3D HAND TRACKING FROM RGB SEQUENCES

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2020

Peter Thompson

School of Engineering
Department of Computer Science

Contents

Word Count: 18414

List of Tables

List of Figures

List of Algorithms

Abstract

3D HAND TRACKING
FROM RGB SEQUENCES
Peter Thompson
A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy, 2020

Vision-based hand tracking has been an active field of research since the early 1990s. Early attempts tended to use generative kinematic models, in which a hand state proposal is quantitatively evaluated according to features extracted from an input image. The approximate hand pose is then found by optimising the pose of the kinematic model according to those features. This paradigm continued to be used in both RGB-based tracking and in later depth-based tracking approaches. More recent attempts have made use of convolutional neural networks (CNN) to predict keypoint locations discriminatively. Here, a contemporary CNN approach is applied to the conventional generative hand tracking paradigm. This is done by using CNNs to semantically segment each frame of an RGB sequence containing hand gestures before using a generative kinematic model to find the optimal hand pose for each frame given the semantic segmentation result.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to thank my parents, Ann and Paul,

and my supervisor, Aphrodite.

Chapter 1

Introduction

Hand tracking has been an active field of research for several decades due to the potential advantages of applying hand tracking systems in human-computer interaction. Hand tracking can be used to create intuitive ways for people to interact with computers without the need for physical contact. This is particularly advantageous in contexts where physical contact is impossible or undesirable, such as sterile environments or public terminals. Hand tracking may also be used in conjunction with other techniques to collect data about how people behave and interact in various situations, which could inform the design of both commercial products and of public spaces.

The goal of hand tracking is to recover the pose of a hand from sensor data. In vision-based tracking, which has been an active field of research since the early 1990s, the sensor is a standard RGB camera. In the late 2000s, depth camera input also became a staple of the field. Depth cameras brought a significant improvement in the accuracy of hand tracking systems, since they provide information about the 3D position of the hand as opposed to an ambiguous projection into 2D that RGB cameras provide. There has, however, in recent years been a trend of moving away from depth camera input and back to RGB. This trend is motivated by the fact that RGB cameras are cheaper, virtually ubiquitous, and typically double the resolution of equivalent depth cameras, as well as not suffering from the range limitations and distortion artefacts common to depth cameras. This allows RGB-based algorithms to be deployed in a much wider variety of contexts than depth-based algorithms. The trend of moving back to RGB is mostly facilitated by deep learning, which allows features to be learned from large quantities of training data.

The primary focus of this thesis is RGB-based hand tracking techniques. Depth-based hand tracking will also be discussed as they had a significant impact on the field and must be discussed in order to contextualise properly contemporary techniques. Hand tracking systems that rely on sensor gloves or visual markers will not be discussed since such techniques represent a very different set technical considerations and potential applications. In addition to a discussion of the history of the field, the different techniques that have been applied, and recent advances and their limitations, a novel technique for hand tracking from RGB is introduced and evaluated in a variety of ways.

The proposed method for hand tracking is partially inspired by the recent resurgence of interest in RGB-based hand tracking and makes use of convolutional neural networks (CNNs)[LBD+90], which are currently ubiquitous in the field[ZB17, POA18, MBS+18, IMBJGK18, CGCY18, DMB+18, ROL18, NOTA18, YY19, BKK19]. It is also inspired by the more conventional generative, model-based approaches to hand tracking and attempts to capture the advantages of such approaches. One such advantage is the ability to take an possible hand pose and evaluate it's likelihood given the state of the input, which allows arbitrary constraints to put on the output of the algorithm. These constraints often reflect the physiological constraints on the pose of the hand and can guarantee that the algorithm produces a physiologically plausible result. Another related advantage is the natural way in which model-based approaches can put limitations on the variation of the pose over time, thus allowing them to be improved through the observation of temporal consistency.

The fundamental limitation of generative RGB-based hand tracking stems from the fact that the appearance of the RGB image is mostly determined by the lighting conditions, the object in the frame besides the hand, and the reflective properties of the hand surface, and is thus grossly underdetermined by the pose of the hand. This necessitates some form of feature extraction be performed on the input RGB before the generative model can be applied. The simple line and region-based features common in early systems were severely limited in that they did not specifically resolve the ambiguity in mapping the pose to the image. The reason for this is the fact that, while the presence of a hand can reasonably be expected to have an impact on the appearance of these features, the reverse is not true, as these features will exist in any image, regardless of whether or not a hand is present. To overcome this limitation, CNNs are used to semantically segment the input images. The result of doing so is a semantic map containing the estimated likelihood that each pixel contains each part of the hand. This resolves the ambiguity in the input RGB images and provides information that is precisely relevant to the pose of the hand.

The goal of the research was to create a pipeline for RGB-based hand tracking that incorporated contemporary deep learning methods into the generative hand tracking paradigm by using CNNs to perform semantic segmentation on input hand images and a generative, model-based tracking algorithm to recover the state of the hand for each frame in a sequence. This was done by rendering a mesh model of the hand as an image of labels. This image of labels was compared with the output of the network in order to calculate a cost. This cost was then minimised by Differential Evolution (DE) to find the estimated pose for each frame.

The semantic segmentation part of the pipeline works well, with the semantic predictions being generally accurate and self-consistent, in that the CNN is capable of learning the composition of the hand parts implicitly from examples, with particular labels being confined to local regions of the image. Within these regions, the relevant label dominates (i.e. they are free from “noise”) and adjacent regions generally correspond to physically plausible hand poses. The tracking algorithm also performs well when tested in a variety of ways, ranging from a simple joint location matching problem to operating as part of the pipeline as described. When tracking from an ideal semantic input, the tracking algorithm performs comparably to the contemporary state-of-the-art RGB hand tracking methods.

The content of the subsequent chapters is as follows. **Chapter 2** gives the technical background of the field of hand tracking, with particular attention paid to those techniques that are relevant to contemporary RGB-based algorithms, and a review of important contributions to the literature in the field. **Chapter 3** introduces a novel approach to hand tracking and describes its operation in detail. **Chapter 4** reports the results of a number of experiments on this novel hand tracking system. **Chapter 5** summarises the contributions of this thesis and discusses potential avenues for future work.

Chapter 2

Background

Hand tracking algorithms can be broadly categorised as being either generative or discriminative. In generative tracking, a model of the hand is placed in a candidate pose and used to predict features of the input data. The discovered pose that is able to predict these features best is selected as the result. This approach may also be referred to as model-based or analysis-by-synthesis. This approach requires a model of the hand that can be put in an arbitrary pose and used to generate corresponding information to that which is extracted from the input. In discriminative tracking, the pose is inferred directly from features in the data, either by an expert system, comparison with predefined pose-feature pairs, or a learned model of some kind. Regression does not necessarily require a hand model, though sometimes one is used to ensure that a physically plausible hand pose is recovered. Aspects of these two broad approaches can also be combined in various ways to create hybrid systems.

The following sections will describe various algorithms, techniques, and approaches that have been relevant to the field of hand tracking, with particular emphasis placed on those that are relevant to the proposed method described in the next chapter. Section 2.5 will then summarise the history of the field and place the proposed method in context.

2.1 Modelling the hand

The natural articulations of the human hand are a result of its skeletal composition. There are three classes of bones in the hand: carpals, metacarpals, and phalanxes. The carpals are the small bones near the wrist, metacarpals the long bones in the palm, and phalanxes the bones in the fingers. These bones are connected by joints, which are referred to according to the bones they connect (e.g. carpal-metacarpal, distal inter-phalangeal, etc.) and can articulate in three ways. The first is medial/lateral rotation, when the bone rotates around its long axis. The other two are flexion/extension and abduction/adduction, which both refer to rotations about a short axis, with the former applied to movements that tend to make the skeleton ”curl up” (i.e. those associated with a grasping motion in the case of hands) and the latter applied to movements that make the skeleton ”spread out”, as well as the respective reverse motions.

In model based tracking, the principle way in which the articulations the hand are described is through kinematic modelling. A kinematic model, is a network of components connected in a hierarchy such that any transformations applied to any component in the hierarchy will affect all of the components beneath it. Mathematically, each component is represented by a relative transformation, consisting of a translation and set of intrinsic rotations. The product of the relative transformations from the root to a particular component is the total transformation applied to the component. The different kinematic states (i.e. poses) of the model are achieved by altering the relative transformations of the components.

In hand tracking, these components represent the bones in the skeleton, and the transformation represents the state of the joint. The palm is generally taken as the root of the hierarchy and rotates around its centre or wrist joint. The palm is typically assumed to be rigid, with the flexion/extension of the fourth and fifth carpal-metacarpal joint ignored. The metacarpal in the thumb is typically assumed to have two degrees

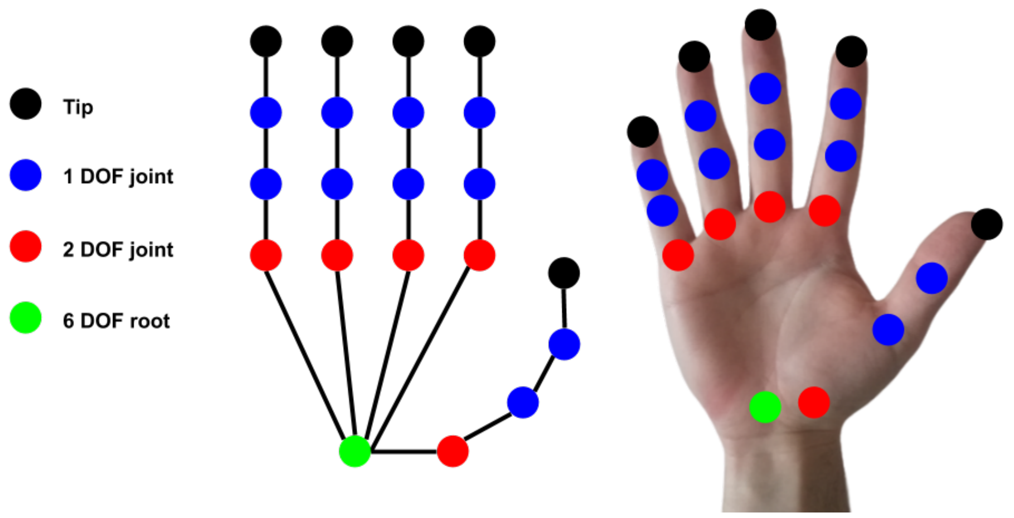

Figure 2.1: Schematic diagram of the conventional 26 degree-of-freedom hand model. An image of a real hand with the joints marked is also shown for clarity.

of freedom, though in reality it rotates slightly about its axis. The other joints in the hand are typically represented accurately, with the metacarpal-phalaxial joints having two degree-of-freedom and the interphalanxial joints having one. The translations are fixed, as the bones in a hand do not change size on the time scale of hand tracking. The result of these approximations is the conventional 26 degree-of-freedom hand model. Figure 2.1 shows the simplified kinematics of this model of the hand.

The joints in the hand are subject to constraints, with degree-of-freedom in each joint having a minimum and maximum rotation, with the exception of the root joint, which can move and rotate freely. An additional simplification is the assumption of orthogonality between the joints, meaning the constraints are independent of the state of the model.

Under these assumptions, the pose of the hand can be specified by a set of 26 parameters. These parameters can be seen as representing points in a 26-dimensional state space, in which each point within an orthotopic region[1] represents a different

[1]This a region defined as the points within a particular set of coordinate intervals. An aligned “hyper-rectangle”.

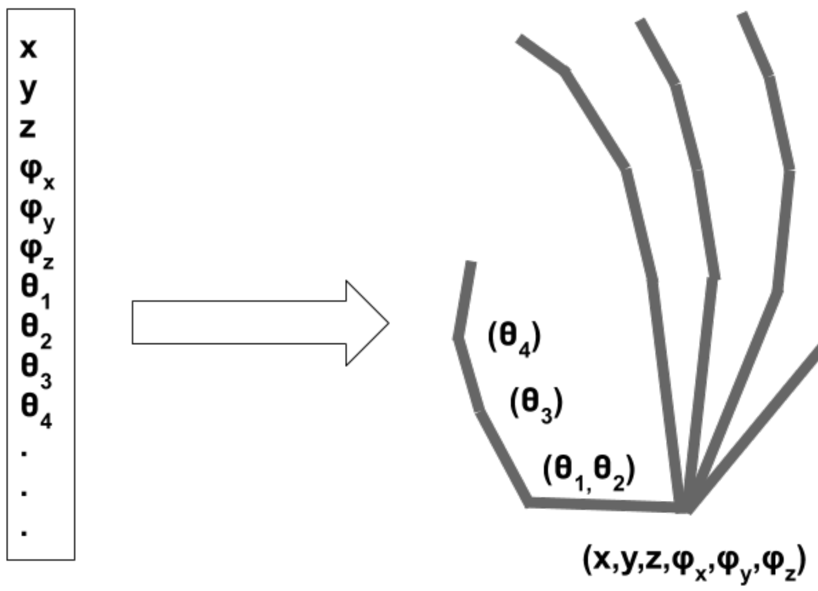

Figure 2.2: Diagram showing a state vector and its relationship to the pose of a particular skeletal model.

possible pose of the hand, and the goal of generative hand tracking can be seen as searching this region to find the best pose. Figure 2.2 shows a state vector and its relationship to the pose of a corresponding skeletal model.

In generative hand tracking, a kinematic model of this kind typically drives a geometric model of the hand, which can be compared against features extracted from input data. The specifics of the geometric model depend heavily on the features and the design of the algorithm. Alternatively, if the extracted features are fingertip or joint locations, a kinematic model model alone may be sufficient. As such, discriminative systems sometimes use kinematic models to find a physiologically plausible pose whose joint positions are close to those found through regression.

2.2 Optimisation Algorithms

In generative hand tracking, two things are assumed to be available: data containing information about the pose of the hand and a model of the hand from which features of that data may be predicted. The objective is then to find the pose that best explains the data. In mathematical terms, it could said that the goal is to maximise the posterior likelihood of a particular pose given the input data. One way to do this is to model the prior and joint probability distribution in order that the an estimate posteriori distribution can be calculated explicitly. Alternatively, the maximisation may be done implicitly through optimisation of an objective function that has been designed to take the same information into account. Objective functions may also be referred to as loss, cost or energy functions when they are to be minimised, and reward functions when they are to be maximised. In either case, the task of hand tracking under the generative paradigm is reduced, at least partially, to one of non-linear optimisation.

Since many hand tracking systems take this approach, some of the different optimisation algorithms that have been used in the hand tracking literature will be described. They generally require some initial guess at the solution that is refined in successive iterations. This initial guess may simply be a random sample from the space of possible poses or may be determined through some initialisation procedure.

Optimisation algorithms can be broadly divided into two categories: those that require the gradient of the objective function to be calculated as part of the update procedure and those that do not[CSV]. Those that do are referred to as gradient-based and those that do not as gradient-free or direct-search algorithms. Gradient-based methods may also be referred to as first-order, second-order, etc. according to the order of derivative that must be calculated. A distinction may also be made between algorithms that sample the search space at a single point that is updated iteratively and those that maintain a population of sample points and combine information from all of them in some way. These types are referred to as single-solution and population-based respectively.

The simplest gradient-based technique is Gradient-Decent[BBV04a], introduced in the nineteenth century by Cauchy. This is a single-solution algorithm in which the next sample point is determined by displacing the current one by a vector proportional to the gradient of the loss function at that point. That is,

$$p_{i+1} = p_i - \alpha \nabla L(p_i) \tag{2.1}$$

where α, the learning rate, is an arbitrary constant that may be decreased as the iterations proceed, and L is the loss function. An important related technique is Stochastic Gradient Descent (SGD), which works in the same way, except the gradient is estimated on the basis of a random subset of the information available[RM51]. This is done to prevent the algorithm from getting stuck at a local minimum. SGD is particularly relevant here, since it is also an important algorithm in the training of neural networks.

Another gradient-based algorithm is Newton's method[BBV04b]. This is a root finding algorithm that assumes the function is a locally linear or locally quadratic. In the simpler version of this algorithm, the sample point is updated according to the ratio of loss function's value to its partial derivative in each dimension. This is usually written as follows,

$$p_{i+1} = p_i - J(p_i)^{-1} L(p_i) \tag{2.2}$$

where J is the Jacobian matrix of partial derivatives,

$$J_i = \frac{\partial L}{\partial x_i} \tag{2.3}$$

Another formulation of this method uses the ratio of first to second-order derivatives, such that,

$$p_{i+1} = p_i - H(p_i)^{-1} \nabla L(p_i) \tag{2.4}$$

where H is the Hessian matrix of second-order derivatives,

$$H_{i,j} = \frac{\partial^2 L}{\partial x_i \partial x_j} \tag{2.5}$$

When formulated as a least-squares problem, this method can be simplified slightly and is referred to as the Gauss-Newton method[Deu11a]. Frequently, the Jacobian is unknown or cannot be computed efficiently. Algorithms that instead attempt to approximate one of these quantities in some way are called Quasi-Newton methods[Deu11b].

Another common gradient-based technique is the Levenberg-Marquardt algorithm[Deu11c]. This algorithm combines elements of the Gauss-Newton method and Gradient-Descent, such that it behaves more like the former when close to the solution and more like the latter otherwise. This is sometimes also referred to as Damped-Least-Squares or Damped-Gauss-Newton[MNT04].

The simplest gradient-free technique is Hill Climbing[Ski98a]. This is a single-solution algorithm that simply samples the region local to the current solution. If the sampled solution is better than the current one, it replaces it. This process is repeated iteratively. This could be seen as the gradient-free analogue of Gradient-Descent. There is also an analogue of SGD, Stochastic Hill Climbing (SHC)[Ski98b], in which each iteration only considers a random subset of the information available. If multiple samples are taken at each iteration and the discovered point with the lowest cost adopted, the resulting algorithm is referred to as best-first search[Kor96]. Beam-search is a finite memory approximation of best-first search that also has a stochastic variant[KC12].

An important variation of Hill Climbing is Simulated Annealing (SA)[Ski98b], in which the sample point is assigned a ”temperature” that will allow it to move to a worse solution with a certain probability. This probability is gradually reduced as the algorithm proceeds. This modification makes the algorithm much less likely to get stuck at a local optimum. SA can also be applied to other algorithms where a random component can be incorporated.

In high dimensional spaces, the approach taken to sampling can be critical. This is because naive approaches that may be appropriate in low dimensional spaces, such as evenly sampling the space around a particular point, quickly become intractable in higher dimensions. As a result, optimisation algorithms frequently draw inspiration from statistical filtering methods. Sequential Monte Carlo (SMC), also known as Particle Filtering, is a class of algorithms in which previous samples are used to guide future sampling[DM97]. This idea is closely related to that of Genetic Algorithms (GA) and Evolutionary Strategies (ES)[2] in which a the characteristics of a population are mutated and/or recombined over successive generations in order to optimise an objective function[BBM08]. Differential Evolution (DE)[SP97], a specific algorithm of this kind, will be discussed in detail in section 3.3.

There are also some sampling techniques designed to work with dynamic systems. Though they are not optimisation algorithms per se, they are frequently used to guide optimisation algorithms by reducing the size of the effective search space. The simplest of these is the Markov Chain[Gag17], in which the probability of the system transitioning to a new state is entirely dependant on the state that it is in. This is referred to as the Markov property. When incorporated with any kind of Monte Carlo sampling method, the result is referred to as Markov chain Monte Carlo (MCMC)[FJs$^+$05]. The Hidden Markov Model (HMM)[Bau72] is a related model in which the state of the

[2]Usually, GA refers to algorithms that operate over discrete or binary search spaces, while ES refers to algorithms that operate over real-valued ones.

system is considered unobservable and the statistical relationship between the observables and the hidden state of the system is approximated along with the transition probabilities. Both of these have been used in hand tracking several times to make dynamical predictions[Bra99, SBS02, NTTC05]. It should also be noted that simple first-order dynamical predictions are applicable to hand tracking, and have been used many times[DD92, Roh94, WN99].

Another gradient-free method is the Nelder-Mead algorithm[NM65]. In this algorithm, a population of sample points is initialised. The number of sample points is always one more than the dimensionality of the search space. In each iteration, the worst sample point is selected for replacement, which is done by reflecting its position about the mean position of the other points. Its distant from the mean is usually also multiplied by a constant slightly less than one to ensure that the algorithm converges.

Force-based optimisation algorithms[LK95] are ones in which the candidate solution is imagined to be in a dynamic system with forces acting on it. The forces represent the various contributors to the loss function, and are sometimes derived from a potential model. The system is updated incrementally such that it converges on a point of equilibrium.

Particle Swarm Optimisation (PSO) is another gradient-free population-based optimisation algorithm, due to Eberhart and Kennedy[EK95], that has become quite important in hand tracking. Like force-based optimisation, PSO also treats the state space as a dynamical one but uses multiple sample points, referred to as particles, and has them traverse the space with initially randomised velocities. The best quality solutions found at a given moment act as attractors, meaning the space around them is sampled more frequently. Pseudocode for a basic PSO algorithm is given in Algorithm 1.

In the very specific case when what is being optimised is the average square distance of the points in one point cloud to the points in a second reference point cloud, the Iterated Closest Points (ICP) algorithm is applicable[CM92]. This algorithm was

Algorithm 1 A basic PSO algorithm with reflecting boundaries. *U(a,b)* randomly samples a uniform distribution between the given bounds. $\mathbf{b}_l$ and $\mathbf{b}_u$ are the lower and upper bounds of the search space respectively.

```
for all $i$ in $S$ do                                          ▷ For each particle
    $\mathbf{x}_i := U(\mathbf{b}_l, \mathbf{b}_u)$            ▷ Initialise particles randomly in search space
    $\mathbf{p}_i := \mathbf{x}_i$                             ▷ Set best position to current one
    if $f(\mathbf{p}_i) < f(\mathbf{g})$ then
        $\mathbf{g} := \mathbf{p}_i$                           ▷ Select global best according to loss function
    end if
    $\mathbf{v}_i := U(-|\mathbf{b}_l - \mathbf{b}_u|, |\mathbf{b}_l - \mathbf{b}_u|)$     ▷ Initialise velocities
end for
for all $g$ in $gen\_max$ do
    for all $i$ in $S$ do
        for all $d$ in $D$ do                                  ▷ Update velocity with inertia and random
            $r_p := U(0,1)$                                    ▷ attraction to local and global best
            $r_g := U(0,1)$
            $\mathbf{v}_{i,d} := \omega \mathbf{v}_{i,d} + \phi_p r_p (p_{i,d} - x_{i,d}) + \phi_g (g_{i,d} - x_{i,d})$
        end for
        $\mathbf{x}_i := \mathbf{x}_i + \mathbf{v}_i$          ▷ Update position
        for all $d$ in $D$ do                                  ▷ Enforce boundary conditions
            if $\mathbf{x}_{i,d} < \mathbf{b}_{l,d}$ then
                $\mathbf{x}_{i,d} := \mathbf{b}_{l,d}$
                $\mathbf{v}_{i,d} := -\mathbf{v}_{i,d}$
            else if $\mathbf{x}_{i,d} > \mathbf{b}_{u,d}$ then
                $\mathbf{x}_{i,d} := \mathbf{b}_{u,d}$
                $\mathbf{v}_{i,d} := -\mathbf{v}_{i,d}$
            end if
        end for
        if $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ then            ▷ Update local and global best
            $\mathbf{p}_i := \mathbf{x}_i$
            if $f(\mathbf{p}_i) < f(\mathbf{g})$ then
                $\mathbf{g} := \mathbf{p}_i$
            end if
        end if
    end for
end for
```

originally intended to work with points that move together as a rigid rotor when the correspondences between the clouds are not known. The algorithm proceeds by temporarily associating each point in one cloud with the closest point in the other cloud and minimising that least-squares problem, which has a closed-form solution[3]. The associations are then remade on the basis of the new positions. This process is repeated until the associations no longer change. In the original case of a single rigid cloud, the algorithm is proven to converge on the true solution. In the case of a multi-part articulated point cloud, such as one driven by a kinematic hand model, the algorithm is not guaranteed to find a perfect solution, but is still widely used as a metaheuristic. The Coherent Point Drift (CPD)[MS10] algorithm is a similar algorithm in which non-rigid deformations of the point cloud are allowed but penalised.

2.3 Convolutional Neural Networks

Convolution Neural Networks (CNN) were first devised by Waibel et al.[WHH+95] for use in the field of speech recognition and subsequently introduced into computer vision for the purpose of handwritten digit classification[DGG+89, LBD+90]. Initially, CNNs were generally inferior to other algorithms at most tasks, as those other techniques made use of domain specific knowledge or more rigorous statistical reasoning. However, the highly parallel nature of the algorithm allowed CNNs to be GPU-accelerated with speed-ups of up to 60 times, meaning vast quantities of data could be processed by correspondingly large networks. As a result, CNNs began to outperform competing algorithms at image classification[CMM+11, KSH12], and were later applied to a wide range of vision problems.

