement noise respectively. Data association of asparagus hypothesis between frames is obtained via Lucas-Kanade Optical Flow (LKOF). Estimates for u_k, v_k, Σ_{u_k} and Σ_{v_k} are obtained empirically from the pixel-wise displacement obtained from the LKOF.

An asparagus hypothesis is considered to be valid if it is tracked for a set number of frames (referred to as 'track length' and labeled m_h in Algorithm 1). If a hypothesis is validated and its height meets the threshold condition it is marked for cutting. The 3D cutting point is fixed at 2cm above the ground plane.

4. EXPERIMENTS

4.1 On-farm

Image data has been captured at a nearby asparagus farm on two occasions (Figure 3). This has allowed for testing

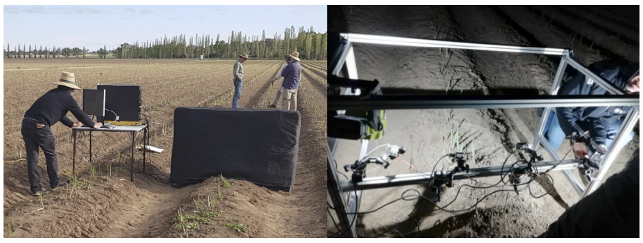


Fig. 3. On-farm day and night experiments.

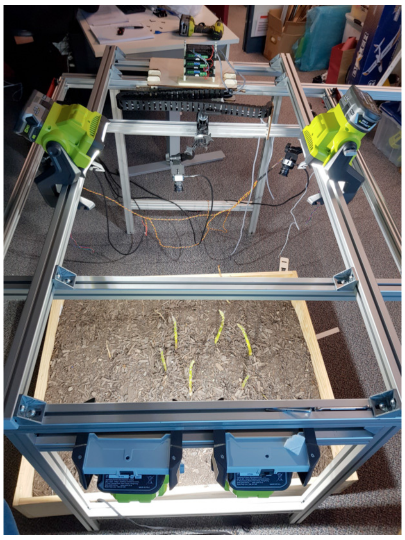


Fig. 4. In-lab rig for harvesting simulation. A sliding bar attached to a motor system allows the camera system to move over an asparagus bed at the expected speed of the harvester.

of the performance of the physical perception system, and robustness of the perception pipeline to outdoor conditions.

Figure 5 presents an example result of hypothesis generation and filtering using on-farm data. While initial hypothesis detections are noisy, the KF reliably filters erroneous detections via an upper limit on the state covariance. Initial small covariance values are due to the size of the Steiner circumellipse. Covariances grow due to process noise, and when an hypothesis is not tracked between frames, and covariances decrease when a data assocation is made. During harvesting, an hypothesis will be considered valid if it has a certain track length. The on-farm dataset contains 30 frames, and the required track length was set to $m_h = 3$. Using this track length, the three hypothesis shown in Figure 5(h) were validated. At this point the asparagus height estimate will be obtained from the KF and harvested if it is tall enough. The 3 tall spears met the target track length ($m_h = 3$) after 8 frames and were marked for harvesting. The 2 smaller targets and a horizontal spear are not detected, but these would not meet the harvest threshold. No other locations surpassed the target track length for the remainder of the dataset (30 frames).

4.2 In-lab

Datasets simulating the perception system moving over an asparagus bed have been captured in-lab via the fabricated rig shown in Figure 4. A sliding bar attached to a motor system allows the camera system to move over an asparagus bed at the expected speed of the harvester. Using this rig, the results shown in Table 1 were generated.

Table 1 contains the results of six experiments, each with a different configuration of asparagus spears. Example configurations are shown in Figure 6. A dense 3D reconstruction computed in OpenMVG is used as a pseudo ground truth in this experiment. RMSE (mm) is the root mean squared error in the KF estimate, ξ_k for a given hypothesis, taken with respect to the estimate provided by the 3D reconstruction, ζ_k. That is

$$\text{RMSE} = \sqrt{1/N \sum (\xi_k - \zeta_k)^2}.$$

Exp	**RMSE (mm)**
1	0.137
2	1.829
3	0.128
4	1.778
5	0.207
6	0.333
Overall	**0.735**

Table 1. Results of In-lab Experiments. **RMSE (mm)** is the root mean squared error in the KF estimate.

Each dataset in Experiments 1-6 comprises of $\sim$ 30 time steps, and the required track length is once again set to $m_h = 3$. No horizontal spears were detected during the experiments. This is due to the ground plane projection, from which horizontal spears do not produce the ‘chicken-foot’ representation of vertical asparagus. This is advantageous as only upright spears are targeted for harvest. This is experiment indicates that the proposed pipeline finds asparagus cutting points robustly with a high level of accuracy.