The basic operation of a CNN is relatively straightforward. The input image is

[3]A closed-formed solution is one that is expressible as a self-contained mathematical formula, as opposed to the result of an iterative process.

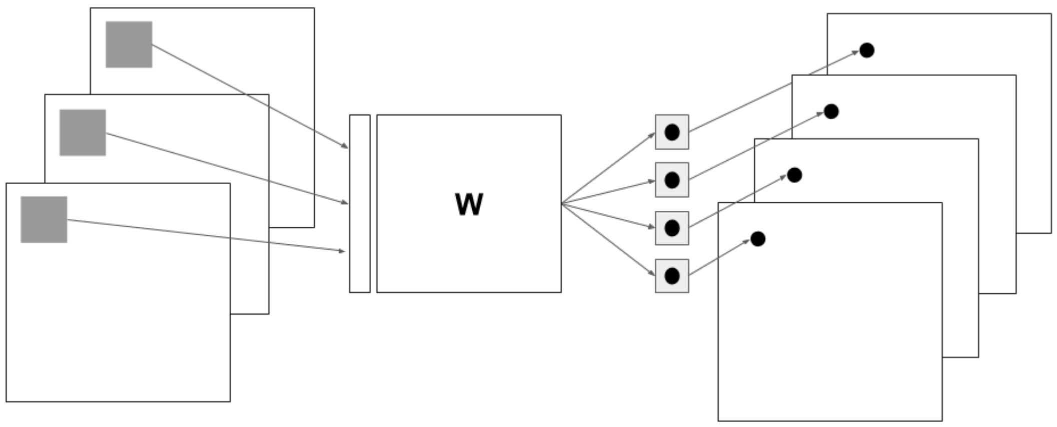

Figure 2.3: An illustration of the convolution operation.

divided into many equally sized and spaced regions. These regions are referred to as receptive fields and typically overlap. A set of kernels is also defined with each the same size as a receptive field. Each kernel is multiplied with the set of input variables in each receptive field and summed to create an output image of size equal to the number of receptive fields and depth equal to the number of kernels. This operation is referred to as convolution. Figure 2.3 contains a diagram of this operation.

A non-linear activation function is then applied to each of the variables in this output image before the process is repeated with a new set of kernels. Examples of common activation functions are shown in figure 2.4. The sigmoid function, $f(x) = (1 + exp(-x))^{-1}$, was originally the most widely used but has now been largely replaced by the rectified linear unit (ReLU)[GBB11], in which negative activations are set to zero and positive ones are left unchanged. This is because ReLU has constant gradient at high activations and therefore trains more efficiently, as neurons do not "saturate" at high activation. The output of each layer besides the last are called latent features or latent variables. Generally, a batch of examples is processed simultaneously

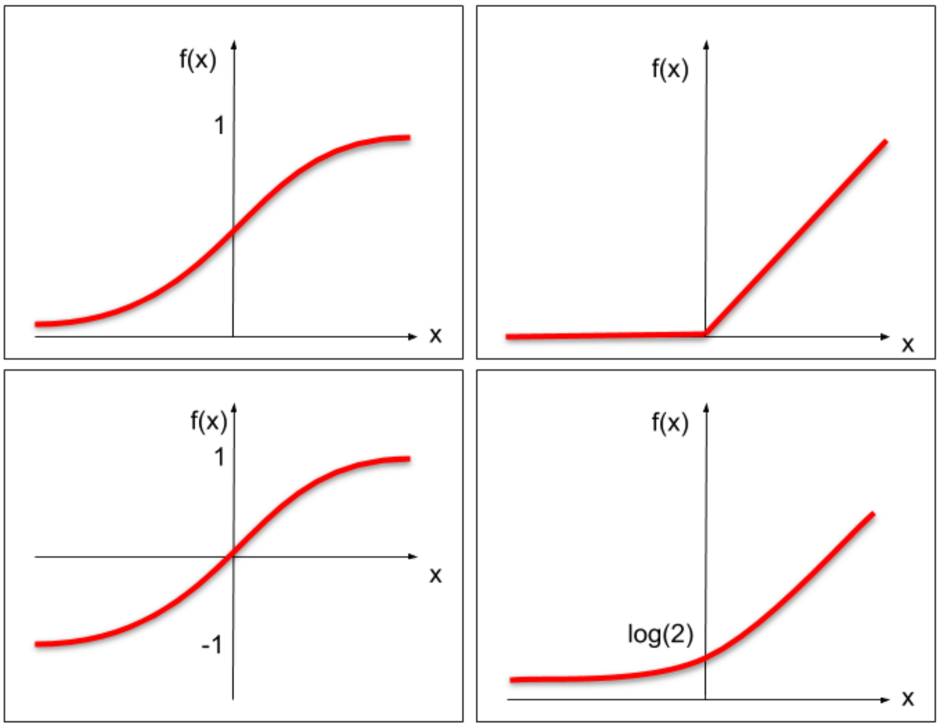

Figure 2.4: Illustrations of different activation functions. Top left shows the sigmoid function, $f(x) = (1+exp(-x))^{-1}$, bottom left shows the hyberbolic tangent function, $f(x) = tanh(x)$, top right shows ReLU, and bottom right shows the BNLL activation, $f(x) = log(1+exp(x))$, which is a continuously differentiable approximation of ReLU.

and training proceeds iteratively.

Most neural networks use some kind pooling, in which multiple latent variables from a local region of a single channel of a feature map are aggregated into a single variable to make a smaller output.[4] Max-pooling, in which the largest latent variable is selected to be the output, is by far the most common kind. Average pooling, in which the mean of the latent variables is used as the output, is also sometimes used. Other options include stochastic pooling, in which a random latent variable is selected from

[4]Although in practice the output of a pooling operation can be made to be the same size as the input, this is rarely done in practice as there is little advantage to doing so.

the input field.

Other operations may be applied between convolutions. Common ones include batch normalisation, in which the distribution of each latent variable in a batch is mapped to a unit normal distribution by subtracting the mean and dividing by the variance. This technique was proposed by Ioffe and Szegedy[IS15] to address the problem of internal covariate shift, a problem encountered in deep neural networks in which the later layers fail because the distribution produced by previous layers changes as training proceeds. Dropout, a technique introduced by Hinton et al.[HSK$^+$12], is also common. This is a form of regularisation in which a subset of features are randomly set to zero during training. The idea is that this will naturally cause the network to acquire some redundancy and generalise more effectively to new data as a result, since features in a new example may not be strongly recognised by a particular neural pathway, but the aggregate of multiple redundant ones may be enough to achieve the correct result.

The output layers, and sometimes also intermediate layers, of a neural network are subject to a loss function. The form of the loss function depends on the problem. In classification, the loss function is usually multinomial logistical loss and is used in conjunction with softmax function, which s the outputs to sum to one and hence be interpretable as the estimated probabilities for each of the classes. This loss function for a single example can be written as follows,

$$L = -log\left(\frac{e^{\hat{y}_i}}{\sum_{m=1}^{M} e^{\hat{y}_j}}\right) \tag{2.6}$$

where $\hat{y}$ is the network output, M is the number of classes and output features, and i is the index of the true class. When the output is permitted to belong to more than one class, cross-entropy loss is used. In this case, the sigmoid function is applied to the output rather than softmax, since the probabilities do not need to sum to one. This can be written as follows,

$$L = \sum_{m=1}^{M} p_m log(\hat{p}_m) + (1 - p_m) log(1 - \hat{p}_m) \tag{2.7}$$

where p is the one-hot encoding of the true classes and $\hat{p}$ is the network output after the sigmoid function has been applied. When the output is real-valued, the Euclidean distance between the output vector and its true value is the most commonly used loss function. This can be written as follows,

$$L = \sum_{m=1}^{M} (\hat{y}_m - y_m)^2 \tag{2.8}$$

where $\mathbf{y}$ is the true value of the output vector. In practice, these losses are summed or averaged across a batch a data samples.

The standard method of training a CNN is through SGD[LBOM12], or a variant of that algorithm, which is used to optimise all of the parameters in the network. The gradient is ascertained through the backpropagation of derivatives via the chain rule. This requires all of the functions used in the network to be piecewise differentiable. This means a finite gradient exists at every point in the function's domain, though the function itself may not be smooth at certain points.[5]

For some applications, such as classification, the output does not have any spatial extent, meaning it is a single value or set values that do not correspond to the geometry of the input image. In this case, the last few layers generally take the entire feature map as their receptive field, rendering them identical to the layers in standard neural network. These layers are sometimes called inner-product, fully-connected, or perceptron layers.

When the output does have a spatial extent the task is referred to as a dense estimation or image-to-image problem. This can be achieved with inner-product layers

[5]Max pooling is differentiable, with the partial derivative of the output being one with respect to the highest value input and zero with respect to the rest.

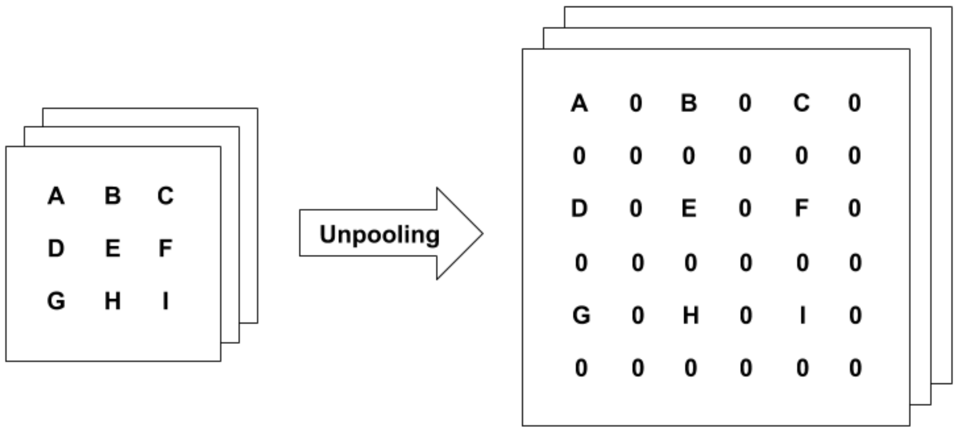

Figure 2.5: Diagram of the unpooling operation.

by making the final output large and reshaping to a 2D feature map. This approach does not take advantage of the locality constraints of the output features and severely limits the resolution of the output. Hence, contemporary architectures for this kind of problem typically forego the inner-product layers and are thus referred to as Fully Convolutional Networks (FCN)[LSD15]. Networks of this kind typically rely on some form of deconvolution. This is a convolution operation that maps a lower resolution feature map to a higher resolution one. In the abstract, this means the kernel is larger than the receptive field, and thus produces an different output for each position within it. In practice, deconvolution is never done this way, but the same effect can be achieved by several different methods, mostly commonly by padding out the feature map with zeros, a process known as unpooling (see figure 2.5) and performing standard convolution[ZTF11].

Residual networks (Resnets) were introduced in 2016 by He et al.[HZRS16] and allowed for much deeper networks than had been seen previously and consequently a significant improvement in classification performance. Residual networks contain residual units, which are so called because they estimate a residual from the identity operation, has been shown to be beneficial, particularly for deeper networks. In practical terms, this means the input of a layer is summed with its output, the convolution operation being defined such that the input and output are the same size. If the input and output are concatenated rather than added, the result is called a dense network[HLVDMW17].

Generative neural network models[GBC16] have come to prominence in the past few years and impacted on the field hand tracking. One such model is the variational autoencoder (VAE)[KW13]. These are similar autoencoders in that they map their input to itself via a low dimensional latent representation, the key distinction being that the latent representation is a zero-mean unit-variance normal distribution. This distribution can be sampled from to generate a realistic output. This is achieved by minimising both the reconstruction error and the Kullback-Leibler divergence[Kul59] of the distribution of latent variables from the unit normal during training. Another important generative neural model is the generative adversarial network (GAN)[GPAM$^+$14]. In this model, two subnetworks, referred to as the generator and the discriminator, are trained simultaneously. The generator maps variables sampled from a random distribution to approximations of an input image, while the discriminator attempts to classify the images it is shown as being either real or products of the generator. During training, the weights in the discriminator are updated to maximise the classification accuracy, while the weights in the generator are updated to minimise it. The end result for both models is the same: a network that will generate plausible data when given variables sampled from a known distribution.

2.4 Semantic Segmentation

Semantic segmentation is the task of associating each pixel in an image with a label corresponding to the class to which the object occupying that pixel belongs[GGOEO+18]. The task has been studied for decades, most early approaches using Markov random fields (MRF)[BKR11] on raw pixels in low resolution images, superpixels (i.e. small homogeneous image regions), or some other simple image feature[HS85, The83, Bes86]. MRFs are a class of graphical models applied to structured data where the state of a node in the graph is statistically dependent on its neighbours only. MRFs belong to the more general class of conditional random fields (CRF) in which nodes can be dependent on non-neighbouring nodes[Hri]. Shapiro et al.[SBB94] proposed a system for semantic segmentation using multiscale MRFs. This approach was adopted by He et al.[HZCP04], who relaxed the Markov properties of in order to learn discriminatively the relationship between features.

More recently, other techniques have been applied to the problem. Shotton et al.[SJC08] proposed an approach using random decision forests. Socher et al.[SLMN11] used recurrent neural networks on hand engineered image features. Farabet et al.[FCNL12] used a CNN to learn dense image features which were then pooled in superpixel regions and used in conjunction with a CRF to label each region. Papandreou et al.[PCMY15] also combined a CNN architecture with a CRF but took a weakly supervised approach to training. Pinheiro and Collobert[PC15] used an image classification network to provide a prior to a non-FCN per-pixel segmentation network. FCNs began to be applied in 2015[RFB15, LSD15, NHH15] and became a standard approach to the problem[JDV+17, LMSR17, CZP+18, FLW+19, LCS+19, GGOEO+18].

Semantic segmentation CNNs have also been applied to hand images on occasion. Neverova et al.[NWTN14] took a semi-supervised approach to train a VGG-like architecture to segment hand parts in depth images. Saleh et al.[SRAN+19] used multiscale low-level feature extraction and an FCN architecture on close-cropped RGB images to perform hand parts segmentation.

2.5 Related Work

In this section, the literature relating to markerless vision-based three-dimensional hand pose tracking will be reviewed. Algorithms that require gloves or visual markers [Dor94, Hol97, Lie05] or an unrealistically specific context for the tracking to be performed [SK99] will not be included. Similarly, algorithms that only track the position of the hand, or algorithms that only track the hand in the two-dimensional image space are considered out-of-scope. Pose reconstruction techniques that use only a single frame will be included, since they are a special case of hand tracking and generally very relevant in terms of the techniques applied. Since hand tracking techniques have tended to derive from human pose tracking, relevant papers in that discipline will also be included.

2.5.1 Early RGB-Based Techniques

The earliest attempts at biometric tracking considered partial or whole human body poses. The earliest such system was that of O'Rourke and Badler[OB80] from 1980, who introduced the idea of a kinematic model and fit it through iterative refinement to a set of keypoint locations on a 2D image. The keypoint locations were hand labelled and the question of how those locations could be automatically determined was not addressed.

Three years later, Hogg[Hog83] proposed a system that fully established the generative paradigm, with the two part process being described explicitly: "The visual problem can be divided broadly into two parts; namely, what should be described and how can such descriptions be derived from a time-varying 2D image." In this case, the kinematic model of the human body had an accompanying appearance model in which different body parts were represented by cylinders. The lines produced by projecting this cylinder model into the image plane were compared against the edge transform of the input intensity image using. Simple geometric primitives, such as cylinders[Roh94, RK94a], elliptical cones[WN99, GD96], as well as mesh models[OK94] became standard ways of representing surface geometry in both body and hand tracking. Edge features also became standard[Roh94, GD96, WN99, DBR00, HH96, OH99, WLH01].

Soon after Hogg, Akita[Aki84] proposed a method of feature extraction that distinguished the figure from the background by using the difference image of consecutive frames. This temporal approach to feature extraction also became common, either in the form of difference images[PT94, Roh94] or optical flow[ST03, YY00, SISB04, WH99, LMSO03]. Downton and Druit[DD92] also took an approach similar to Hogg, but constrained their search with first-order dynamics, in which the state of the hand is projected to the next frame according to its current estimated rate of change. Dynamical constraints would continue to be feature of tracking systems going forward. Kalman Filtering also became common[Roh94, WN99].

Hogg set the standard for markerless vision-based tracking and, as such, the first hand tracking system, due to Rehg and Kanade[RK94a, RK94b], worked in a very similar way, with a cylinder-based appearance model being compared to the edge features in the image. In this case, the Levenberg-Marquardt algorithm was used to fit the kinematic hand model, which had 26 degrees-of-freedom.

Almost all model-based tracking used kinematic models. There were some exceptions, however. Bregler and Malik[BM98] adopted a soft kinematic approach in which each body part was able to move as a rigid rotor, but the joint likelihood of two body parts being in a particular configuration falls away exponentially as the kinematic constraints between them are violated. This approach in combination with a Gaussian observation model effectively transformed the problem of tracking into one of maximising the a posteriori likelihood of the observation and the joint distributions. It also allowed for the formulation of an overall energy function that was differentiable, meaning a Quasi-Newton method could be applied. This idea has been used several times since in both body [ST03, SISB04, GPKT10, DF15] and hand[HSKMVG09, MES18] tracking. Another interesting exception was that of Heap and Hogg[HH96], who defined a deformable mesh model of the hand and used Principle Components Analysis to determine statistically the primary modes of deformation from synthesised data. These primary modes of deformation served in place of joint angles as the low dimensional pose representation.

Some researchers took the view that hand tracking can be seen as a high-dimensional non-convex optimisation and began to experiment with different kind of optimisation algorithms, all in much the same general framework as Rehg and Kanade. Lee and Kunii[LK95] used a force-based optimisation technique. Nirei et al.[NSMO96] and Wu and Huang[WH99] used genetic algorithms. Shimada et al.[SSKM98] used Beam-search. Ouhaddi and Horain[OH99] compared Levenberg-Marquardt, Nelder-Mead, and Powell's method[Pow64][6], finding that they performed similarly in terms of accuracy, with Nelder-Mead being preferred on the basis of computational complexity. Wu et al.[WLH01] used a Monte Carlo variant with importance sampling. Lin et al. used Sequential Monte Carlo[LWH02] and Stochastic Nelder-Mead[LWH04], which is essentially the simplex algorithm with some Monte Carlo like features.

Other researchers took the view that the search problem is trivialised if the tracking is good enough, and so attempted to learn statistical motion models. This was mostly done in body tracking, were movement is more smooth and predictable than it is in hand tracking, and has included HMMs[Bra99, SBS02, NTTC05], Baysian

[6]Powell's method is an approach to finding the root of a function with multiple arguments, in which each argument in optimised individually and iteratively. The actual optimisation is performed by some other algorithm.

networks[SBS02], and Markov Models[CGH05]. One interesting example in hand tracking was that of Zhou et al.[Z^+03], who used eigenanalysis to predict the motion of hands. In general, hand tracking seems not to benefit from elaborate dynamical models. The utility of even first-order dynamics is questionable, given how quickly hands can accelerate and the fact that it is difficult to maintain accurate dynamical information when parts of the hand are occluded, as they often are.

Besides these generative approaches, a significant strand of research was in discriminative pose estimation, which attempts to map appearance features directly to the pose either through learned regression or example-based methods. These approaches often benefit from more informative features than just edges or regions. An early example would be that of Ahmad[Ahm94], who attempted to find 2D fingertip locations on a binary image produced by skin colour segmentation. Pose estimation was then done by finding the nearest neighbour in a set of predefined poses. Shimada et al.[SKS01] found the closest matching silhouette in a database of predefined poses, then refined the relevant pose generatively. Athitsos and Sclaroff[AS03] also performed a database lookup to find a ranked list of hypothesis poses from which the best was selected according to how well it corresponded to edge features in the image. Romero et al.[RKK09] used histogram of gradient (HOG) features to encode the hand region, then k-nearest-neighbours to find the pose. Techniques such as this became more common after the introduction of depth cameras.

Some researchers also used more descriptive features to help fit generative models. Nolker and Ritter[NR02] also extracted fingertip locations but did so using neural networks on regions of interest discovered around the silhouette of the hand. A model was then fit to these fingertip locations. Lu et al.[LMSO03] attempted to track shading changes over time in order to fit hand surface normals under the assumption of Lambertian lighting conditions[7].

[7]Lambertian lighting conditions are a set simplifying about lighting in which surfaces are assumed

Some approaches required multiple calibrated cameras, beginning with Kuch and Huang[KH95] who simultaneously fit their generative model according to multiple viewpoints. Ueda et al.[UMIO03] used multiple views to perform a full voxel-based volumetric reconstruction of the hand, to which a kinematic model was fit. Delamarre and Faugeras [DF01] and Dewaele et al.[DDH04] both used short baseline stereo to produce dense point clouds to which kinematic models were fit, with Delamarre and Faugeras using gradient-descent with a differentiable loss function that matched surface normals, and Dewaele et al. using ICP. These methods are quite similar to the generative approaches used with depth cameras.

More recently, Zhang et al.[ZJC$^+$16] revived the idea of tracking from stereo pairs and introduced a benchmark dataset for this purpose. Their approach was similar to approaches that use a depth camera (see section 2.5.2) but a dense disparity map acquired from short baselines stereo pairs rather than camera input. Panteleris et al.[PA17] took a similar approach and also modelled hand-object interactions.

2.5.2 Tracking With Depth Cameras

Towards the end of the 2000's, depth cameras became increasingly important in hand and body tracking. A depth camera is a camera that outputs a 2.5d point cloud representing the surfaces in its field of view, usually while simultaneously outputting a corresponding RGB image. The most common kind are time-of-flight cameras, such as the Microsoft Kinect, which precisely measure the time it takes an emitted pulse of infrared light to reflect off a surface and be detected by a sensor. Structured light cameras were also used early on. These work by simultaneously shining two lasers onto a surface at slightly different angles and determine the distance from the interference

to be perfectly opaque and effects such as self-occlusion, self-shadowing, self-lighting, and specular reflection are ignored. Under these conditions, the brightness of a surface is determined only by its reflectance, its angle to the light source, and the strength of the source.

pattern produced. The relatively high cost and low accuracy of structured light cameras meant they became largely disused after time-of-flight cameras became available.

Although cameras of this type have no more to do with the human visual system than a sensor glove, and depth-based algorithms are therefore not strictly in the scope of vision-based hand tracking, the trend of using depth was significant enough that it would not be possible to give a full context for the current state-of-the-art without discussing them since many of the techniques used in contemporary RGB-based tracking were first used in this domain.

It should be noted that, while RGB-based hand tracking became much less common in the era of depth cameras, RGB-based body tracking has been a continuously active field of research. Sarafianos et al.[SBIK16] provide a comprehensive review of contributions to this field from 2008 to 2016.

As has been discussed, model-based tracking can be reduced to two main components: feature extraction and pose optimisation. To an extent, depth information trivialises the first part, since the model can, in principle, be fit to the raw input.

The initial attempts at hand tracking from depth data resembled the initial attempts at hand tracking from image data, with an early contribution from Bray et al.[BKMM$^+$04] reformulating the generative framework for input from a structured light camera. The model was fitted according to its surface normals using a differential loss function and stochastic gradient descent. Another early contribution was from Mo and Neumann[MN06], who took a data-driven approach that tried to identify fingers on the input depth map. Hamer et al.[HSKMVG09] took an approach similar to Bregler and Malik[BM98], using soft kinematics and MAP estimation through belief propagation.

Oikonomidis et al.[OKA11] used PSO to fit their model, which became a popular choice of algorithm. Melax et al.[MKO13] used ICP and incorporated first-order dynamics into their tracking. Sridhar et al.[SOT13] used a sum-of-gaussians model and fit it using least-squares gradient descent.

The optimisation problem was generally posed as trying to keep the near surface of the model as close to the point cloud as possible while not allowing it to exceed the boundaries of the hand segment. Sometimes other constraints, such as self-collision[QSW$^+$14], were also represented as terms in the loss function.

Qian et al.[QSW$^+$14] proposed ICP-PSO, a hybrid of ICP and PSO, and used it to fit a sphere-based hand model. Tagliasacchi et al.[TST$^+$15] used ICP to coarsely fit their model, then Levenberg-Marquardt to fine tune it. Sharp et al.[SKR$^+$15] combined PSO with GAs in order to make the tracking more robust to noise. Li and Zhou [LZ15] combined DE with particle filtering to optimise a mesh model to the input depth map and learned semantic priors. Taylor et al.[TBC$^+$16] generated smooth mesh models on the fly for whatever pose was required. The pose was then fit using Levenberg-Marquardt and a loss function that had terms representing dynamical constraints and a learned pose prior, as well as a term that matched the models fingertips to ones discovered in the data.

It became clear at this time that hand tracking systems cannot rely entirely on tracking from depth, since the results tend to degrade over time. This became known in the literature as the drifting problem[YSZ$^+$11]. In response, researchers began to consider automatic reinitialisation techniques, which were generally based on some kind of discriminative pose estimation, and integrate them into their systems. The most common approach was fingertip detection[SOT13, QSW$^+$14, TST$^+$15, TBC$^+$16]. Sharp et al.[SKR$^+$15] regress on global orientation from the current depth map.

Regression-based systems also became important in this period. Like generative approaches, regression is easier when depth data is available, since only the 2D location of the joint needs to be estimated as the 3D joint can be readily reconstructed using the depth value at that location. Some of these regression-based systems were very similar to the RGB-based ones described above, such as Rogez et al.[RKSI$^+$14], who used a cascade classifier on HOG features extracted from the depth map.

A significant trend was the application of Random Decision Forests (RDFs). These were first applied in body tracking by Shotton et al.[SFB+11] and Girshick et al.[GSK+11] at Microsoft Research and came to dominate that field for a few years[HVZM+12, SKS12, TSSF12, YKC13, CN13, PMTS+13, HWLX15, **?**, YJLSHDY15].

When applied to hands, however, RDFs did not work quite as well, probably because of the lack of visual indicators on the surface of the hand and the fact hands articulate and self-occlude in much more complex ways than arms and legs. The first application of RDFs to hand tracking was by Keskin et al.[KKKA12], and relied on a more complicated two-stage approach than was generally used in body tracking. Tang et al.[TYK13] used RDFs to classify the hand as being in a predefined pose, then refined this pose according to the depth data using a Gaussian Mixture Model. Similarly, Xu et al.[XC13] used an RDF classifier on Hough features extracted from the depth map then refined the result using gradient descent. Tang et al.[TJCTK14] and Sun et al.[SWL+15] both used cascade regression to find the joint locations on the depth map. Wan et al.[WYVG16] used surface normal features to regress on joint locations using an RDF conditioned on the results from the previous frame.

A more recent trend that has come dominate the field of hand tracking is the application of CNNs. Tompson et al.[TSLP14] used a CNN to generate joint heatmaps to which a kinematic model was fit using PSO. Oberweger et al.[OWL15] experimented with various CNN architectures to regress directly on 2D joint locations. Ye et al.[YYK16] used a cascaded CNN classifier to the same effect. Zhou et al.[ZWZ+16, ZSZ+16] used a CNN to regress directly on the state of a kinematic model. Forward kinematics were then performed so a loss based on the depth could be calculated. The kinematic part was all differentiable, allowing the network to be trained from this loss using standard backpropagation. Neverova et al.[NWNT17] used a CNN to label each pixel in the depth map as belonging to a part of the hand, with the small region around the pixel being the input. This information was then used as input into another CNN which regressed on joint locations.

Wan et al.[WPVGY17] used a generative model based on VAEs and GANs to create a low-dimensional latent hand pose space to which the input depth image could be mapped to with a CNN. Ge et al.[GLYT17] and Moon et al.[MYCML18] used a 3D CNN to produce a volumetric reconstruction of the hand, similar to Ueda et al.[UMIO03] but using depth data rather than multiple viewpoints. Guo et al.[GWC$^+$17] found regions in which each joint was likely to be and input them into separate CNNs to find the precise location of the relevant joint. Chen et al.[CWGZ19] took a similar approach but used a coarsely fit kinematic model to determine the regions. Choi et al.[CKR17] also regressed on 2D joint locations but used a proximity function on the surface of hand to guide the training. Ge et al.[GCWY18] regressed on the joint locations but took the whole 3D point cloud as input. Chen et al.[CWZ$^+$18] took this same approach but included a semantic segmentation network to provide a semantic prior to a network that regressed on joint locations. Wan et al.[WPVGY18] used FCNs to find joint heatmaps as an intermediate representation before regressing on joint locations. Malik et al.[MEN$^+$18] trained a CNN to predict both the hand pose and a detailed mesh model of the hand simultaneously. Malik et al.[MES18] proposed a system in which the kinematic constraints of the hand are encoded into the training loss of a regression CNN.

2.5.3 RGB-Based Hand Tracking With CNNs

In recent years, the widespread application of CNNs has lead to renewed interest in RGB-based hand tracking.

Zimmermann and Brox[ZB17] proposed a CNN approach that regressed directly on relative keypoint locations. This was achieved by first regressing on 2D heatmap joint location and estimating the viewpoint, i.e. the global orientation of the hand relative to the camera, then estimating the relative pose from this information.

Panteleris et al.[POA18] used an off-the-shelf 2D keypoint locator to find the joints in an image. The discrepancy between the reprojected joint locations in a kinematic hand model and the discovered joint location in the image was then minimised using the Levenberg-Marquardt algorithm in order to find the 3D pose and location of the hand.

Mueller et al.[MBS+18] took a similar approach to Zimmermann and Brox but used GANs to produce large amounts of realistic training data. They also localised the hand in 3D space as well as retrieving the relative pose. Iqbal et al.[IMBJGK18] regressed on both latent 2D heatmaps and a latent depth map representation, which were combined to estimate the 3D joint locations.

Cai et al.[CGCY18] and Dibra et al.[DMB+18] also regressed on the pose but also used depth images corresponding to the input RGB during training. This was done by rendering a model of the hand as a synthetic depth image. This allowed the training set to be extended to include unlabelled images. Rad et al.[ROL18] also took advantage of depth information during training but did so by mapping depth images to pose via a low-dimensional representation to which the input RGB was then mapped. Nicodemou et al.[NOTA18] attempted to learn to map RGB to depth directly, so that the resulting depth map could be used as input into any algorithm that requires depth.

Spurr et al. used VAEs to encode multiple modes of data to the same latent space. The modes included RGB, depth, and 2D and 3D pose. The shared latent space was achieved by training separate encoder and decoders for each modality in a round-robin fashion. The resulting model allowed any modality to be generated from any other. Yang and Yao[YY19] took the same approach but explicitly embedded variation due to background and viewpoint in a subset of their latent variables, the assumption being that the remaining variation will be more relevant to the pose.

Baek et al.[BKK19] proposed a CNN architecture that predicted a mesh representation of the hand simultaneously with the hand pose. The mesh was then used to refine the pose according to extracted 2D features.

2.6 Summary

As discussed in section 2.5.1, the earliest hand tracking systems were RGB-based and took a generative approach. The limitations of those early systems stemmed from the ways in which features were extracted from the RGB input, with an overwhelming reliance on simple line and region features. Contemporary RGB-based hand tracking systems, such as those discussed in section 2.5.3, generally use deep learning to extract features, which are used in a discriminative manner. The approach described in the following chapter attempts to bring together the feature extraction capabilities of deep learning with the conventional generative approach to hand tracking. It does this by semantically segmenting input hand image in a manner similar to the systems described in section 2.4. Dense semantic features allow for better tracking performance compared to the simple features used in early generative hand tracking systems, as they are much less ambiguous and more relevant to the pose of the hand.

Chapter 3

Methodology

In this chapter, a novel method of hand tracking will be presented. The basic idea is inspired by conventional generative hand tracking systems, in which features were extracted from the input RGB image and then matched to those generated by a kinetic model of the hand in order to find the pose. The features extracted are probabilities for each pixel being of a particular class, with classes corresponding to parts of the hand, and are produced by a convolutional neural network. The generative part of the system synthesises an equivalent label map by way of a kinematically-driven mesh model. The pose of the model is optimised using a specialised version of the Differential Evolution algorithm.

This method, which will be described in detail in the following sections, is comparable to the contemporary RGB-based pose estimation algorithms in that it makes use of deep learning but also follows in the tradition of the earliest hand tracking systems by taking a generative, model-based approach to recovering the kinematic state of the hand.

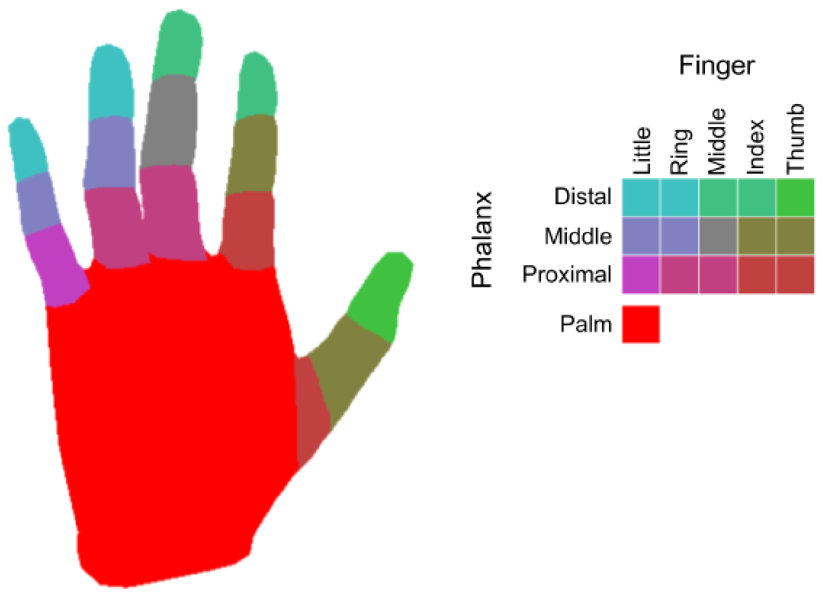

Figure 3.1: Hand mesh model with semantic labels.

3.1 Hand Model

The hand is modelled using the conventional 26 degree-of-freedom hand model described in section 2.1. This kinematic model is used to drive a low-polygon mesh model adapted from one made freely available online[Mat13]. The parts labels were added to the model by colouring the vertices in Blender. Each vertex in the mesh is attached to one bone in the model. Since there are no textures and the angular range of the joints is small, there is little advantage to interpolation. When placed in a particular pose and rendered, the model effectively provides a prediction of the semantic segmentation of an image of a hand in that pose. Figure 3.1 show the hand mesh model coloured according to the parts labels.

3.2 Semantic Segmentation Of Hand Images

Semantic segmentation is the task of assigning a label to each pixel in an image according to the object contained in that region of the image, or lack thereof. In the context of hand tracking, an ”object” is a part of the hand. The parts of the hand are defined in order that they correspond to the rigid components of the hand model described above, i.e. the palm, the phalanxes and the thumb metacarpal. A pixel in an image of a hand can belong to one of these classes or be part of the background.

3.2.1 Estimating Semantic Ground Truth

Supervised learning requires that every training example be labelled. This means a ground truth semantic map is required for each image in the dataset. There are two ways in which this might be done. One is to use synthetic images, for which corresponding semantic images may be rendered simultaneously with RGB ones. This approach has the advantage that the semantic maps will correspond exactly to the relevant regions of the image, but also has the problem that the synthesised RGB images may not be realistic enough for the resulting model to be used on natural images.

The alternative is to estimate the semantic ground truth of natural images. It is possible to do this in the context of hand tracking as ground truth joint location estimates are available. This means that an estimate of the semantic ground truth can be acquired by fitting the model described above to the joint location ground truth and rendering the result. To do this, the algorithm described in section 3.3 is used to minimise the least-squares error between corresponding joints in the model and ground truth.

3.2.2 Architecture

In order to perform semantic segmentation, an FCN architecture inspired by current state-of-the-art semantic segmentation systems is used. The architecture consists of an

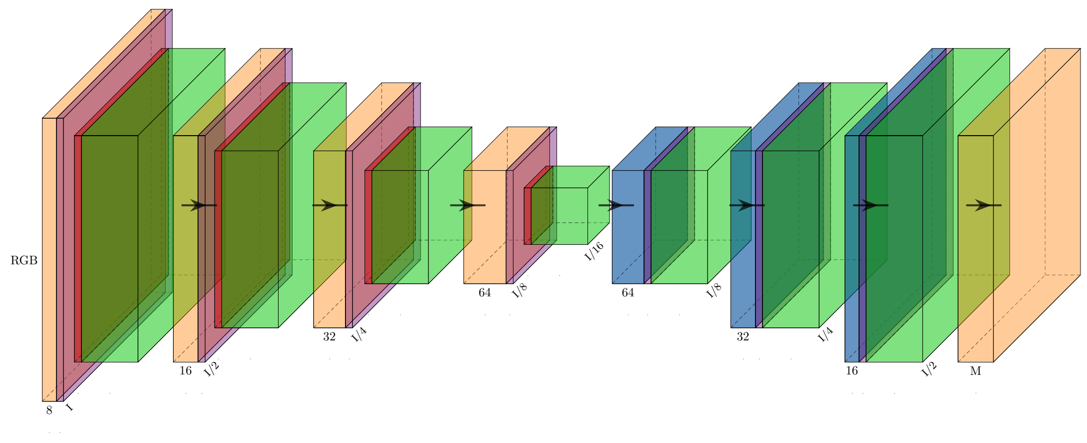

Figure 3.2: The network architecture. Orange blocks represent convolution. Purple blocks represent sequences of batch normalisation, scaling, ReLU, and dropout. Red blocks represent max pooling. Green blocks represent four-layer residual units. Blue blocks represent interleaving deconvolution.

encoder part and a decoder part. The encoder part contains convolution and pooling operations that decrease the linear dimension of the feature map by a factor of two. Correspondingly, the decoder part contains deconvolution operations that increase the linear dimension of the incoming feature map by a factor of two. These are interspersed with residual units consisting of four convolutional layers. Batch normalisation, scaling, dropout, and ReLU layers are also used throughout. Softmax is applied to the output of the final convolutional layer. The end result is a probability map that is half the size of the input RGB image. Figure 3.2 shows the network architecture.

The deconvolution operation was defined through interleaving the results of four stride one convolutions. This technique was proposed by Laina et al.[LRB$^+$16] in the context of depth estimation, another dense estimation problem. They showed that the technique is mathematically equivalent to conventional deconvolution, in which unpooling is performed before a convolution, but more efficient as zero-multiplications

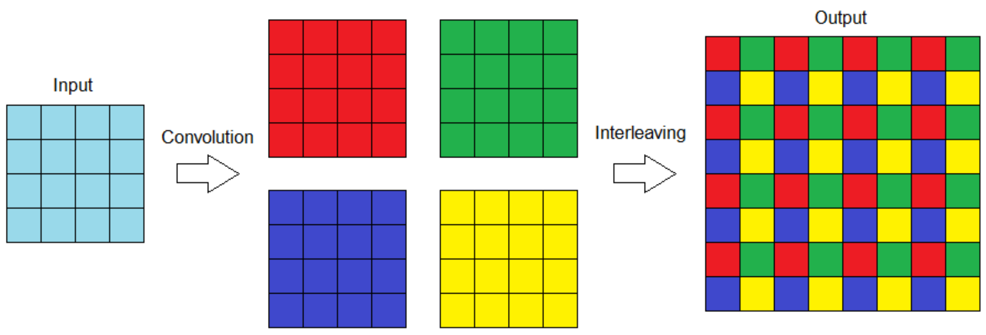

Figure 3.3: Diagram showing deconvolution through interleaving.

are avoided[1]. Figure 3.3 shows a diagram of how interleaving deconvolution operates.

3.2.3 Training

The model was trained using a loss function designed to maximise the per-class accuracy of the final result. To this end, the standard multinomial logistic loss was modified such that each class was weighted in the overall loss calculation according to the inverse of its frequency in the ground truth. This is necessary, as, unlike in other semantic segmentation context where the dataset can be selected to contain a roughly equal number of instances of each class, hand images are intrinsically biased toward classes that tend to occupy more space in the image. This loss function can be written as follows,

$$E = -\frac{1}{N}\sum_{i}^{N} H_{c_i} log(\hat{p}_{c_i}) \tag{3.1}$$

where

$$H_c = \frac{M}{N}\sum_{i}^{N}\sum_{c'}^{M}\delta_{c'c} \tag{3.2}$$

[1]The technique is precisely equivalent to conventional deconvolution when specifically shaped kernels are used, and practically equivalent otherwise.

c_i is the ground truth label of pixel i, N is the total number of pixels in the batch, and M is the number of classes.

Training was done using SGD with batches of 16 examples, as this was roughly the maximum that could be processed using the graphics memory available. A momentum of 0.9 and weight decay of 0.0005 were used, as these are fairly common choices in CNNs having been used by Krizhevsky et al.[KSH12]. Training proceeded for a million iterations, which was more than enough for the networks to converge. The initial learning rate was 0.01 and decreased by a factor of ten every fifty-thousand iterations.

3.2.4 Factoring out hand segmentation

Two variations of the architecture were trained and deployed. One only classifies pixels as background or hand, the other ignores any pixel labelled background and assigns a hand part label to every pixel. The is necessary since, when both of these task are learned simultaneously, the network does not train well. It is also convenient to isolate the hand segmentation because it requires data with a large diversity of backgrounds to generalise effectively, whereas the parts labelling only requires a representative sample of hand poses.

The results of both networks were then recombined as follows,

$$p(b) = kp_1(b), \tag{3.3}$$

$$p(c) = (1 - kp_1(b))p_2(c), \forall c \neq b, \tag{3.4}$$

where p_1 and p_2 are the outputs of the background and hand part networks respectively, b is the background label, and k is a constant between 0 and 1, which will be referred to as the *background factor*. The background factor can be adjusted to create final output depth maps that are more or less sensitive to hand part labels. The effect of this factor is discussed more in section 4.2.1.

3.3 Semantic Fitting

In order to evaluate a pose of the hand model, a cost function must be defined. This can be done in several ways. The most obvious would be to calculate the total negative-log-likelihood of the whole image. This cost function would be written as follows,

$$L(\theta) = -\sum_{i}^{N} log(p_i(R_i(\theta))) \tag{3.5}$$

where θ is the state vector of the hand model, p_i is the probability distribution over the possible labels for the ith pixel in the network output, and R_i is the label of the ith pixel in the rendered semantic image. This function is essentially the posterior probability of the hand being in the state given the network output on a logarithmic scale.

Alternatively, the sum-of-complements cost function may be used. Using this function means the probability for each pixel is being considered independently on a linear scale. It is written as follows,

$$L(\theta) = \sum_{i}^{N} 1 - p_i(R_i(\theta)) \tag{3.6}$$

Another cost function that will be considered is the number of incorrect pixels. This is simply the number of pixels whose probability for the true label falls below a certain threshold and written as follows,

$$L(\theta) = \sum_{i}^{N} \begin{cases} 1 & p_i(R_i(\theta)) < t \\ 0 & p_i(R_i(\theta)) \geq t \end{cases} \tag{3.7}$$

The performance of these cost functions will be compared in section 4.4.1.

3.3.1 Optimisation

Given a choice of cost function, the task of hand tracking is now reduced to one of non-linear optimisation. There are several algorithms that could potentially be used for this. The one that is used is Differential Evolution (DE), which is a population-based algorithm that falls into the general category of Evolutionary Strategies and was proposed by Storn and Price[SP97].

DE is suitable for this task because its gradient-free, fast to converge and robust to noise and non-linearities. Given the nature of the optimisation problem, there is no obvious way to use ICP, as this requires the problem be formulated as one of matching 3D point-clouds whereas only 2D semantic information is available in this case. Because semantic maps are subject to self-occlusions, the search space is highly non-linear, meaning algorithms such as stochastic hill climbing or Nelder-Mead would not work unless the model was initialised very close to the solution and the quality of the tracking would be fragile. Since DE is oriented towards global optimisation it can handle these non-linearities and should also be somewhat robust to drifting and tracking failures. Since PSO was designed to search high-dimensional spaces efficiently, it may seem like a good choice. PSO does not work well in practice, however (see section 4.4.3).

As is the case in all evolutionary strategies, DE maintains a population of sample points, referred to as agents, in a real-valued search space. Over successive generations, new agents are created by mutating and recombining existing ones and replacing them according to a set of stochastic rules. The distinguishing feature of DE is that the new agents are created by adding to one agent a vector proportional to the average difference between pairs of existing agents. This allows the population to self-organise incrementally according to the structure of the search space, without out the need to define or estimate a covariance matrix explicitly. This mutated vector is randomly combined with one from the current population to create a trial vector. The specific way in which this combination happens in called the crossover scheme. The trial vector replaces the original if its loss is lower. For each generation, this process is repeated for each agent in the population. Pseudocode for the basic algorithm is shown in Algorithm 2.

Many variants of DE are possible depending on how the agents are selected, how many pairs are selected, and the specific of the replacement rule. The original proposers of the algorithm suggested the following notation to describe its variants,

$$DE/x/y/z \tag{3.8}$$

x represents the way in which the agent that will be mutated is selected, with common choices being "rand" and "best", referring to the case in which the agent is selected randomly and the case in which the current best solution is selected. y represents the number of pairs used to calculate the mutation. z represents the crossover scheme, with the most common being "bin", which is an abbreviation of "binomial" and refers to the case in which each component is either selected or not selected independently with a certain probability.

The tracking algorithm is based and DE, with modifications made to make it suitable for the task at hand. A variant in which one pair of agents is used to calculate the mutation and a binomial crossover scheme is used as well as an approach to agent selection in which the best agent is selected half of the time and a random one the other half. This variant can be written in the above notation as,

$$DE/best-rand/1/bin \tag{3.9}$$

Algorithm 2 The basic DE algorithm ($DE/rand/1/bin$) due to Storn and Price[SP97]. The functions *randi()* and *randu()* randomly sample integers and real numbers from the the interval 0 to 1 respectively. *gen_max* is the number of generations the algorithm runs for, *NP* is the number of agents in the population, *D* is the dimensionality of the search space, *CR* is the crossover rate (this version of the algorithm uses binomial crossover), *F* is the differential weight, and *evaluate* is the loss function. *x1* is a randomly initialised set of vectors and *cost* the corresponding losses, the initial evaluations of which are not shown.

```
$count := 0$
while $count < gen\_max$ do
    for all $i < NP$ do
        do                                          ▷ Pick three distinct vectors
            $a := randi()\ mod\ NP$
        while $a \neq i$
        do
            $b := randi()\ mod\ NP$
        while $b \neq i$ and $b \neq a$
        do
            $c := randi()\ mod\ NP$
        while $c \neq i$ and $c \neq a$ and $c \neq b$
        $j := randi()\ mod\ D$                      ▷ Randomly pick and mutate first parameter.
        $trial(j) := x1(c, j) + F \times (x1(a, j) - x1(b, j))$
        for D repeats do
            $j := (j+1)\ mod\ D$
            if $i = j$ or $randu() \leq CR$ then    ▷ Randomly select parameters to mutate
                $trial(j) := x1(c, j) + F \times (x1(a, j) - x1(b, j))$
            else                                    ▷ according to weighted differential
                $trial(j) := x1(i, j)$
            end if                                  ▷ or keep the same.
        end for
        $score = evaluate(trial)$                   ▷ Evaluate trial vector with loss function.
        if $score \leq cost(i)$ then                ▷ Keep trial vector if better than current one.
            $x2(i) := trial$
            $cost(i) := score$
        else
            $x2(i) := x1(i)$
        end if
        $x1 := x2$
    end for
    $count := count + 1$
end while
```

Algorithm 3 DE-based tracking algorithm.

```
randomly initialise population of N agents
for each frame do
    run DE/(best-rand)/1/bin for gen_max generations
    sort agents according to loss
    select best agent as result for current frame
    replace worst M agents with mutations of best
end for
```

This variant of the algorithm is run on each frame for a fixed number of generations. After each frame, the highest cost agents are replaced. The new agents are created by sampling a Gaussian distribution centred on the current best agent and a fixed diagonal covariance matrix, with each entry proportional to the interval between the boundaries of the search space in the relevant dimension. This allows the algorithm to deal with the changing loss landscape without losing the information gained on the previous frames. Pseudocode for the tracking algorithm is shown in Algorithm 3.

3.4 Summary

The proposed technique allows hand tracking to be performed on RGB images. Although it makes use of CNNs like the recent RGB-based described in section 2.5.3, the approach more closely resembles more conventional RGB-based tracking algorithms (such as many of those described in section 2.5.1), since it first extracts features from the RGB input, then recovers the hand pose through analysis-by-synthesis. The approach could therefore characterised as an application of deep learning and generative paradigm to RGB-based hand tracking. A diagram of the full proposed hand tracking pipeline is shown in figure 3.4.