5. CONCLUSION

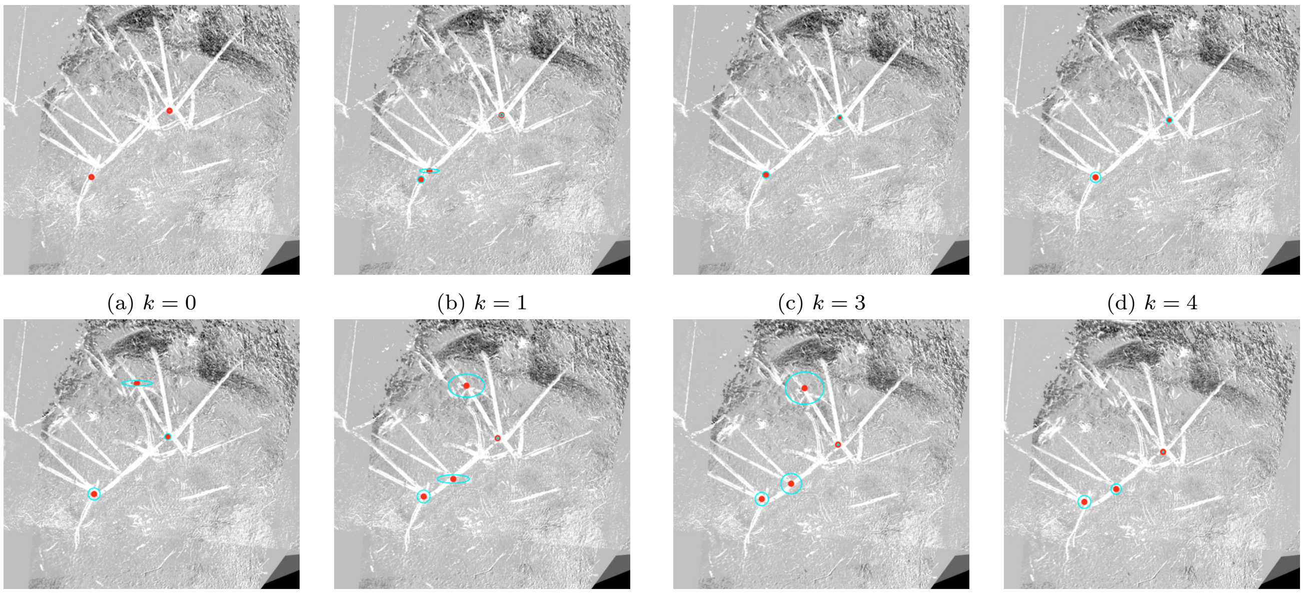
In this paper we have presented a perception pipeline for robotic harvesting of green asparagus. The pipeline utilises simple algorithms and a multi-camera rig to obtain estimates of asparagus location in real-world conditions. A novel single-view ground plane projection is presented as a novel approach to condensing multi-camera information into a single image. Results indicate that the pipeline produces accurate location estimates, and performs well in on-farm conditions.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Alex Martin and Dr. Tao Hu for their help in developing the presented hardware and software, and in the undertaking of experiments.

REFERENCES

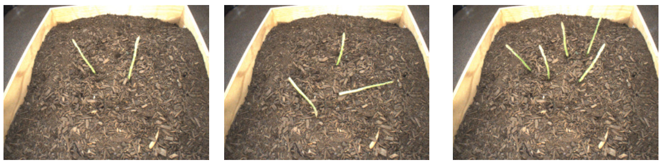
A. Parrish, E., , J., and K. Goksel, A. (1977). Pictorial pattern recognition applied to fruit harvesting. *Transactions of the ASAE*, 20, 0822–0827. doi:10.13031/2013.35657.



(a) $k = 0$ (b) $k = 1$ (c) $k = 3$ (d) $k = 4$

(e) $k = 5$ (f) $k = 6$ (g) $k = 7$ (h) $k = 8$

Fig. 5. Kalman filtering of hypothesis using the first 6 frames of on-farm data. The 3 tall spears met the target track length ($m_h = 3$) after 8 frames and were marked for harvesting. The 2 smaller targets are not detected, but these would not meet the harvest threshold. A horizontal spear is also not detected. No other locations surpassed the target track length for the remainder of the dataset (30 frames). Covariance ellipses increase between frames if the hypothesis is not detected, and decrease if the hypothesis is detected.



(a) Exp **2** (b) Exp **3** (c) Exp **6**

Fig. 6. In-lab Experiments. Only upright asparagus should be targeted for harvesting.

ACRV (2018). A robotics roadmap for australia. `https://www.roboticvision.org/wp-content/uploads/Robotics-Roadmap_FULL-DOCUMENT.pdf`.