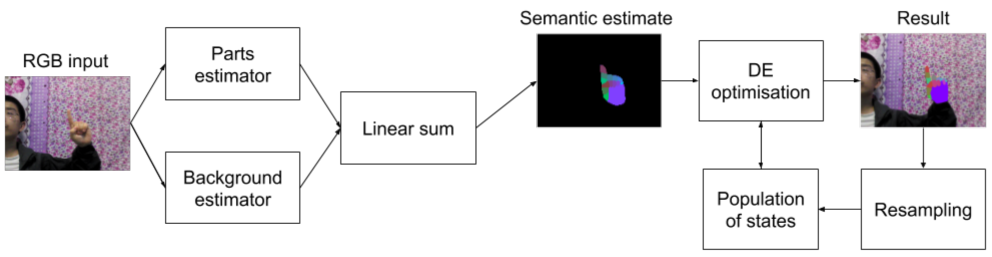

Figure 3.4: The proposed hand tracking pipeline. The left hand side shows the semantic segmentation of each input frame. The shows the optimisation using these semantic maps to produce a 3D pose estimate for each frame.

Chapter 4

Experiments and results

4.1 Datasets

The technique described in the previous chapter was tested using several different publicly available datasets, the main one being the stereo hand tracking benchmark (STB)[ZJC$^+$16]. This is a challenging dataset consisting of twelve 1500 frame sequences of hand motion performed against six highly varied backgrounds. For each background, two routines, “Counting” and “Random”, are performed, with “Random” being significantly more challenging in terms of the speed at which the subject moves and the diversity of gestures.

RGB and depth from a depth camera and ground truth joint location estimates are provided. The dataset also provides paired stereo images from a short-baseline stereo camera. The baseline of the stereo camera and the focal lengths of both cameras are also given. Unfortunately, the field-of-view (or other quantities from which it could be calculated) are not given. For the depth camera, this quantity was found on the manufacturer’s website. For the stereo camera, it was estimated by rendering the ground truth keypoint locations against the left RGB image. This was necessary for the experiment in section 4.4.5.

This dataset is the only one on which the entire pipeline is tested. Other common hand tracking datasets are not applicable, either because they only contain single images and not sequences (RHD[ZB17]) or because they either do not provide RGB (MRSA[QSW$^+$14], ICVL[YYS$^+$17]) or the provided RGB is not of good quality (NYU[TSLP14], B2RGB[PA17]).

The Rendered Handpose Dataset (RHD)[ZB17] is a large dataset of synthetic images generated procedurally by randomly combining sampling from a set of background and human models as well a range of viewpoints and hand poses. The dataset consists of 41258 training and 2728 testing examples, with RGB, depth, ground truth joint locations, and ground truth semantic maps. This dataset was used in the training and evaluation of the semantic segmentation CNNs.

MSRA[QSW$^+$14] is a dataset of depth sequences output from a depth camera. The dataset contains six sequences of 400 frames each from six different subjects. Each frame has had the background removed algorithmically. Semi-manually determined ground truth joint locations are also provided. This dataset was used to evaluate the tracking algorithm independently of the semantic estimation and fitting part of the pipeline.

4.2 Semantic Segmentation

In order to perform semantic segmentation, the STB dataset is divided into a training set and a testing set, with the testing set containing two sequences with the same background ("B1Counting" and "B1Random") and the training set containing the rest, all which have different backgrounds from the testing set. All images are downsampled by a factor of two and ground truth semantic maps are estimated in the manner described in section 3.2.1.

While this training set is sufficient for the parts estimator, the background estimator struggles, presumably due to there only being five backgrounds. The RHD training set is therefore included when training this network, since it contains a greater diversity of backgrounds. Given this training data, the background estimator performs quite well. Table 4.1 shows a comparison of the background estimation for the whole system with some contemporary state-of-the-art hand segmentation techniques. The metrics shown in the table are calculated as follows,

$$IOU = \frac{TP}{TP+FP+FN} \tag{4.1}$$

$$precision = \frac{TP}{TP+FP} \tag{4.2}$$

$$recall = \frac{TP}{TP+FN} \tag{4.3}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{4.4}$$

where TP, FP, TN, and FN and the true positive, false positive, true negative, and false negatives rates respectively. For the purpose of these metrics, the hand is the positive class and the background is the negative one.

According to IOU and f1-score, the proposed system is roughly as good or better than the examples from the literature. The example with higher recall[ZB17] also had very low precision. This is because the network was being used to coarsely segment the hand so it could be cropped out of the input and passed to a subsequent network stage. This is not true in the case of the proposed system, as the tracking algorithm requires a per-pixel estimate of the location of the hand in the input. The example with higher precision and lower recall[BKK19] tended to give results in which the hand area was smaller than it appeared in the input. This would cause problems in the proposed

	IOU	Precision	Recall	F1-score
Proposed	0.633	0.733	0.822	0.775
Baek et al.[BKK19]	0.651	0.828	0.753	0.789
Zimmermann et al.[ZB17]	0.354	0.365	0.921	0.523
Urooj and Borji[UB18]	0.527	0.717	0.666	0.690

Table 4.1: Comparison of the background estimation with examples from the literature on the RHD testing set. The metrics are intersect-over-union (IOU), precision, recall, and f1-score. Higher is better in all cases.

system, since the determination of the position of the hand in 3D space requires a good estimate of the projected area of the hand.

4.2.1 Effect of background factor

The performance of the semantic segmentation algorithm as a whole depends on the background factor, as well as the metric used to evaluate it. Figures 4.1 and 4.2 show there is a trade-off to be made between per-class and per-pixel accuracy, with the former tending to decrease as the background factor is raised and the latter tending to decrease. The mean class accuracy also decreases as the background factor approaches zero and the background is misclassified as one the hand regions (typically the palm, as this is the next most populated class). Figure 4.3, which contains qualitative results for a particular example across a range of background factors, also demonstrates this trade-off, as segmentation results for higher background factors are less blurry, with the boundary between the background and hand more precisely defined, but also tend to lose important details around the edge of the hand region, such as the shape of the palm and protruding fingers.

This trade-off can also be seen in figures 4.4 and 4.5, which show the f1-score and intersect-over-union metric respectively (with background as the negative class and hand as the positive class, as above), with both being maximal when the background factor somewhere around 0.5. As such, this is the value that was used to produce all

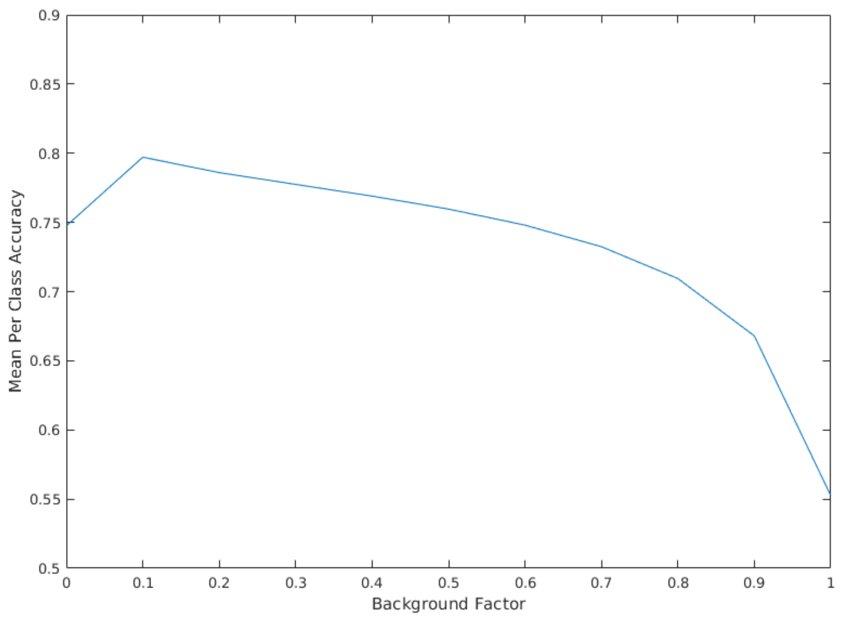

Figure 4.1: Mean per-class accuracy across a range of background factors.

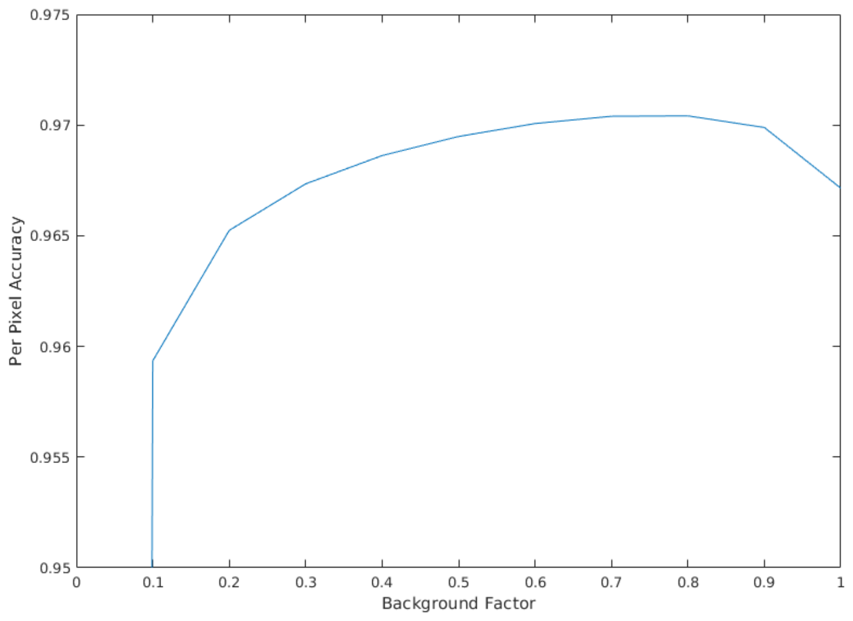

Figure 4.2: Per-pixel accuracy across a range of background factors.

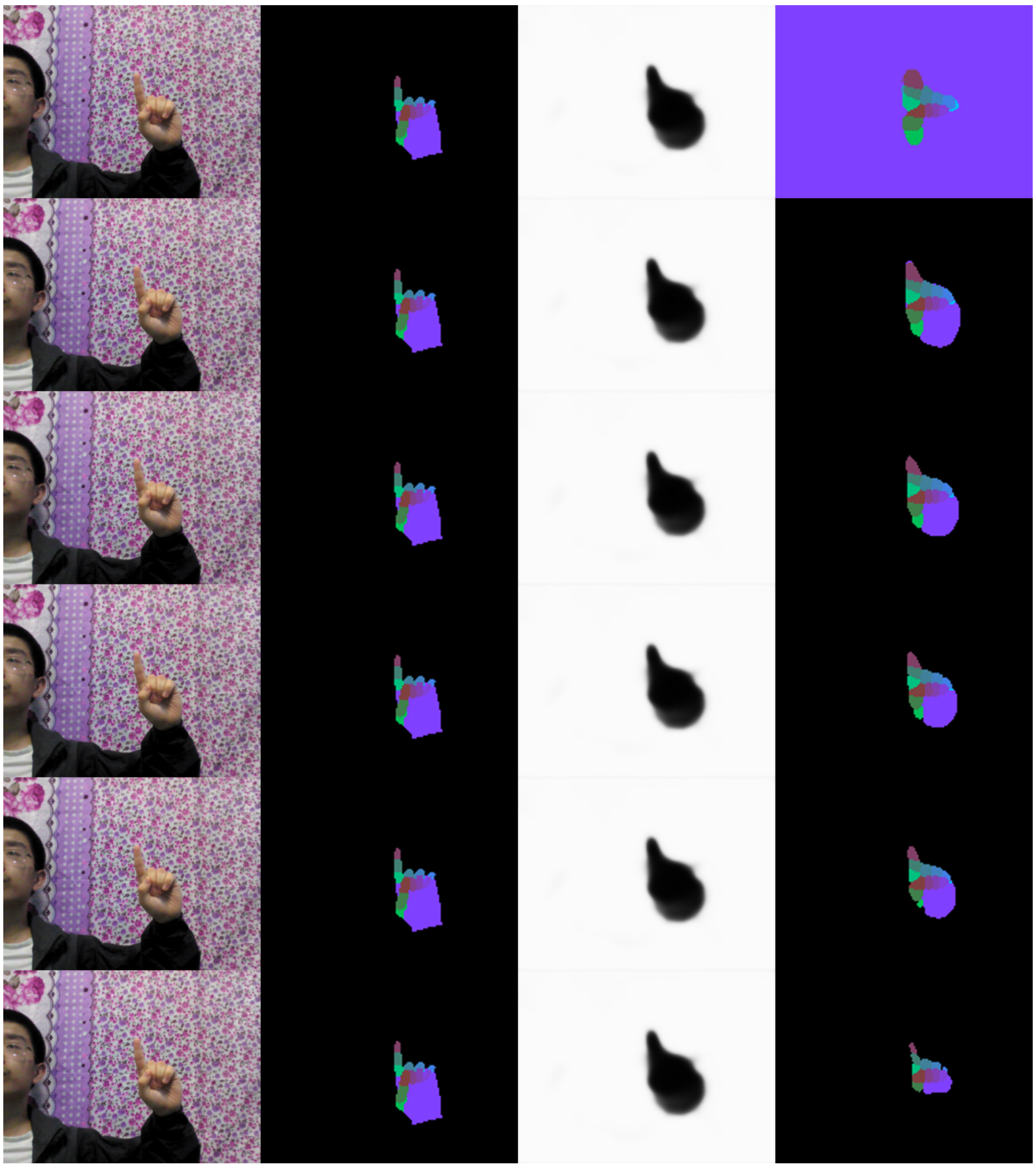

Figure 4.3: Segmentation results on the same input frame for different background factors. The columns from left to right show the input RGB image, the estimation semantic ground truth, the output of the background network, and the final segmentation result. (The first three columns are identical, having been repeated for clarity and consistency with subsequent figures.)

semantic segmentations used as input to the tracking algorithm.

Figures 4.6 and 4.7 show qualitative examples for a background factor of 0.5. These examples were selected from the STB testing set at evenly spaced intervals.

4.3 Tracking Algorithm

In order to evaluate the tracking algorithm described in section 3.3.1, it was applied to two hand tracking problems, both making use of the MRSA dataset. The first was a very simple case in which the algorithm was used to match the keypoints in a hand model to the ground truth keypoint locations. The cost function in this case was simply the sum of square distances from the keypoints in the model to the corresponding keypoints in the ground truth. The second was a depth-based hand tracking problem that made use of the algorithm devised by Qian et al.[QSW$^+$14] by adopting the same cost function. The DE-based algorithm was compared against an implementation of Qian et al's ICP-PSO in both cases[1]. This algorithm depends heavily on a data-driven fingertip detection subroutine, which is was approximated by coarsely fitting the model to the ground truth fingertip location using a short run of DE, with results reported both with and without this reinitialisation performed. All quantitative tracking results are in millimetres. In all cases, the relevant algorithm was run five times and the results averaged.

Table 4.2 and figure 4.8 show the results for the first case with fingertip reinitialisation. It can be seen that DE generally outperforms ICP-PSO, though both algorithms perform near perfectly, with error mainly arising from differences in the positions of joints in the model and ground truth and noise in both the sensor data and estimated

[1]For clarity, Qian et al.'s algorithm will be referred to as ICP-PSO even in the first case when the point correspondences are known.

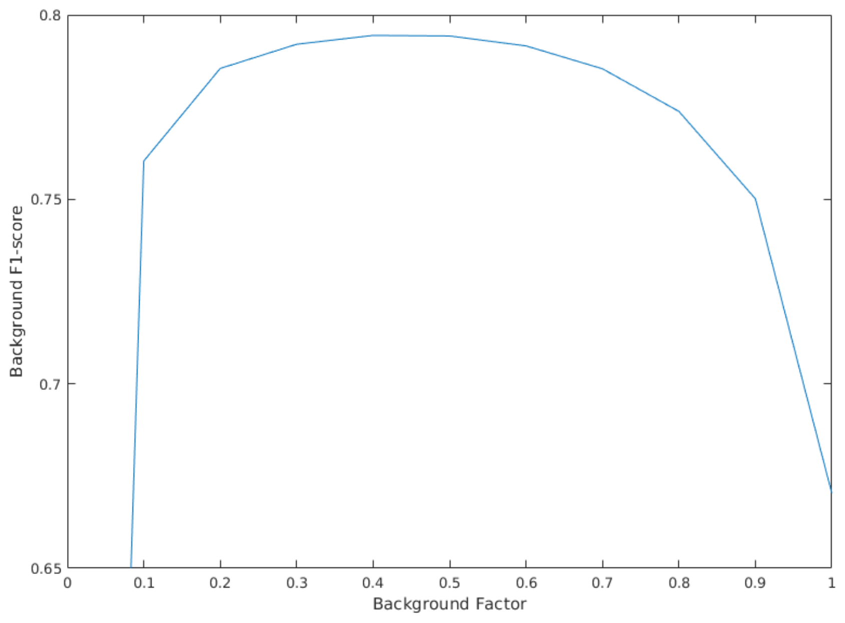

Figure 4.4: F1-score for a range of background factors.

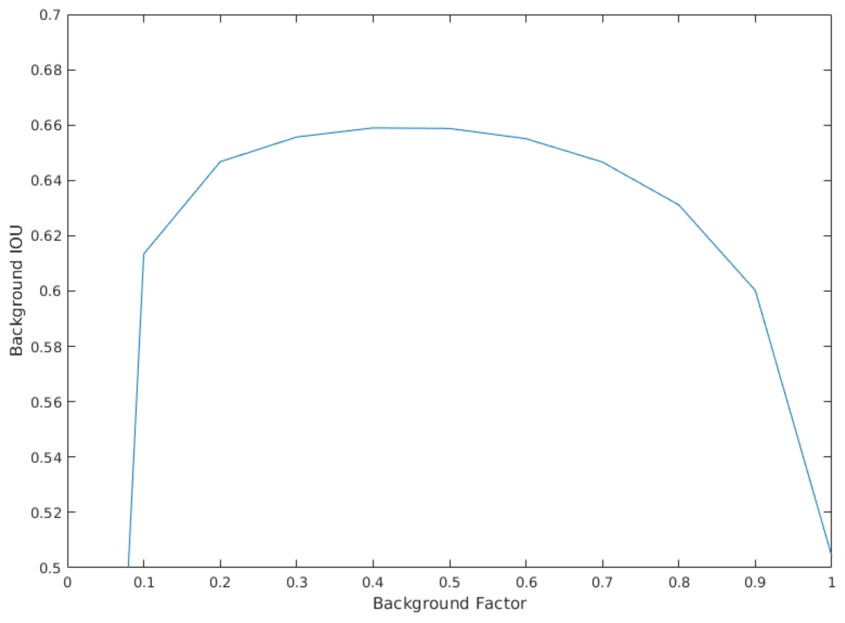

Figure 4.5: IOU for a range of background factors.

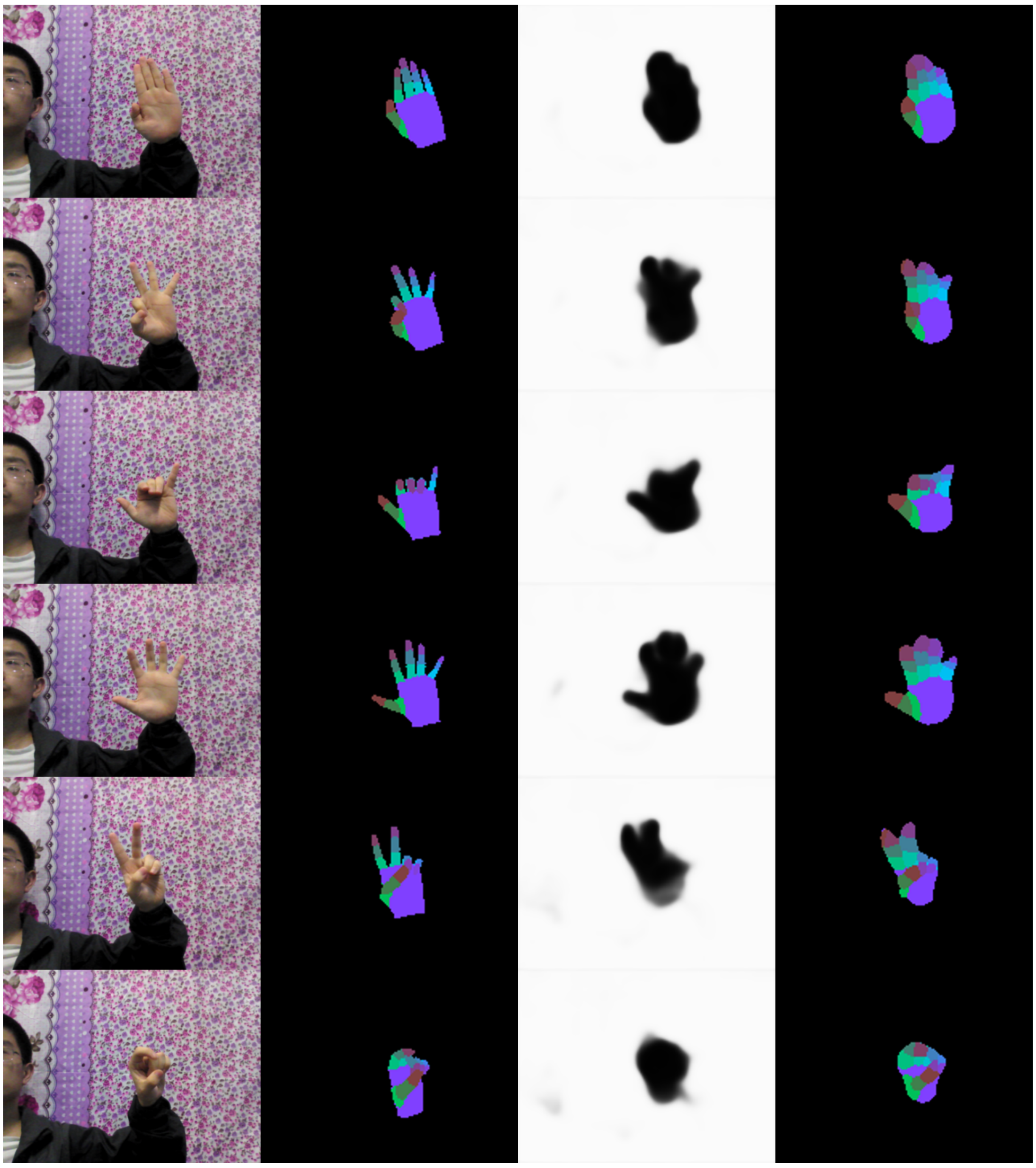
Figure 4.6: Qualitative segmentation results. The columns from left to right show the input RGB image, the estimation semantic ground truth, the output of the background network, and the final segmentation result.

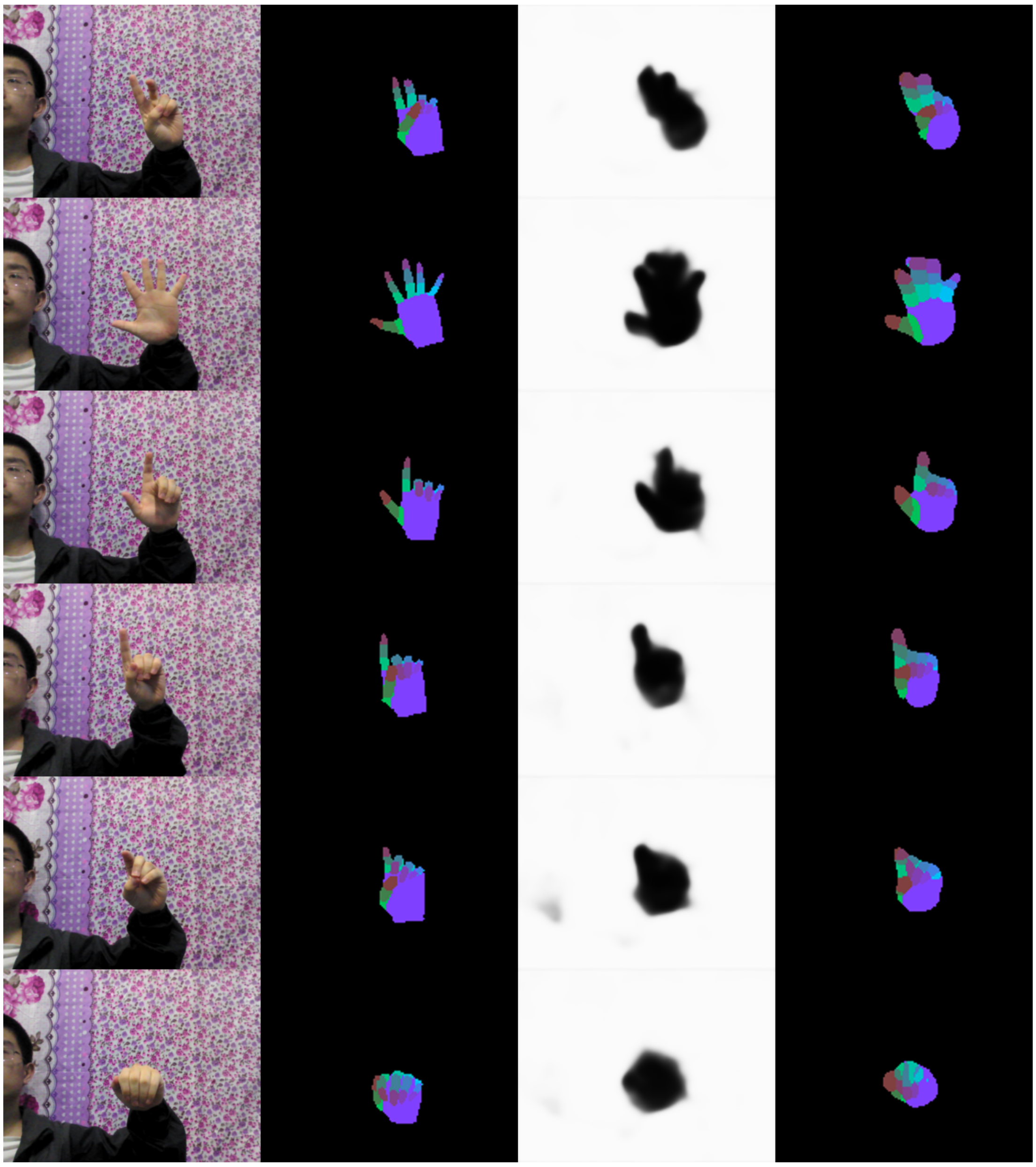
Figure 4.7: More qualitative segmentation results. The columns from left to right show the input RGB image, the estimation semantic ground truth, the output of the background network, and the final segmentation result.

ground truth keypoint locations. Table 4.3 and figure 4.9 show equivalent results without fingertip reinitialisation. Whereas DE-based algorithm is largely unaffected by this change, the accuracy of ICP-PSO decreases significantly.

Subject	DE	ICP-PSO
1	6.8	11.0
2	5.0	9.3
3	5.6	9.1
4	6.2	9.9
5	6.2	10.9
6	7.3	11.0

Table 4.2: Mean keypoint error in millimetres for each sequence in MRSA when tracking joint locations with fingertip reinitialisation.

Subject	DE	ICP-PSO
1	6.8	35.9
2	5.1	34.2
3	5.9	35.7
4	6.0	48.4
5	6.2	43.9
6	9.9	58.1

Table 4.3: Mean keypoint error in millimetres for each sequence in MRSA when tracking joint locations without fingertip reinitialisation.

Table 4.4 and figure 4.10 show results for the second case. This is the case in which ICP-PSO was intended to be used and it can be seen that both algorithms perform similarly well. Table 4.5 and figure 4.11 show equivalent results without reinitialisation. It can be seen that, though the accuracy of both algorithms decreases significantly, DE generally outperforms ICP-PSO.

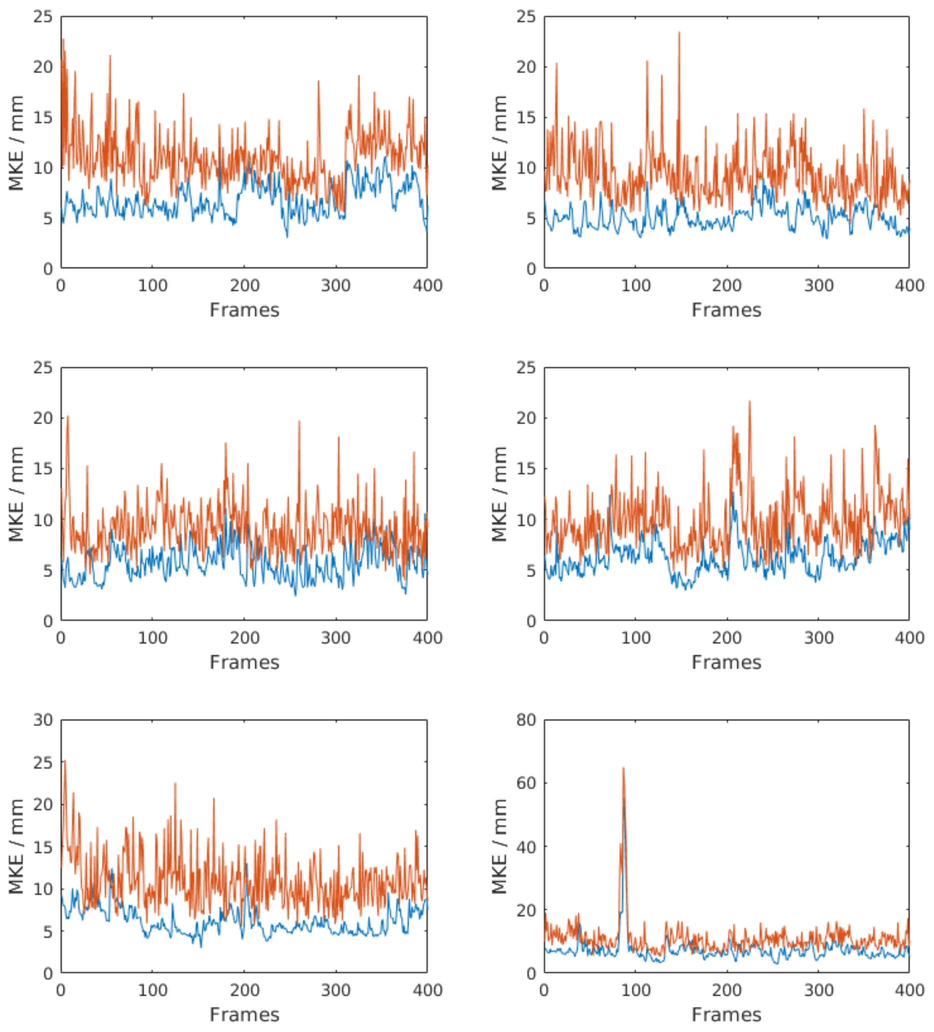

Figure 4.8: Mean keypoint error across each sequence in MRSA when tracking joint locations with fingertip reinitialisation. DE is in blue, ICP-PSO is in red. Subjects 1-6 are shown in order left-to-right top-to-bottom.

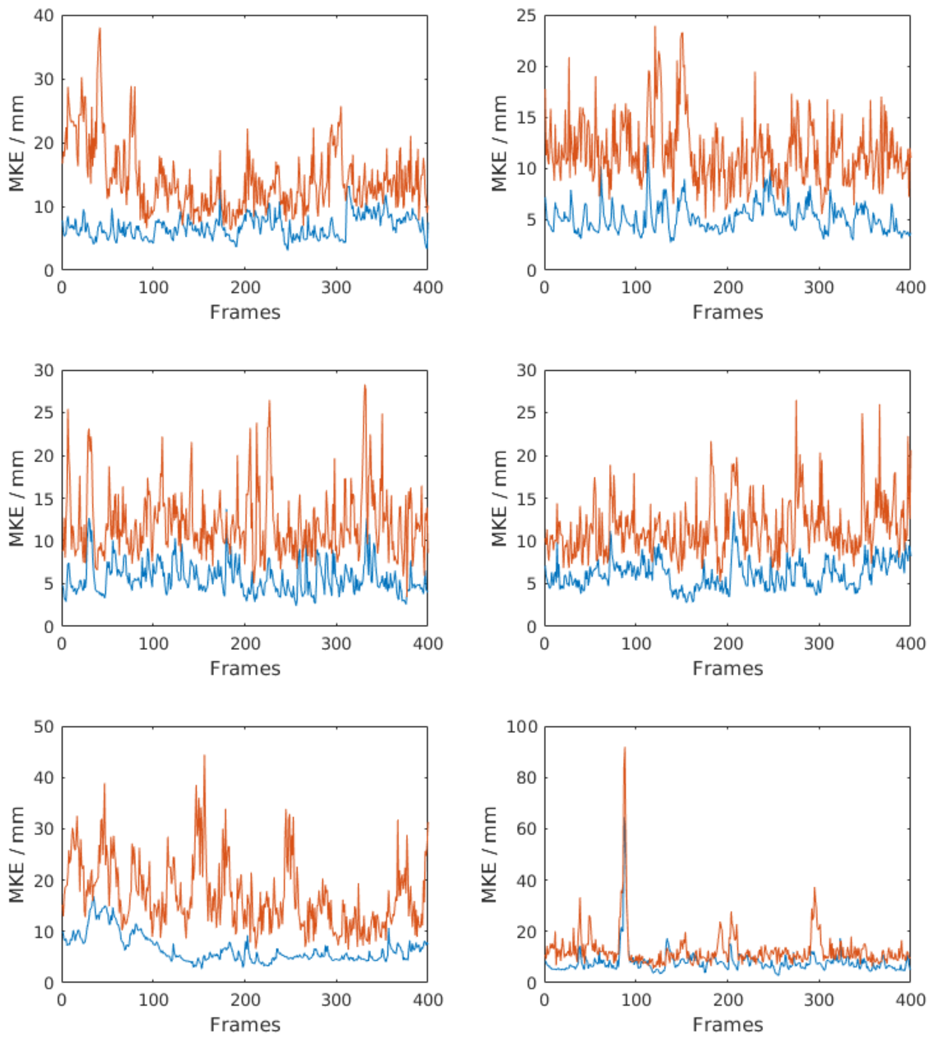

Figure 4.9: Mean keypoint error across each sequence in MRSA when tracking joint locations without fingertip reinitialisation. DE is in blue, ICP-PSO is in red. Subjects 1-6 are shown in order left-to-right top-to-bottom.

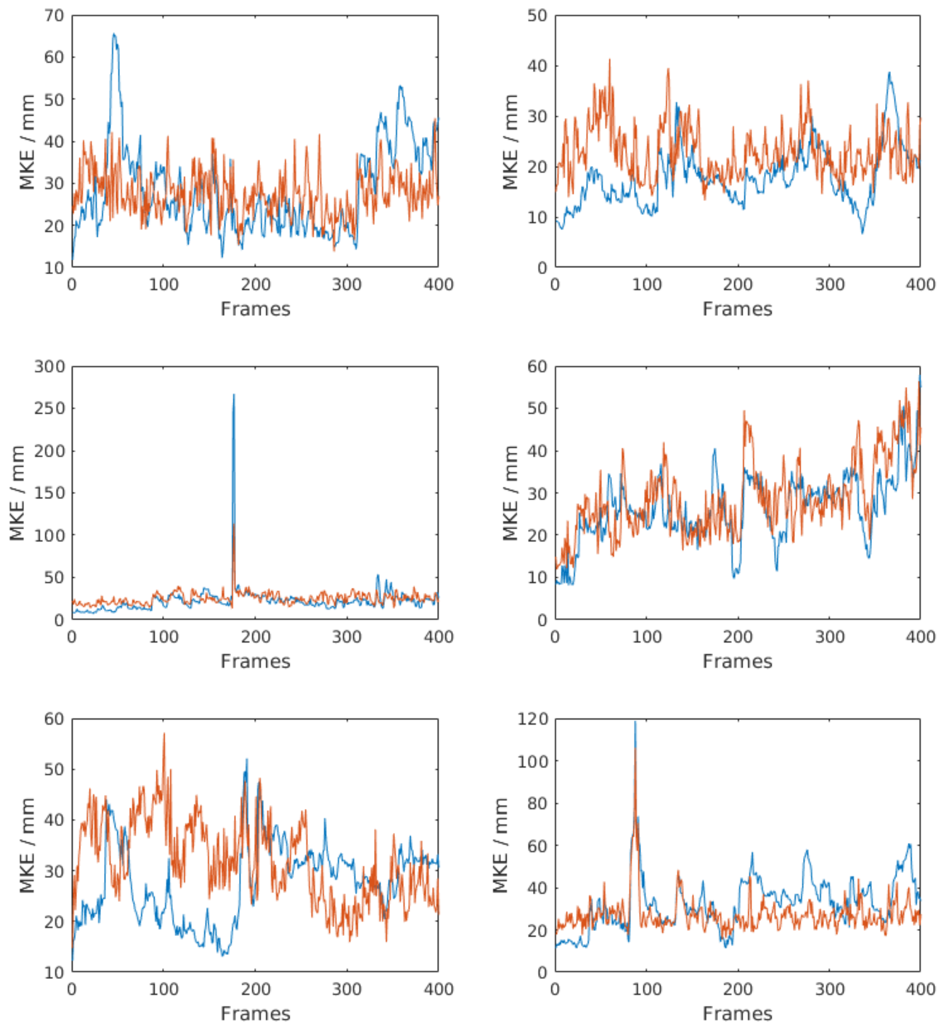

Figure 4.10: Mean keypoint error across each sequence in MRSA when tracking from depth with fingertip reinitialisation. DE is in blue, ICP-PSO is in red. Subjects 1-6 are shown in order left-to-right top-to-bottom.

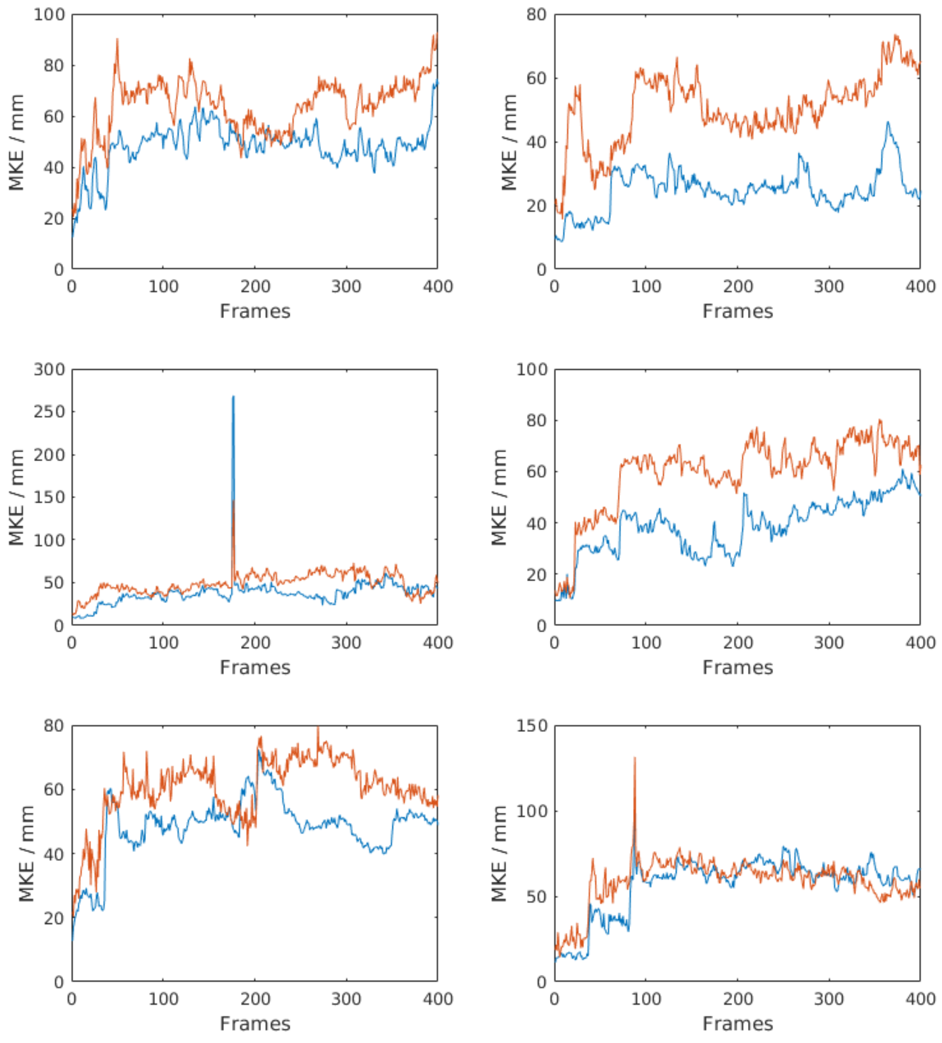

Figure 4.11: Mean keypoint error across each sequence in MRSA when tracking from depth without fingertip reinitialisation. DE is in blue, ICP-PSO is in red. Subjects 1-6 are shown in order left-to-right top-to-bottom.

Subject	DE	ICP-PSO
1	28.2	27.6
2	17.7	22.9
3	22.6	25.4
4	26.4	28.7
5	27.6	31.8
6	32.8	27.3

Table 4.4: Mean keypoint error in millimetres for each sequence in MRSA when tracking from depth with fingertip reinitialisation.

Subject	DE	ICP-PSO
1	51.2	57.2
2	25.4	51.3
3	39.1	54.9
4	36.5	53.7
5	46.3	59.0
6	52.3	53.5

Table 4.5: Mean keypoint error in millimetres for each sequence in MRSA when tracking from depth without fingertip reinitialisation.

4.4 Semantic Fitting

In this section, the full algorithm will be evaluated on the STB dataset. All results are based on the semantic segmentation results reported in section 4.2. All quantitative results are in millimetres and represent the average of five repeats of the relevant experiment.

4.4.1 Choice of cost function

As mentioned in section 3.3, there are several different cost functions that may be used with the tracking algorithm. The cost functions under consideration are negative-log-likelihood (NLL), sum-of-complements (SoC), and number of incorrect pixels (IP). All three were compared using 16 agents and 50 generations per frame, and with three different thresholds (t=0.1, 0.3, and 0.5) used with the IP cost function.

Tables 4.6 and 4.7 show the relative keypoint errors and root errors respectively for each cost function and sequence. Figures 4.12-4.15 show how the same quantities varying over the course of each sequence. It can be seen that IP with too high a threshold does little better than random chance, as it would with a threshold close to zero. NLL also tends to perform poorly compared to the others. SoC and IP with $t = 0.3$ perform similarly, with SoC being slightly more robust due to its taking more information into account. IP with $t = 0.3$ is better at localising in some cases due to its enforcing a strict boundary around the hand region, but also appears to be more fragile. For this reason, SoC will be used in all following experiments.

Cost function	B1Counting	B1Random
NLL	37.1	33.0
SoC	35.2	31.6
IP, t=0.1	39.1	31.1
IP, t=0.3	32.0	39.5
IP, t=0.5	76.7	70.3

Table 4.6: Mean relative keypoint error in millimetres for each sequence in the STB testing set for each cost function.

Cost function	B1Counting	B1Random
NLL	43.2	47.0
SoC	34.8	43.4
IP, t=0.1	43.3	46.8
IP, t=0.3	31.5	64.7
IP, t=0.5	160.0	145.6

Table 4.7: Mean root error in millimetres across each sequence in the STB testing set for each cost function.

4.4.2 Hyperparameters

The tracking algorithm has two significant hyperparameters. The first is the number of agents in the population. The second is the number of generations of DE performed on each frame. In this section, the impact of these hyperparameters on the performance of the tracker will be assessed.

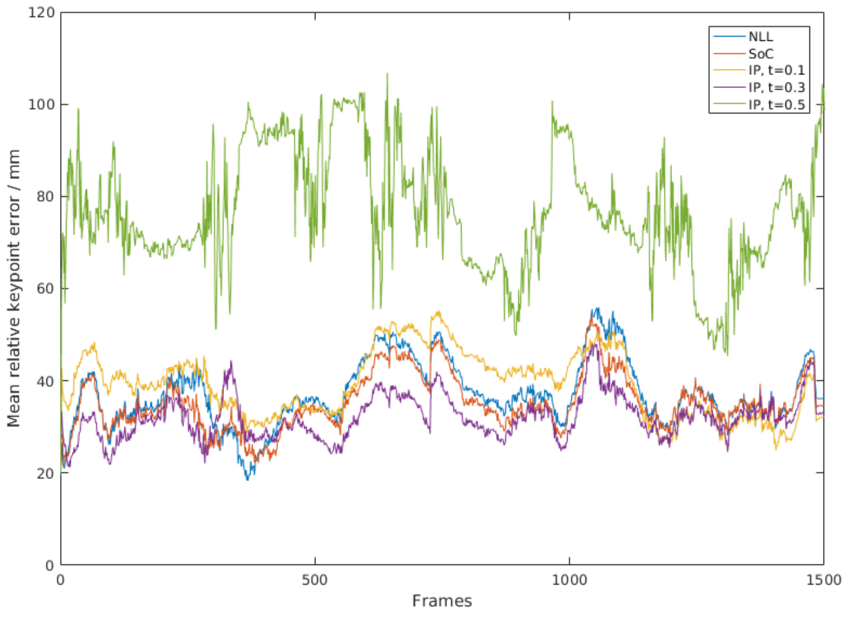

Figure 4.12: Mean relative keypoint error for “B1Counting” for each cost function.

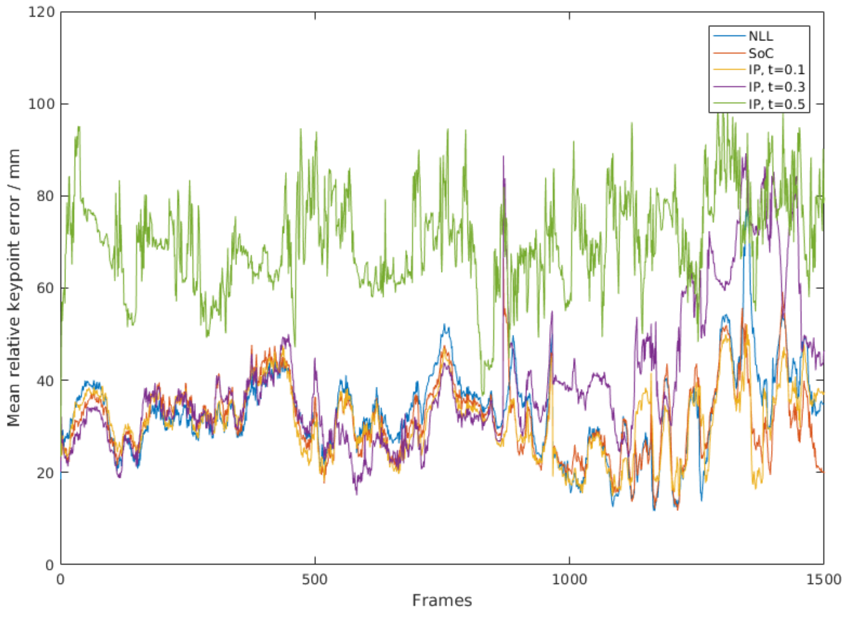

Figure 4.13: Mean relative keypoint error for “B1Random” for each cost function.

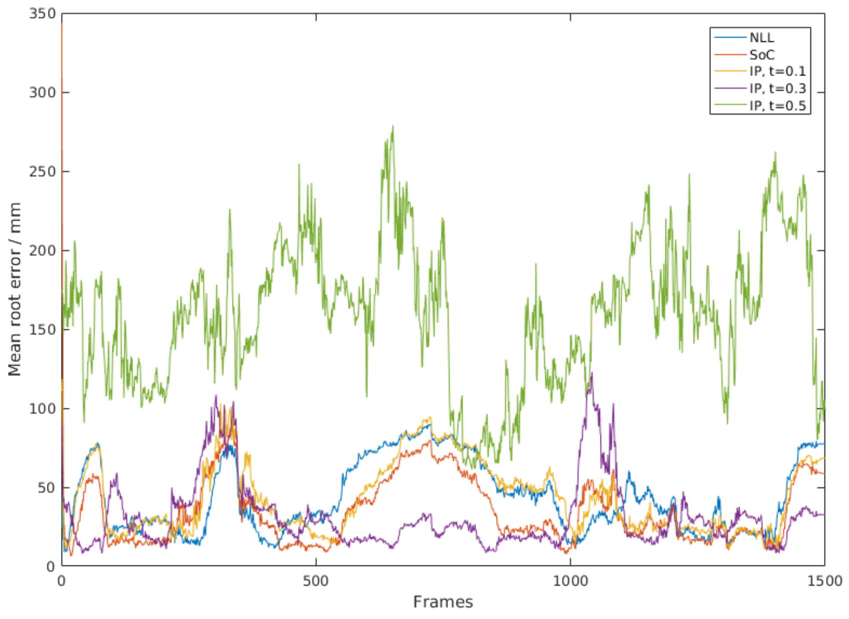

Figure 4.14: Mean root error for “B1Counting” for each cost function.