Arndt, G., Rudziejewski, R., and A. Stewart, V. (1997). On the future of automated selective asparagus harvesting technology. *Computers and Electronics in Agriculture - COMPUT ELECTRON AGRIC*, 16, 137–145. doi:10.1016/S0168-1699(96)00033-6.

Bac, C.W., van Henten, E.J., Hemming, J., and Edan, Y. (2014). Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6), 888–911. doi:10.1002/rob.21525. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21525`.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools.*

C Schertz, G.B. (1968). Basic considerations in mechanizing citrus harvesting. *Transactions of the ASAE*, 343–346.

Duckett, T., Pearson, S., Blackmore, S., and Grieve, B. (2018). Agricultural robotics: The future of robotic agriculture. *CoRR*, abs/1806.06762. URL `http://arxiv.org/abs/1806.06762`.

FAO (2009). Global agriculture towards 2050. `http://www.fao.org/fileadmin/templates/wsfs/docs/Issues\_papers/HLEF2050\_Global\_Agriculture.pdf`. Accessed: 2019-05-23.

Geiger-Lund (2019). Geiger-lund selective asparagus harvesters. `http://www.asparagusharvester.com/Drawing-Model-H-24.html`. Accessed: 2019-05-23.

Haws, K. (2019). Haws harvester. `http://hawsharvester.com/`. Accessed: 2019-05-23.

Jimnez, A., Ceres, R., and Pons, J. (2000). A survey of computer vision methods for locating fruit on trees. *Trans. ASAE*, 43. doi:10.13031/2013.3096.

K. Carpenter, E. Glasgow, T.R.D.S. (2019). Mantis robotic asparagus harvester. `https://sites.google.com/site/mrsdproject201213teamc/home`. Accessed: 2019-05-23.

Lehnert, C., English, A., McCool, C., Tow, A.W., and Perez, T. (2017). Autonomous sweet pepper harvesting for protected cropping systems. *IEEE Robotics and Automation Letters*, 2(2), 872–879. doi:10.1109/LRA.2017.2655622.

Leu, A., Razavi, M., Langstdtler, L., Risti-Durrant, D., Raffel, H., Schenck, C., Grser, A., and Kuhfuss, B. (2017). Robotic green asparagus selective harvesting. *IEEE/ASME Transactions on Mechatronics*, 22(6), 2401–2410. doi:10.1109/TMECH.2017.2735861.

Matlab (2019). Single camera calibration app. `https://au.mathworks.com/help/vision/ug/single-camera-calibrator-app.html`.

Moulon, P., Monasse, P., Marlet, R., and Others (2017). Openmvg. an open multiple view geometry library. `https://github.com/openMVG/openMVG`.

Wolfram (2019). Steiner circumellipse. `http://mathworld.wolfram.com/SteinerCircumellipse.html`.

References

1. A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–13, 02 2018.

2. R. K. Sinha, R. Pandey, and R. Pattnaik, "Deep learning for computer vision tasks: A review," *CoRR*, vol. abs/1804.03928, 2018. [Online]. Available: http://arxiv.org/abs/1804.03928

3. Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

4. R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

5. K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

6. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

7. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

8. D. Yoo, S. Park, J. Lee, A. S. Paek, and I. S. Kweon, "Attentionnet: Aggregating weak directions for accurate object detection," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2659–2667.

9. J. H. Lee, M. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *ArXiv*, vol. abs/1907.10326, 2019.

10. D. Wofk, F. Ma, T. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6101–6108.

11. H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep Ordinal Regression Network for Monocular Depth Estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

12. R. Garg, B. V. Kumar, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *European Conference on Computer Vision*. Springer, 2016, pp. 740–756.

13. C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, 2017.

14. Z. Fu, Y. Zheng, H. Ye, Y. Kong, J. Yang, and L. He, "Edge-aware deep image deblurring," *CoRR*, vol. abs/1907.02282, 2019. [Online]. Available: http://arxiv.org/abs/1907.02282

15. W. Yang, X. Zhang, Y. Tian, W. Wang, and J. Xue, "Deep learning for single image super-resolution: A brief review," *CoRR*, vol. abs/1808.03344, 2018. [Online]. Available: http://arxiv.org/abs/1808.03344

16. J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 341–349. [Online]. Available: http://papers.nips.cc/paper/4686-image-denoising-and-inpainting-with-deep-neural-networks.pdf

17. A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2938–2946.

18. Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," 2017.