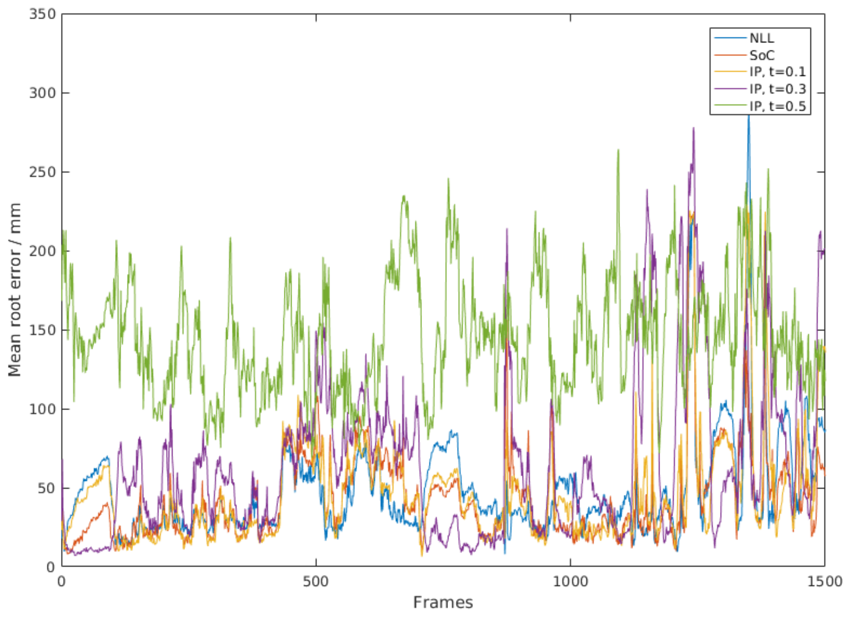

Figure 4.15: Mean root error for “B1Random” for each cost function.

To assess the impact of the number of agents in the population, the algorithm was run with 8, 16, 32, and 64 agents. In each case, 50 generations of DE were performed per frame. The results are shown in tables 4.8 and 4.9 and figures 4.16-4.19. It can be seen that the number of agents does not generally have a substantial impact on the performance of the algorithm. With smallest population size of 8, the algorithm is slightly more accurate in some cases but also takes longer to recover from tracking failures.

Agents	Counting	Random
8	34.1	30.3
16	34.7	32.3
32	35.3	33.8
64	36.0	33.8

Table 4.8: Mean relative keypoint error in millimetres across each sequence in the STB testing set for a range of population sizes.

Agents	Counting	Random
8	37.2	52.5
16	33.6	46.2
32	36.1	44.4
64	37.2	44.2

Table 4.9: Mean root error in millimetres across each sequence in the STB testing set for a range of population sizes.

The algorithm was also run with 10, 50, 100, and 200 generations per frame. A population of 16 agents was used in each case. Similar to the population size, the number of generations per frame only has a significant effect on the accuracy of the algorithm when it is small, with the accuracy being improved in the general case, but also more susceptible to tracking failures. This is due to the population being less able to adapt any particular frame, which improves its ability to adapt to each new frame when it is close to the solution, but also makes it harder to recover when it is not.

In subsequent experiments, 16 agents and 50 generations per frame will be used, since there is no advantage to using larger values.

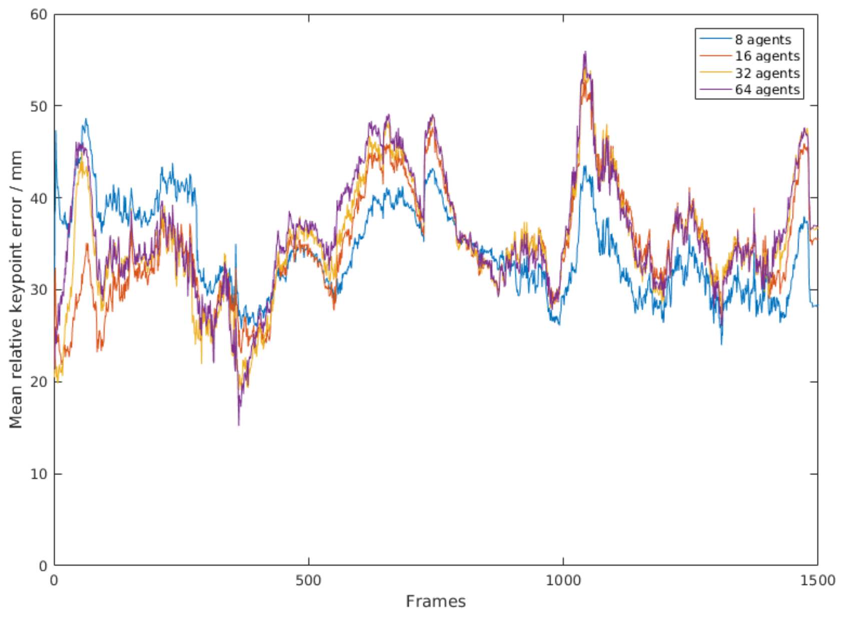

Figure 4.16: Mean relative keypoint error for “B1Counting” for a range of population sizes.

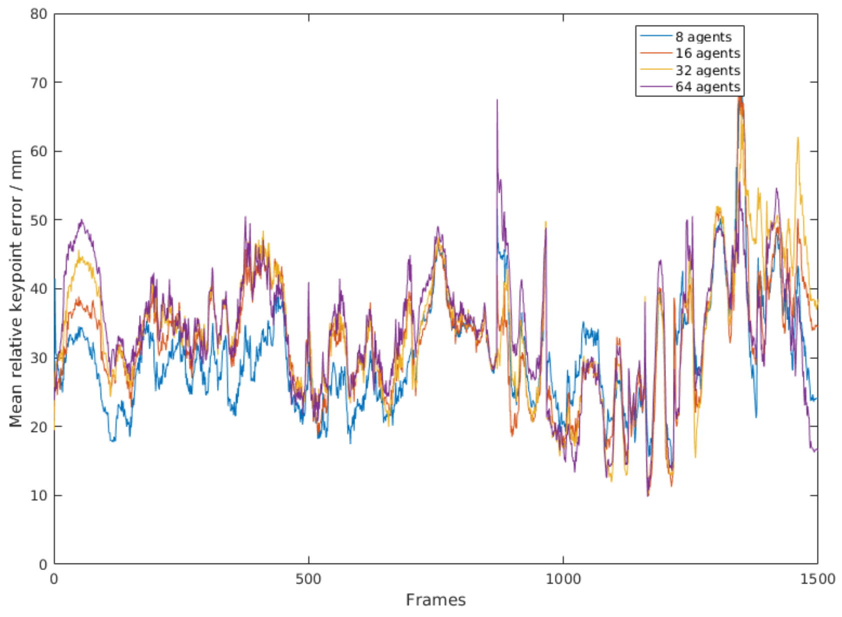

Figure 4.17: Mean relative keypoint error for “B1Random” for a range of population sizes.

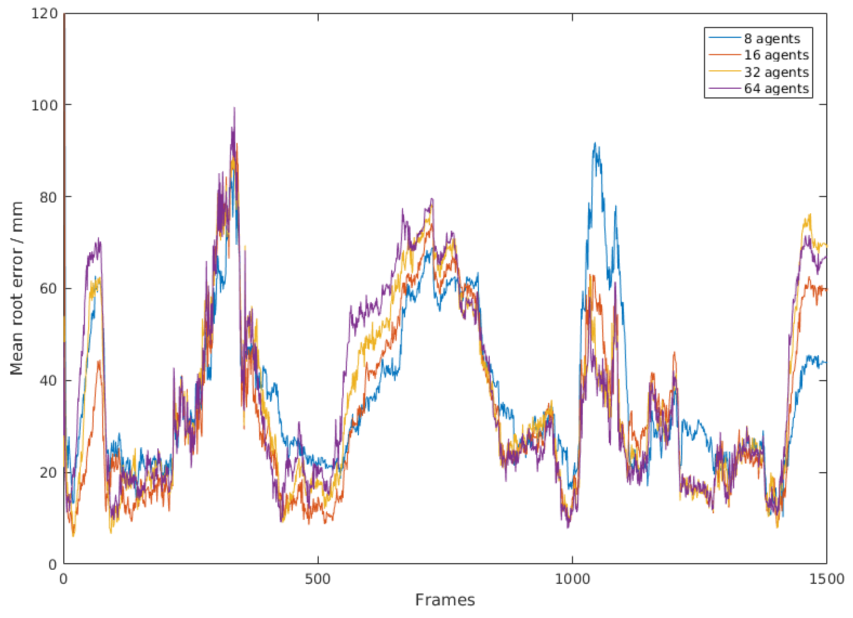

Figure 4.18: Mean root error for “B1Counting” for a range of population sizes.

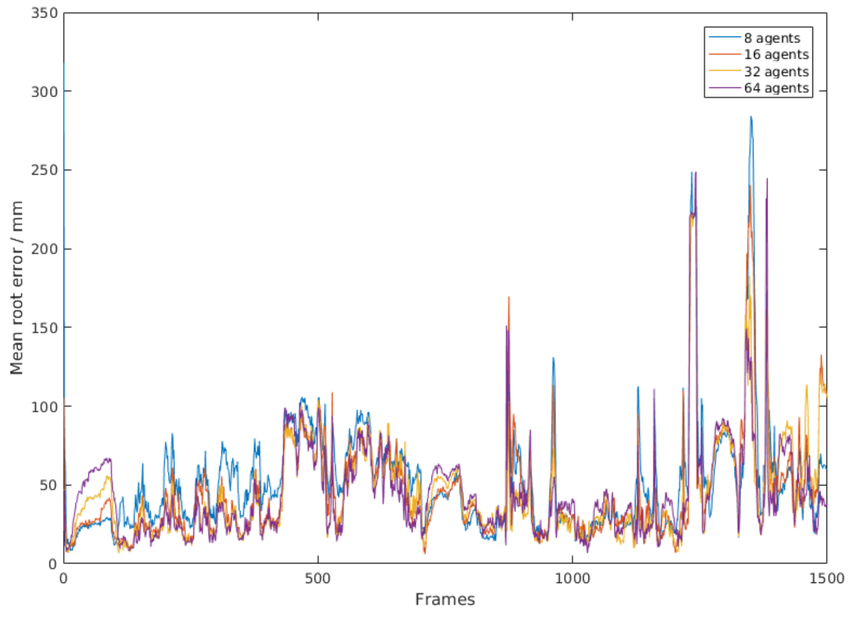

Figure 4.19: Mean root error for “B1Random” for a range of population sizes.

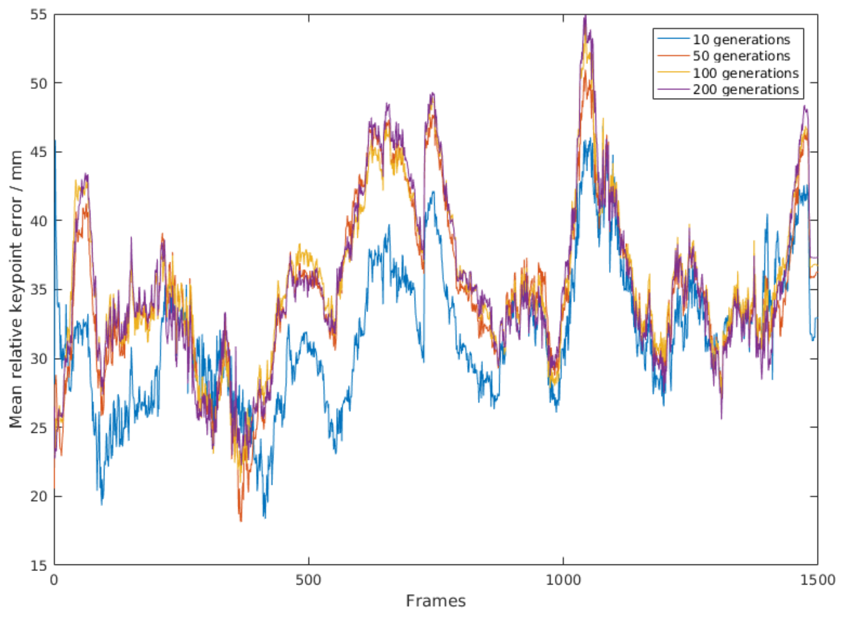

Figure 4.20: Mean relative keypoint error for "B1Counting" for a range of generations per frame.

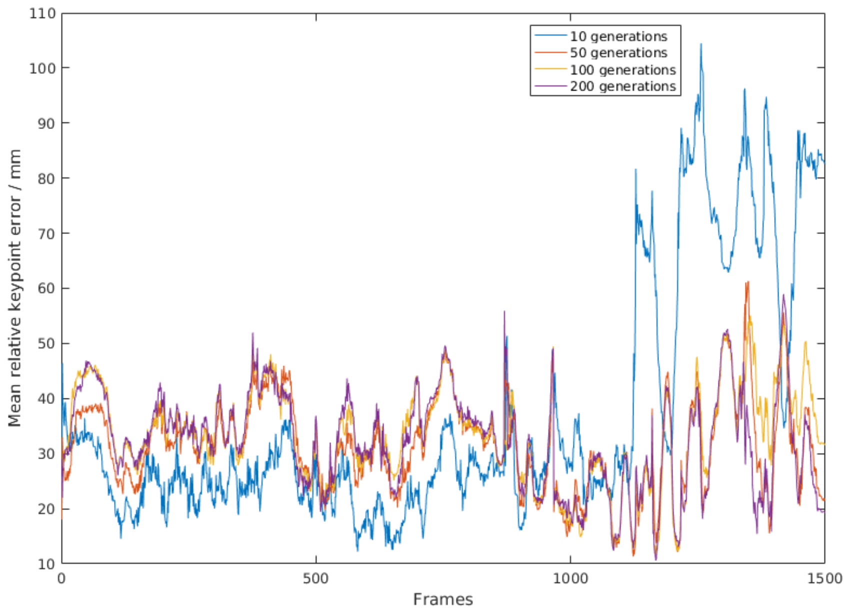

Figure 4.21: Mean relative keypoint error for "B1Random" for a range of generations per frame.

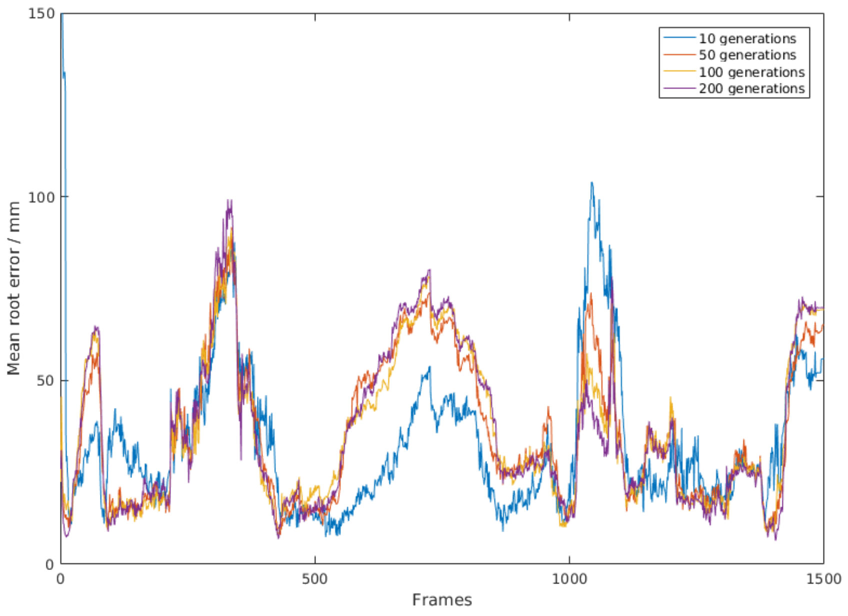

Figure 4.22: Mean root error for “B1Counting” for a range of generations per frame.

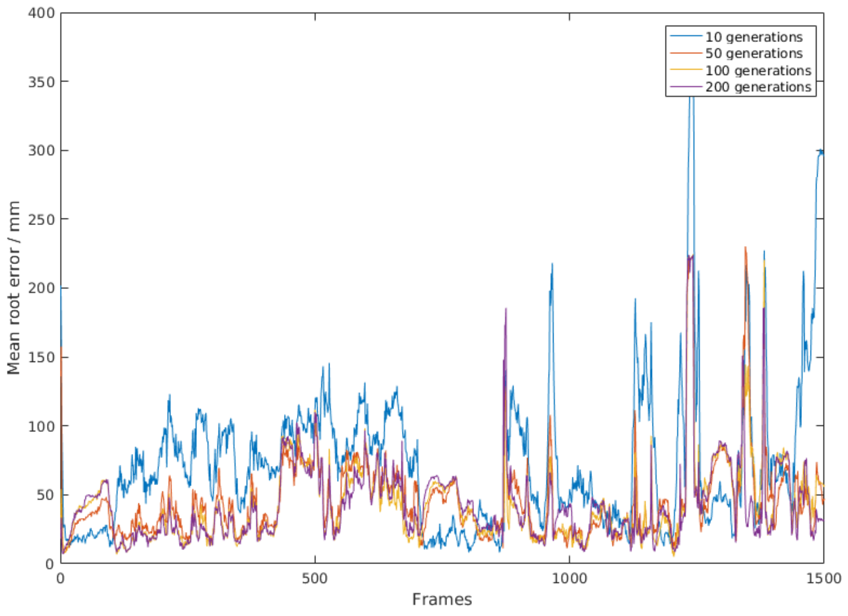

Figure 4.23: Mean root error for “B1Random” for a range of generations per frame.

Generations	Counting	Random
10	31.8	37.0
50	35.0	31.4
100	35.6	33.3
200	35.4	32.4

Table 4.10: Mean relative keypoint error in millimetres across each sequence in the STB testing set for a range of generations per frame.

Generations	Counting	Random
10	33.7	74.4
50	36.1	45.7
100	36.1	42.8
200	36.3	42.3

Table 4.11: Mean root error in millimetres across each sequence in the STB testing set for a range of generations per frame.

4.4.3 Comparison with PSO

To verify that DE is an appropriate algorithm for this task, an equivalent algorithm based on PSO was implemented. PSO is a commonly used algorithm in depth-based hand tracking and one of only a few that are applicable to this problem, as it is gradient-free and does not the problem to be reducible to one of matching point clouds. Since PSO well known to be highly dependent on initial conditions, the first frame of each sequence was initialised by matching the model to the ground truth keypoint locations using DE, and the populations were initialised by randomly perturbing the result of this optimisation.

The results from both the DE and PSO-based algorithms are shown in table 4.12 and figures 4.24-4.27. It can be seen that DE is generally more accurate, particularly in localising the hand. PSO generally fails to search the space efficiently, with subsequent frames failing to improve on the result from a previous frame in many cases. Also shown are tracking results when the semantic ground truth used as the input for the tracking algorithm. DE also does better in this case. Apart from a severe tracking failure in the second sequence from which the algorithm quickly recovers, using the ground truth input leads to results that are high quality in terms of both gesture recovery and hand localisation.

	Counting	Random
DE estimate	34.9	32.7
DE ground	16.1	24.0
PSO estimate	71.5	93.7
PSO ground	93.7	106.7

Table 4.12: Mean relative keypoint error in millimetres across each sequence in the STB testing set for DE and PSO-based tracking.

4.4.4 Impact of global orientation

The main source of error in the tracking results is ambiguity in global orientation. To demonstrate this, an orientation oracle was determined for each frame by fitting the model to the ground truth keypoint locations. The algorithm was then run with the agents fixed to this orientation. Figures 4.28 and 4.29 show the effect of doing so on the qualitative results. It can be seen that, without the oracle, the model tends to be in a position that covers the semantic regions, but with the wrong orientation, which puts the 3D joint locations out of position by a few centimetres. This is shown quantitatively in table 4.13 and figures 4.30-4.33. The effect is most striking on the relative keypoint error, which improves considerably when the oracle is accessed. The localisation error tends to increase when the oracle is accessed due to the 3D position of the hand being more sensitive to the input noise when the orientation is fixed.

	Counting	Random
Free	34.9	32.7
Oracle	11.8	10.7

Table 4.13: Mean relative keypoint error in millimetres across each sequence in the STB testing set for the tracking algorithm with and without the global orientation oracle.

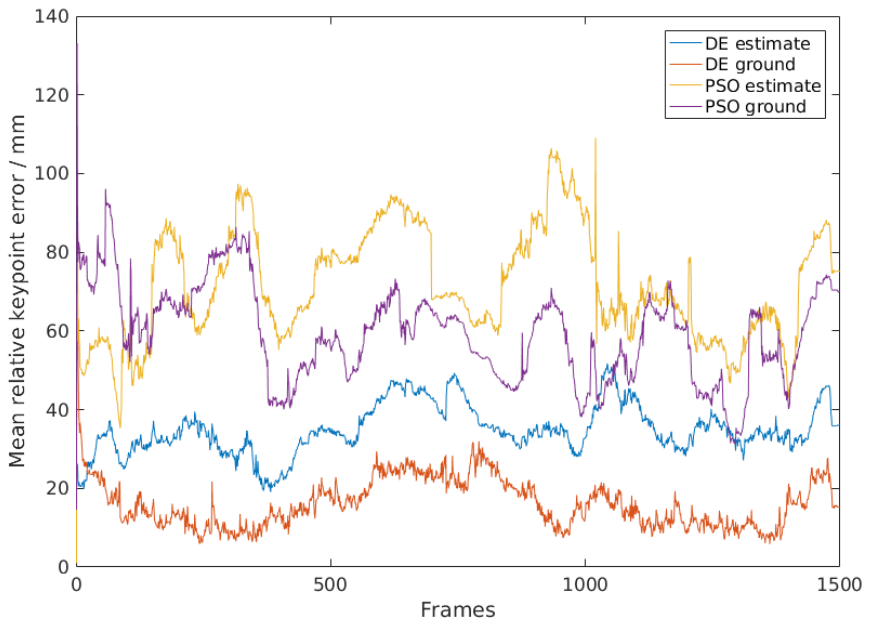

Figure 4.24: Mean relative keypoint error for “B1Counting” for DE and PSO-based tracking.

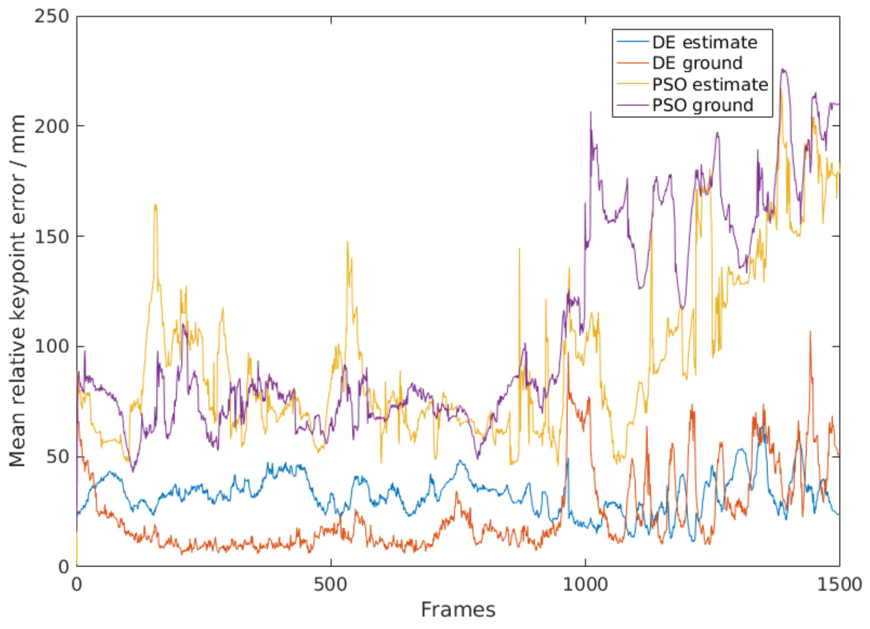

Figure 4.25: Mean relative keypoint error for “B1Random” for DE and PSO-based tracking.

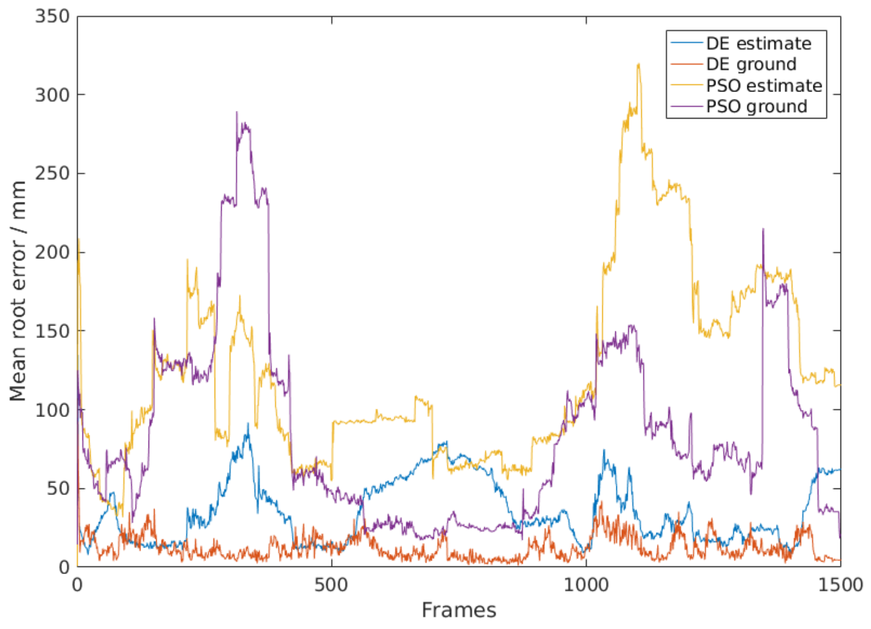

Figure 4.26: Mean root error for "B1Counting" for DE and PSO-based tracking.

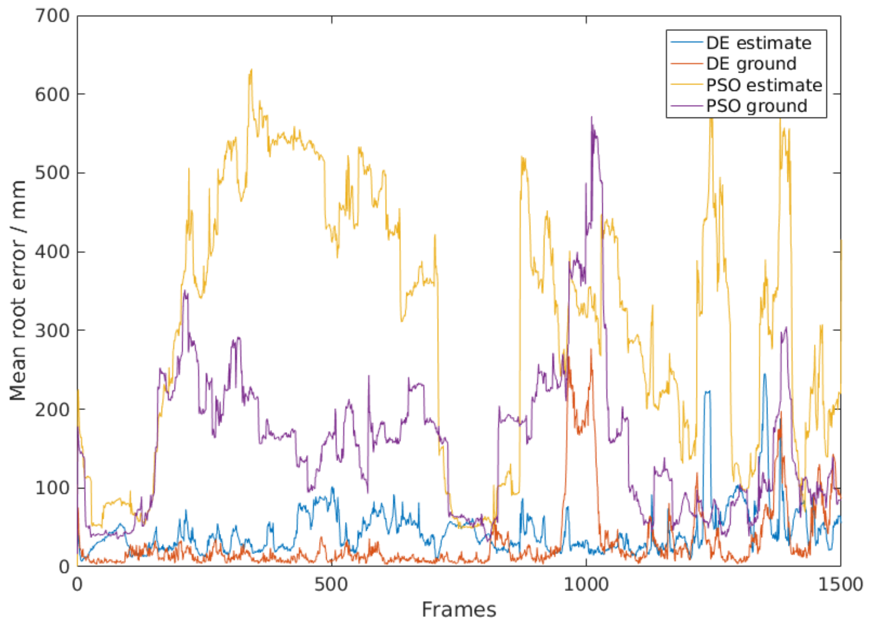

Figure 4.27: Mean root error for "B1Random" for DE and PSO-based tracking.

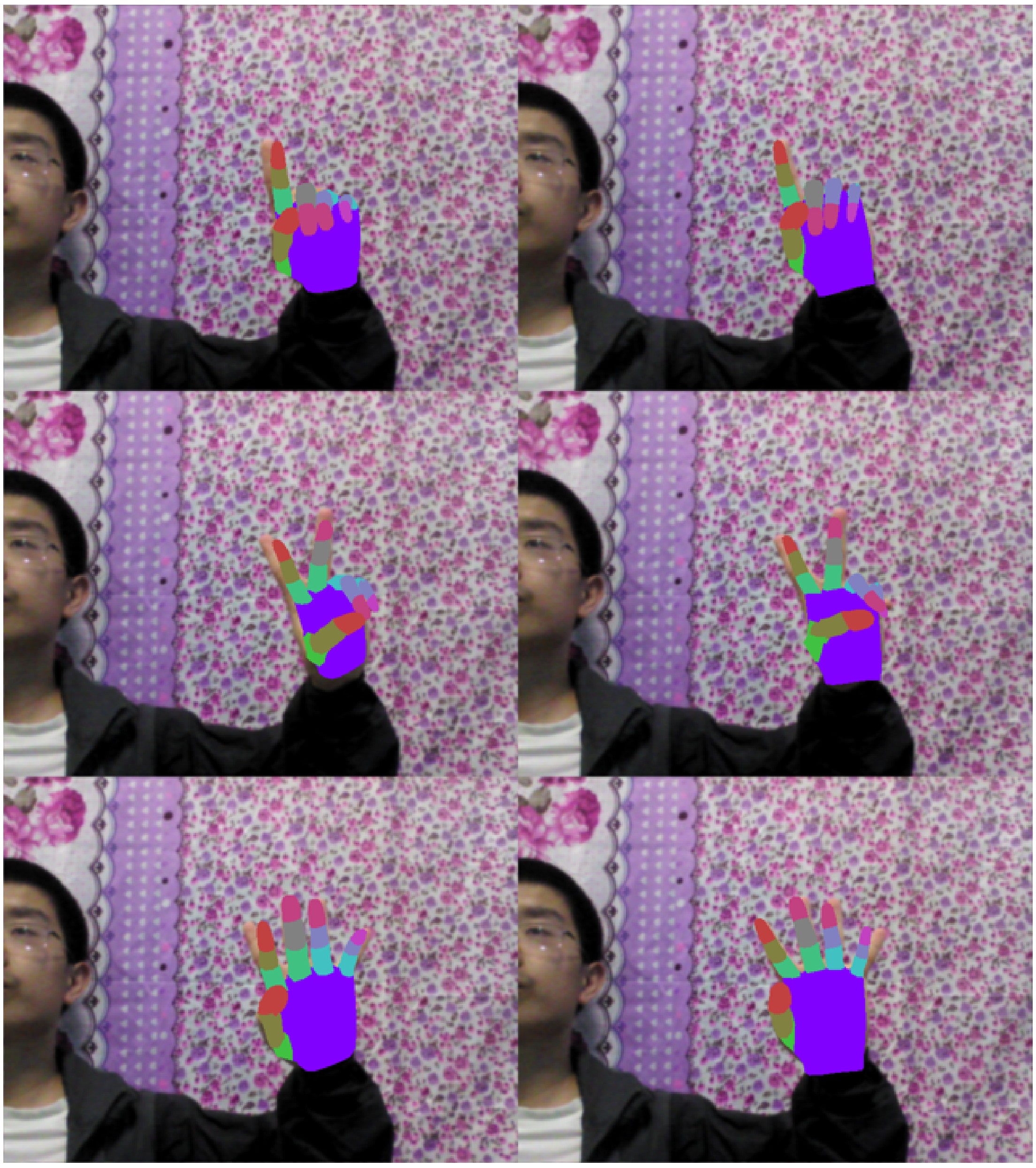

Figure 4.28: Qualitative examples showing the effect of the global orientation oracle on the final results. The left and right columns show results attained without and with the oracle respectively.

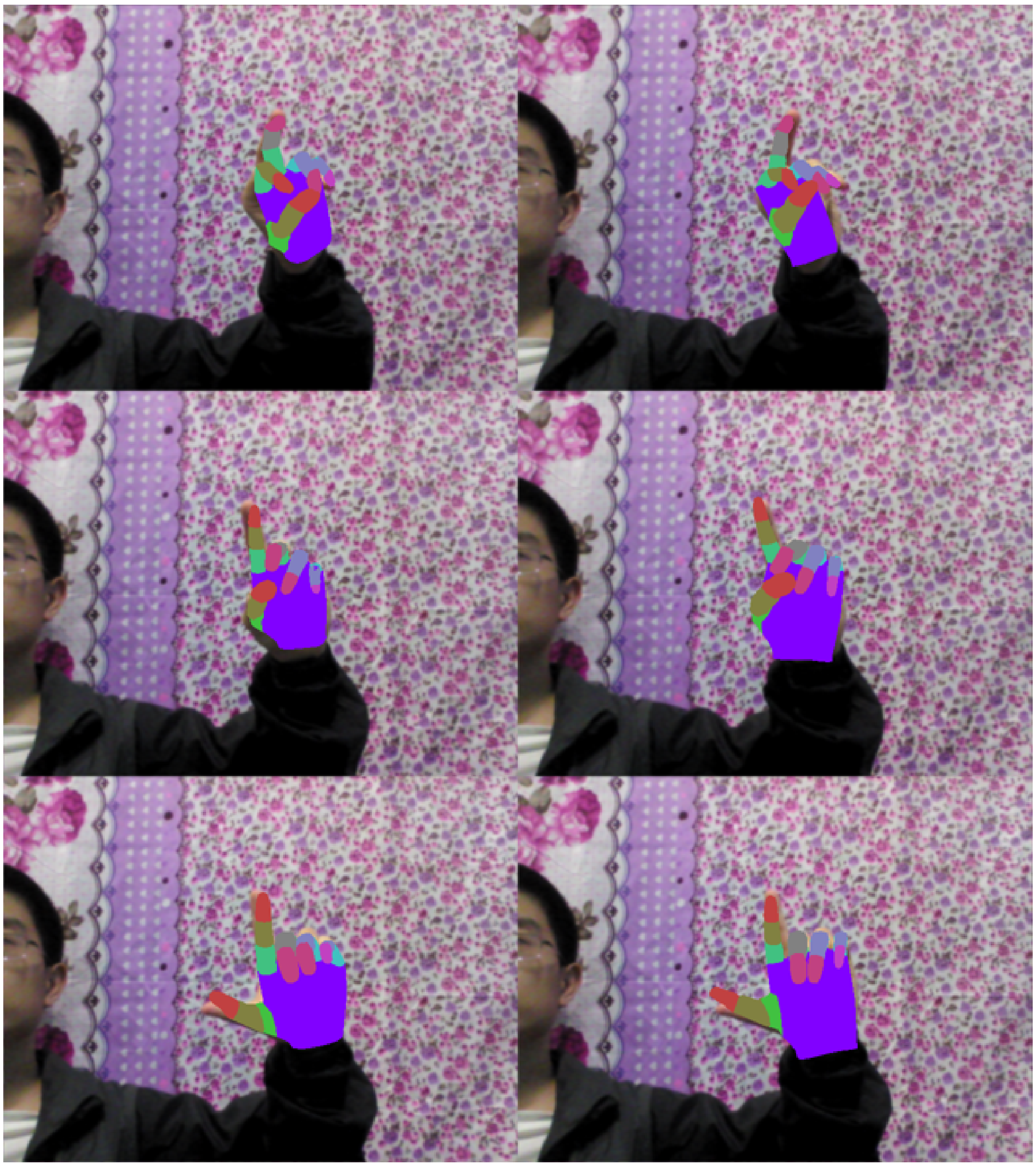

Figure 4.29: Qualitative examples showing the effect of the global orientation oracle on the final results. The left and right columns show results attained without and with the oracle respectively.

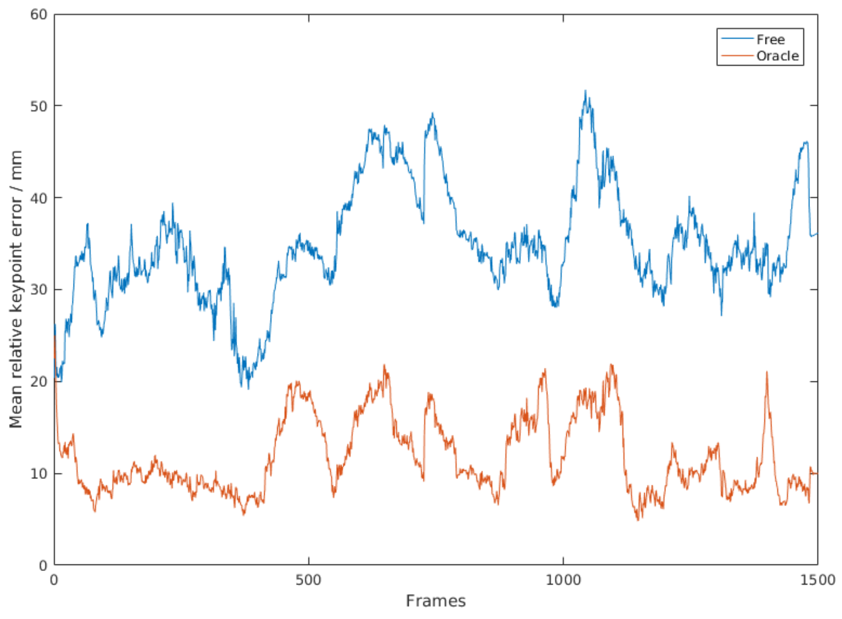

Figure 4.30: Mean relative keypoint error for “B1Counting” for the tracking algorithm with and without the global orientation oracle.

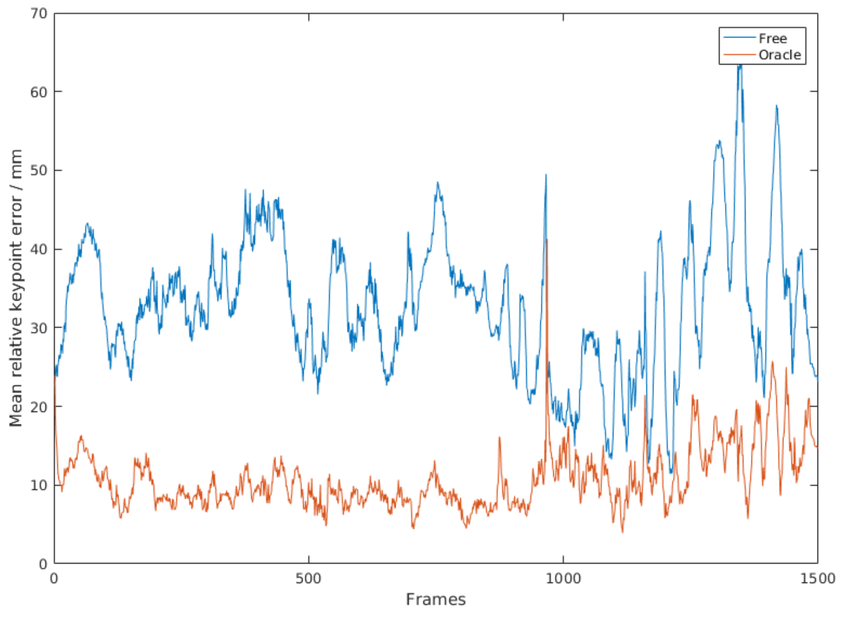

Figure 4.31: Mean relative keypoint error for “B1Random” for the tracking algorithm with and without the global orientation oracle.

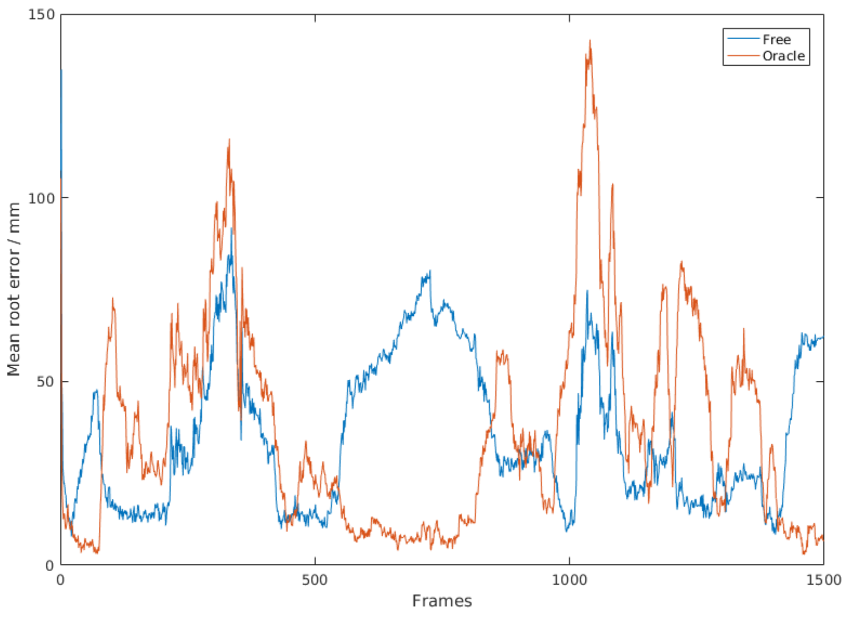

Figure 4.32: Mean root error for “B1Counting” for the tracking algorithm with and without the global orientation oracle.

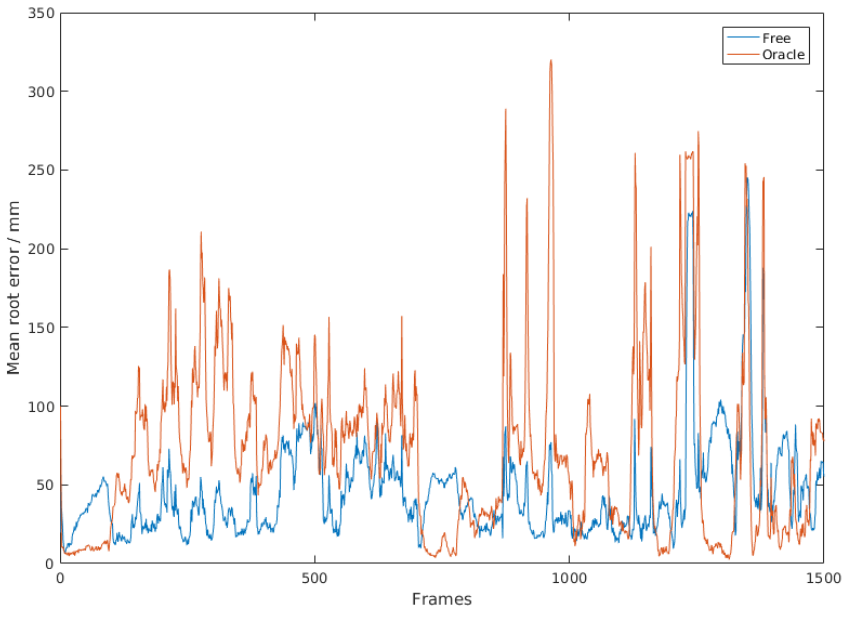

Figure 4.33: Mean root error for “B1Random” for the tracking algorithm with and without the global orientation oracle.

4.4.5 Tracking from stereo pairs

One potential way to reduce the ambiguity in global orientation is to use stereo pairs, which are provided in the STB dataset. When using stereo pairs, the tracking algorithm remains the same but the model is rendered in the frame of both the left and right image and the combined cost function is minimised. The calculation of the stereo cost function is illustrated in figure 4.34.

The background estimator is not used in the stereo case, since the triangulation of the parts labels is sufficient to segment the hand implicitly. The results are shown in tables 4.14 and 4.15 and figures 4.35-4.38. It can be seen that the localisation of the hand is substantially improved by the addition of stereo information. The gesture recognition results are mixed, with some cases improving and others worsening. In general, it seems that short baseline stereo is sufficient to localise the hand but does not provide enough spatial information to improve the gesture recognition.

	Counting	Random
Mono	34.9	32.7
Stereo	30.7	41.3

Table 4.14: Mean relative keypoint error in millimetres across each sequence in the STB testing set for mono and stereo tracking.

	Counting	Random
Mono	35.6	44.7
Stereo	22.1	32.3

Table 4.15: Mean root error in millimetres across each sequence in the STB testing set mono and stereo tracking.

4.4.6 Comparison with state-of-the-art

Figure 4.39 shows a comparison of the proposed approach with algorithms from the literature, including several that reconstruct the hand pose from RGB. The graph shows

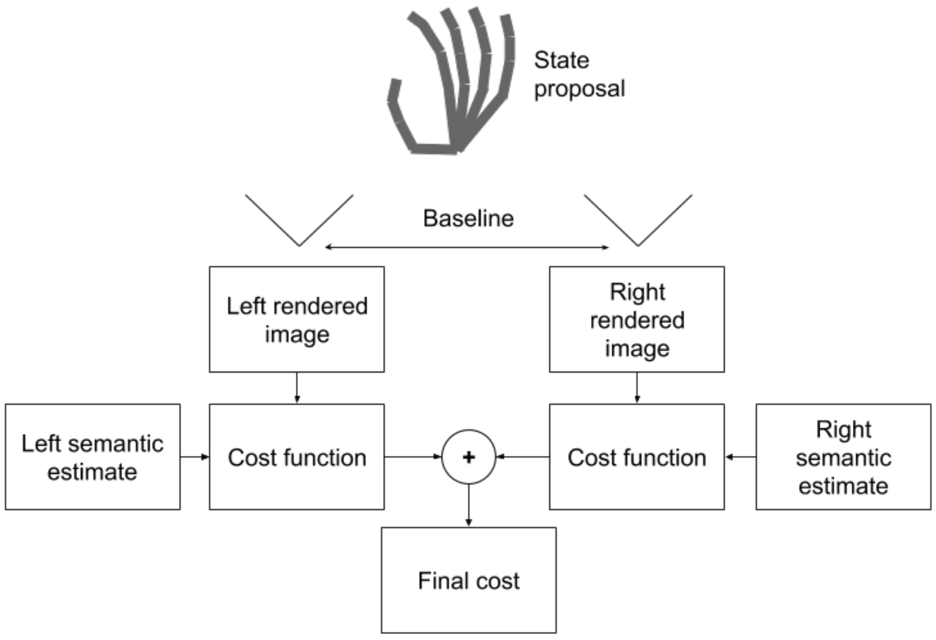

Figure 4.34: Calculation of the cost when using stereo pairs.

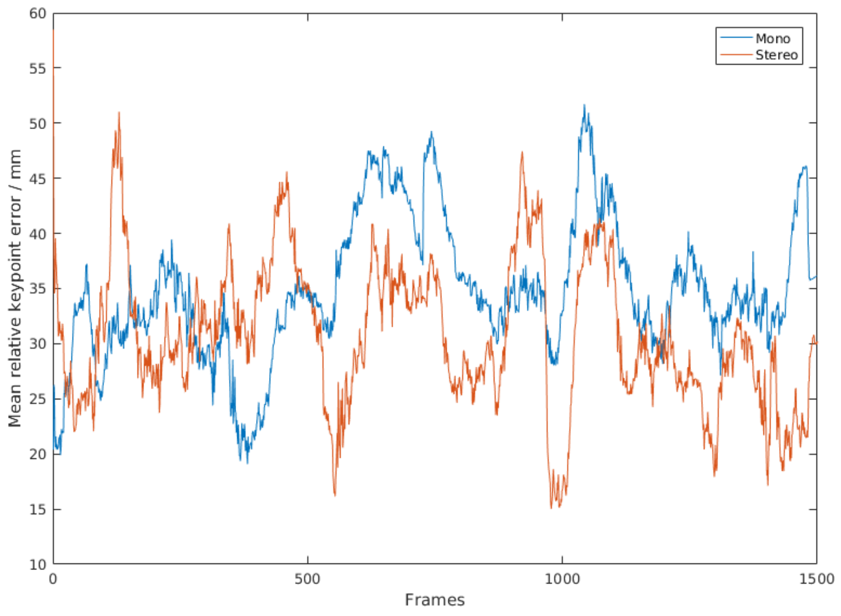

Figure 4.35: Mean relative keypoint error for “B1Counting” for mono and stereo tracking.