19. S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 494–495.

20. M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," 2019.

21. W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1530–1538, 2017.

22. Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," *International Journal of Computer Vision*, vol. 128, no. 3, p. 657–678, Nov 2019. [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01250-9

23. B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," *CoRR*, vol. abs/1711.08848, 2017. [Online]. Available: http://arxiv.org/abs/1711.08848

24. M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3848–3856, 2017.

25. S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *CVPR*, 2019.

26. C. Song, J. Song, and Q. Huang, "Hybridpose: 6d object pose estimation under hybrid representations," 2020.

27. S. Zakharov, I. S. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1941–1950, 2019.

28. S. Hinterstoisser, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes."

29. E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *ECCV*. Springer, September 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/learning-6d-object-pose-estimation-using-3d-object-coordinates/

30. A. Aristidou and J. Lasenby, "Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver," 2009.

31. J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

32. J. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. Sagastizabal, *Numerical Optimization – Theoretical and Practical Aspects*, 01 2006.

33. S. Phaniteja, P. Dewangan, P. Guhan, A. Sarkar, and K. M. Krishna, "A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2017, pp. 1818–1823.

34. O. Wiles and A. Zisserman, "Silnet : Single- and multi-view reconstruction by learning from silhouettes," *CoRR*, vol. abs/1711.07888, 2017. [Online]. Available: http://arxiv.org/abs/1711.07888

35. F. Manhardt, W. Kehl, N. Navab, and F. Tombari, "Deep model-based 6d pose refinement in RGB," *CoRR*, vol. abs/1810.03065, 2018. [Online]. Available: http://arxiv.org/abs/1810.03065

36. S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

37. D. Morrison, J. Leitner, and P. Corke, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," in *RSS*, 06 2018.

38. Z. Zhuang, J. Leitner, and R. Mahony, "Learning real-time closed loop robotic reaching from monocular vision by exploiting a control lyapunov function structure," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 4752–4759.

39. D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

40. C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 15–22.

41. J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

42. S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2043–2050.

43. K. Konda and R. Memisevic, "Learning visual odometry with a convolutional network," vol. 1, 03 2015.

44. B. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 08 1981.

45. J. Hur and S. Roth, "Iterative residual refinement for joint optical flow and occlusion estimation," *CoRR*, vol. abs/1904.05290, 2019. [Online]. Available: http://arxiv.org/abs/1904.05290

46. Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," 2020.

47. M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *Robotics and Automation, IEEE Transactions on*, vol. 17, pp. 229 – 241, 07 2001.

48. R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

49. R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, p. 1147–1163, Oct 2015. [Online]. Available: http://dx.doi.org/10.1109/TRO.2015.2463671

50. R. Mahony and T. Hamel, "A geometric nonlinear observer for simultaneous localisation and mapping," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 2408–2415.

51. P. van Goor, R. E. Mahony, T. Hamel, and J. Trumpf, "An equivariant observer design for visual localisation and mapping," *CoRR*, vol. abs/1904.02452, 2019. [Online]. Available: http://arxiv.org/abs/1904.02452

52. R. Kang, J. Shi, X. Li, Y. Liu, and X. Liu, "DF-SLAM: A deep-learning enhanced visual SLAM system based on deep local features," *CoRR*, vol. abs/1901.07223, 2019. [Online]. Available: http://arxiv.org/abs/1901.07223

53. S. Milz, G. Arbeiter, C. Witt, B. Abdallah, and S. Yogamani, "Visual slam for automated driving: Exploring the applications of deep learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

54. H. Zhou, B. Ummenhofer, and T. Brox, "Deeptam: Deep tracking and mapping," *CoRR*, vol. abs/1808.01900, 2018. [Online]. Available: http://arxiv.org/abs/1808.01900

55. M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," *CoRR*, vol. abs/1606.04474, 2016. [Online]. Available: http://arxiv.org/abs/1606.04474

56. J. Adler and O. Öktem, "Solving ill-posed inverse problems using iterative deep neural networks," *Inverse Problems*, vol. 33, no. 12, p. 124007, Nov 2017. [Online]. Available: http://dx.doi.org/10.1088/1361-6420/aa9581

57. P. Putzky and M. Welling, "Recurrent inference machines for solving inverse problems," *CoRR*, vol. abs/1706.04008, 2017. [Online]. Available: http://arxiv.org/abs/1706.04008

58. G. Kennedy, V. Ila, and R. Mahony, "A perception pipeline for robotic harvesting of green asparagus," in *6th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2019*, vol. 52, no. 30, 2019, pp. 288 – 293, iFAC-PapersOnLine. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896319324553

59. R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

60. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

61. C. Lee, S. Jung, K. Kim, and G. G. Lee, "Hybrid approach to robust dialog management using agenda and dialog examples," *Computer Speech & Language*, vol. 24, no. 4, pp. 609 – 631, 2010.

62. R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.