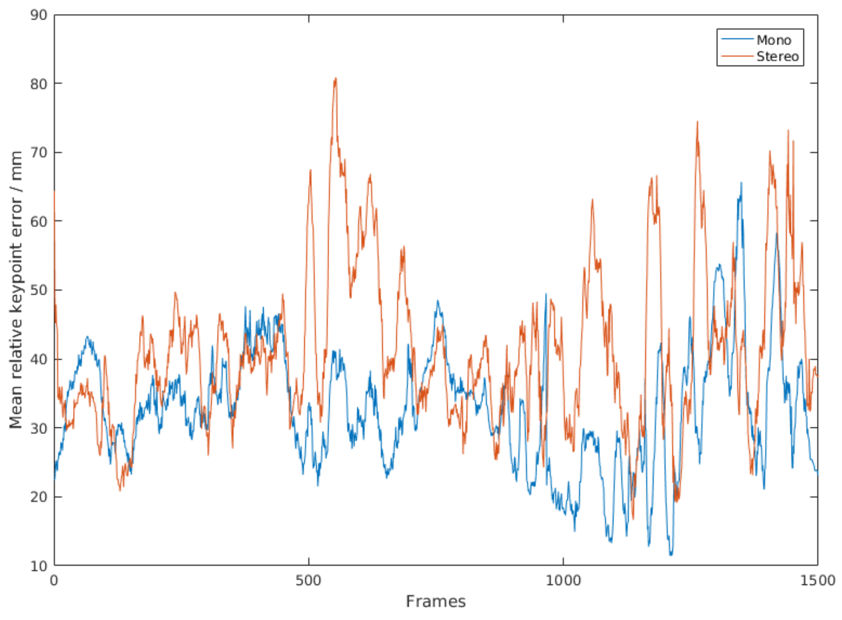

Figure 4.36: Mean relative keypoint error for “B1Random” for mono and stereo tracking.

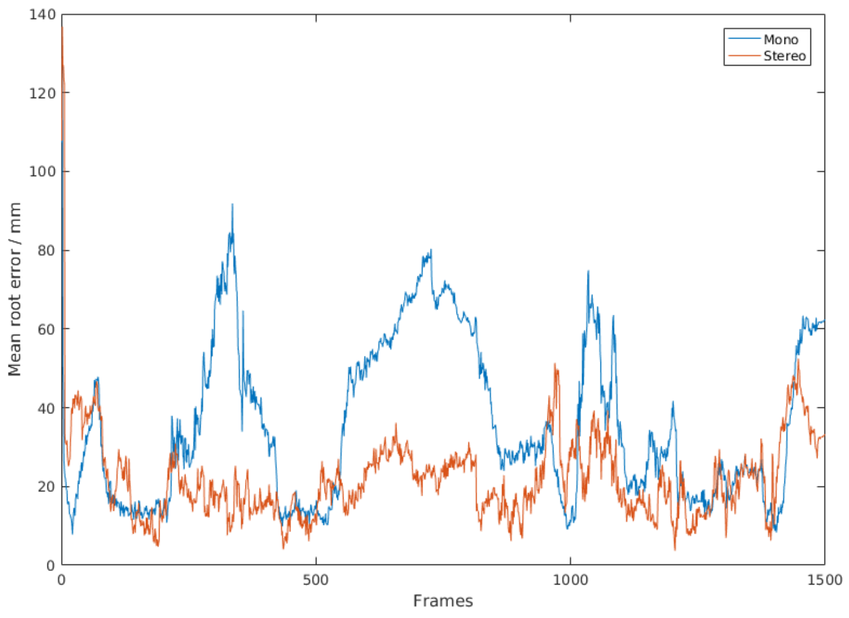

Figure 4.37: Mean root error for “B1Counting” for mono and stereo tracking.

the percentage correct keypoint (PCK) curves for a number of algorithms. PCK curves show on y-axis the percentage of keypoints that fall inside the error threshold denoted by the value on the x-axis when the position of the palms are aligned. It can be seen that the proposed method falls short of the contemporary state-of-the-art in the general case. Accessing the oracle global orientation brings the results into line with the rest of the state-of-art, as does using the ground truth semantic input.

Qualitative results for the whole system applied to the STB testing sequences are shown in 4.40 and 4.41. It can be seen that, with a few exceptions (such as the example on the bottom row of figure 4.40), the algorithm generally finds poses for which the hand model covers the appropriate region of the semantic image.

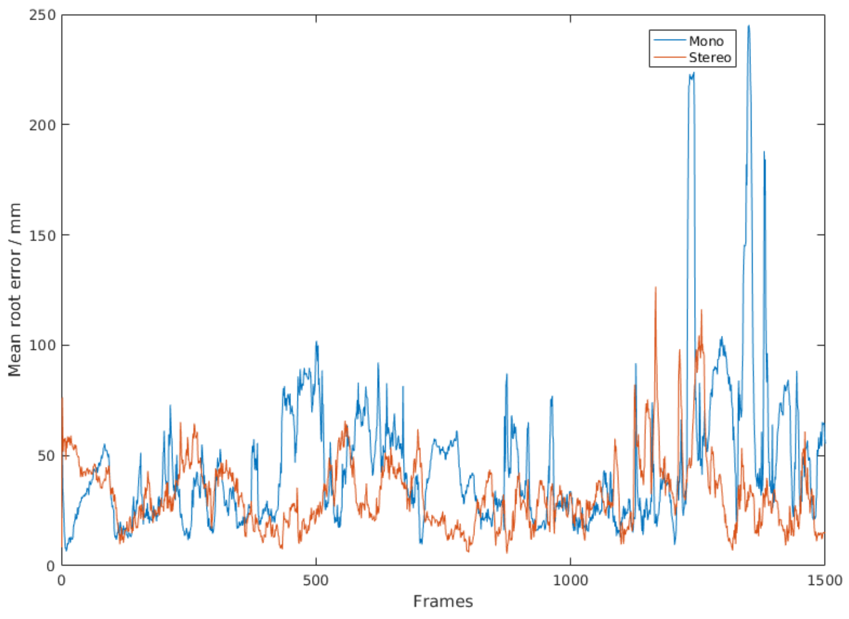

Figure 4.38: Mean root error for “B1Random” for mono and stereo tracking.

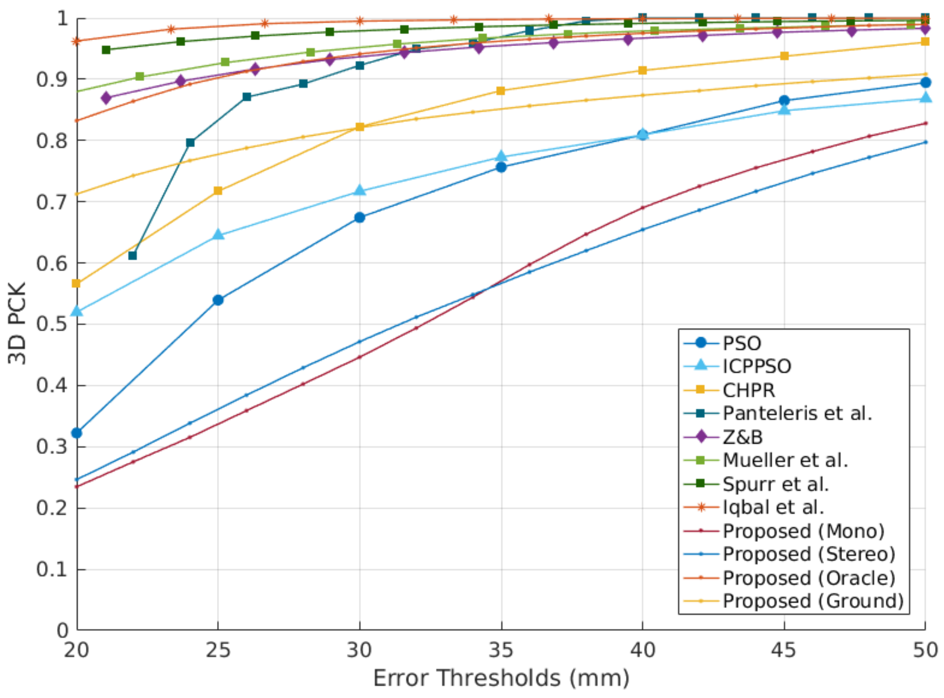

Figure 4.39: PCK curves for comparison with the current state-of-the-art on STB.

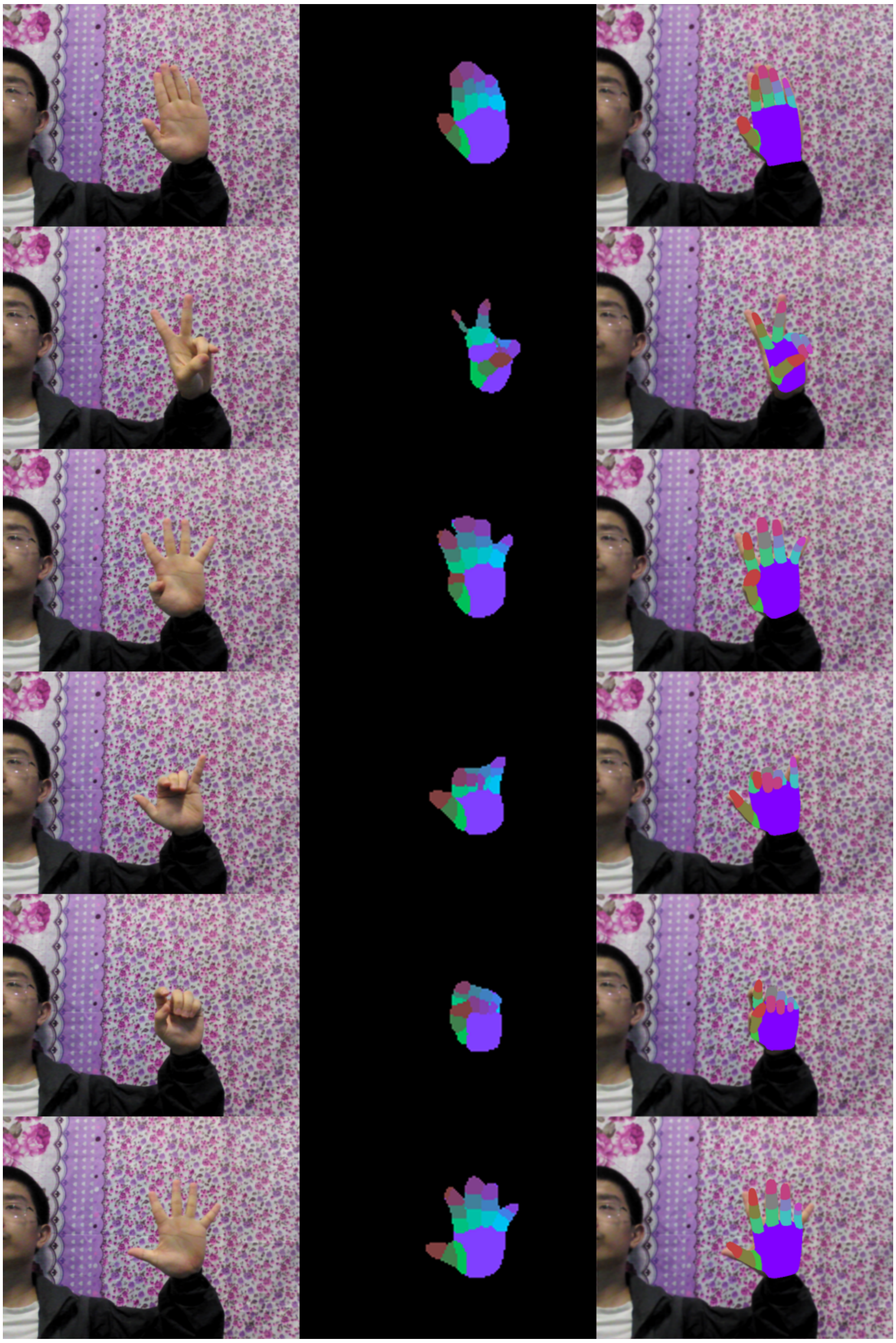

Figure 4.40: Qualitative tracking examples for “B1Counting”. The leftmost column shows the input, the middle the segmentation results, and the rightmost shows the final tracking results for that frame overlaid on the input.

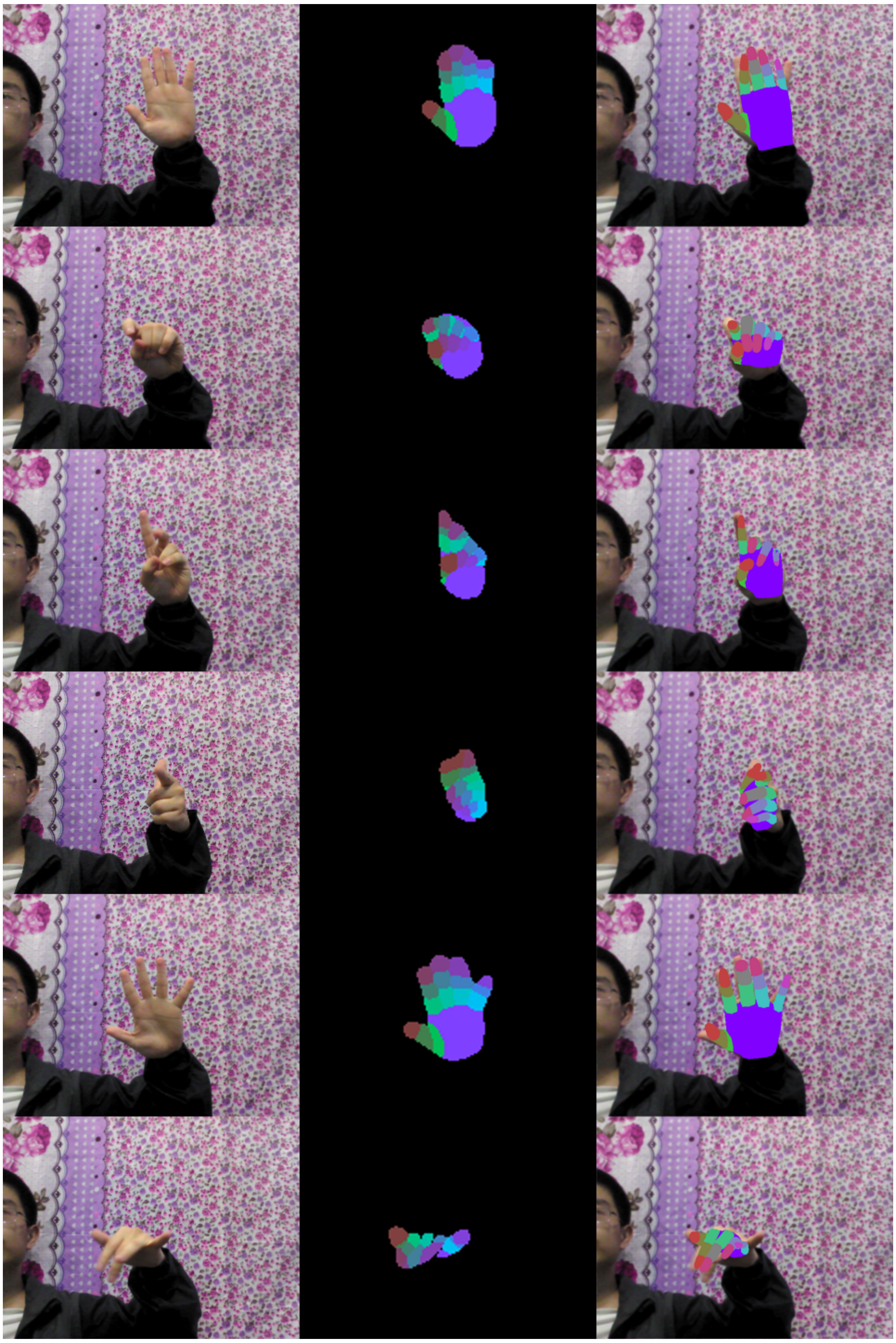

Figure 4.41: Qualitative for "B1Random". The leftmost column shows the input, the middle the segmentation results, and the rightmost shows the final tracking results for that frame overlaid on the input.

Chapter 5

Conclusion

In this thesis, the problem vision-based hand tracking was discussed and a novel approach to performing hand tracking was presented and evaluated. This approach used CNNs to extract features from RGB images in the form of dense semantic labels. A generative model was then used to recover the pose of the hand. This was achieved by using a DE-based tracking algorithm to find the pose that minimised the difference between the output of the network and a rendered semantic image created using labelled mesh model of the hand in a particular pose.

The use of semantic segmentation provides features that are precisely relevant to the pose of the hand and therefore overcomes one of the main limitations of generative hand tracking. This allows for accurate and robust tracking with all of the advantages of a model-based algorithm, such as the ability to easily incorporate physiological and temporal constraints.

The semantic segmentation performs well, with the hand parts localised in the image in an accurate and consistent manner, insofar as the results generally show that the network learned the plausible configurations of the parts of the hand in an image from the examples given. Additionally, the background estimator, for which a quantitative comparison is possible, it performs similarly to the contemporary state-of-the-art.

A significant part of the approach is the use of a DE-based hand tracking algorithm. This algorithm was tested against ICP-PSO on a joint location tracking problem and a depth-based tracking problem and compared favourably in both cases. The algorithm performed well when the semantic ground truth was provided as input. It also performed favourably to an equivalent PSO-based algorithm on the semantic fitting problem with both estimated and ground truth input.

Several factors that affect the performance of the tracking algorithm were examined, including the choice of cost function and the value of the algorithm's main hyper-parameters. The impact of these variables on the final results was generally slight. The main source of error was determined to be ambiguity in global orientation. As a result of this ambiguity, the algorithm generally falls short of contemporary CNN approaches to hand tracking from RGB in terms of relative keypoint error. When tracking from the semantic ground truth or with a global orientation oracle available, however, the tracking results are comparable to the contemporary state-of-the-art.

More generally, these results and the current direction of the literature demonstrate the potential for CNNs to help address long outstanding issues in computer vision. Vision-based hand tracking and the generative paradigm are both decades old and the limitations of conventional RGB-based algorithms were severe enough that the field was fairly stagnant before the introduction of CNNs, but has now seen real progress as a result. The application of CNNs to problem areas that have in the past been limited by a lack of determinative features is a trend that will no doubt continue to bring progress in the future.

5.1 Future work

There are several ways in which this work could extended in the future. The main focus of future work should be resolving the main source of error, which is ambiguity in the global orientation of the hand. This could be done by using some form of discriminative system to predict the orientation directly. It may also be possible to learn more descriptive labels or localised features, such as landmark points on the surface of the hand. Such features would give more information about the orientation of the palm and could guide the optimiser towards a better solution. A learned prior on the pose of the hand may also help in this regard, as the orientation of the palm would be constrained by the requirement for a likely hand pose rather than just a physically plausible one.

The tracking algorithm works well when the ground truth semantic maps are used as input. This is due to the same model being used to generate semantic estimates as is used in the tracking algorithm. When using estimated semantic maps (as in 4.40 and 4.41), the shape of the hand corresponds to the shape of the subjects hand in the image, frequently causing the palm to displaced from its correct location. There are several ways in which this could be addressed. One way would be to impose constraints on the estimated semantic maps to ensure the hand is the same shape as in the ground truth. This could be be done with some kind of CRF. Another approach would be to simply model the subject's hand more accurately. This approach may not generalise well to other subjects, however. A more generalisable way to address the problem would be to have an adaptive mesh model that is refined continuously during tracking, in a manner similar to that of Malik et al.[MEN+18], but with semantic information instead of depth. A discriminative mesh estimation approach similar to that of Baek et al.[BKK19] could also be used. It may also be useful to exploit the interdependency of the mesh and the semantic prediction by predicting both jointly using a single model.

Bibliography

[Ahm94] Subatai Ahmad. A usable real-time 3d hand tracker. In *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1257–1261. IEEE, 1994.

[Aki84] Koichiro Akita. Image sequence analysis of real world human motion. *Pattern recognition*, 17(1):73–83, 1984.

[AS03] Vassilis Athitsos and Stan Sclaroff. Estimating 3d hand pose from a cluttered image. Technical report, Boston University Computer Science Department, 2003.

[Bau72] Leonard E Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3(1):1–8, 1972.

[BBM08] Roberto Battiti, Mauro Brunato, and Franco Mascia. *Reactive search and intelligent optimization*, volume 45, pages 152–156. Springer Science & Business Media, 2008.

[BBV04a] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*, pages 466–474. Cambridge university press, 2004.

[BBV04b] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*, pages 484–495. Cambridge university press, 2004.

[Bes86] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society: Series B (Methodological)*, 48(3):259–279, 1986.

[BKK19] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. Pushing the envelope for rgb-based dense 3d hand pose estimation via neural rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1067–1076, 2019.

[BKMM$^+$04] Matthieu Bray, Esther Koller-Meier, Pascal Müller, Luc Van Gool, and Nicol N Schraudolph. 3d hand tracking by rapid stochastic gradient descent using a skinning model. In *In 1st European Conference on Visual Media Production (CVMP*. Citeseer, 2004.

[BKR11] Andrew Blake, Pushmeet Kohli, and Carsten Rother. *Markov random fields for vision and image processing*. Mit Press, 2011.

[BM98] Christoph Bregler and Jitendra Malik. Tracking people with twists and exponential maps. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*, pages 8–15. IEEE, 1998.

[Bra99] Matthew Brand. Shadow puppetry. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1237–1244. IEEE, 1999.

[CGCY18] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *European Conference on Computer Vision*, pages 1–17. Springer, Cham, 2018.

[CGH05] Fabrice Caillette, Aphrodite Galata, and Toby Howard. Real-time 3-d human body tracking using variable length markov models. In *BMVC*, volume 1. Citeseer, 2005.

[CKR17] Chiho Choi, Sangpil Kim, and Karthik Ramani. Learning hand articulations by hallucinating heat distribution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3104–3113, 2017.

[CM92] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[CMM$^+$11] Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[CN13] Ju Yong Chang and Seung Woo Nam. Fast random-forest-based human pose estimation using a multi-scale and cascade approach. *ETRI Journal*, 35(6):949–959, 2013.

[CSV] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*, volume 8.

[CWGZ19] Xinghao Chen, Guijin Wang, Hengkai Guo, and Cairong Zhang. Pose guided structured region ensemble network for cascaded hand pose estimation. *Neurocomputing*, 2019.

[CWZ$^+$18] Xinghao Chen, Guijin Wang, Cairong Zhang, Tae-Kyun Kim, and Xiangyang Ji. Shpr-net: Deep semantic hand pose regression from point clouds. *IEEE Access*, 6:43425–43439, 2018.

[CZP$^+$18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[DBR00] Jonathan Deutscher, Andrew Blake, and Ian Reid. Articulated body motion capture by annealed particle filtering. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 2, pages 126–133. IEEE, 2000.

[DD92] AC Downton and H Drouet. Model-based image analysis for unconstrained human upper-body motion. In *1992 International Conference on Image Processing and its Applications*, pages 274–277. IET, 1992.

[DDH04] Guillaume Dewaele, Frédéric Devernay, and Radu Horaud. Hand motion from 3d point trajectories and a smooth surface model. In *European Conference on Computer Vision*, pages 495–507. Springer, 2004.

[Deu11a] Peter Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, volume 35, pages 173–232. 2011.

[Deu11b] Peter Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, volume 35. 2011.

[Deu11c] Peter Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, volume 35. 2011.

[DF01] Quentin Delamarre and Olivier Faugeras. 3d articulated models and multiview tracking with physical forces. *Computer Vision and Image Understanding*, 81(3):328–357, 2001.

[DF15] Meng Ding and Guoliang Fan. Articulated gaussian kernel correlation for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 57–64, 2015.

[DGG$^+$89] John S Denker, WR Gardner, Hans Peter Graf, Donnie Henderson, Richard E Howard, W Hubbard, Lawrence D Jackel, Henry S Baird, and Isabelle Guyon. Neural network recognizer for hand-written zip code digits. In *Advances in neural information processing systems*, pages 323–331, 1989.

[DM97] Pierre Del Moral. Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 325(6):653–658, 1997.

[DMB$^+$18] Endri Dibra, Silvan Melchior, Ali Balkis, Thomas Wolf, Cengiz Oztireli, and Markus Gross. Monocular rgb hand pose inference from unsupervised refinable nets. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1188–118810. IEEE, 2018.

[Dor94] Brigitte Dorner. *Chasing the colour glove: Visual hand tracking*. PhD thesis, Theses (School of Computing Science)/Simon Fraser University, 1994.

[EK95] Russell Eberhart and James Kennedy. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer, 1995.

[FCNL12] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012.

[FJs$^+$05] Liang Faming, Wang Jian-sheng, et al. *Markov chain Monte Carlo: innovations and applications*, volume 7. World Scientific, 2005.

[FLW$^+$19] Jun Fu, Jing Liu, Yuhang Wang, Jin Zhou, Changyong Wang, and Hanqing Lu. Stacked deconvolutional network for semantic segmentation. *IEEE Transactions on Image Processing*, 2019.

[Gag17] Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, pages 651–716. MIT press, 2016.

[GCWY18] Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. Hand pointnet: 3d hand pose estimation using point sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8417–8426, 2018.

[GD96] Dariu M Gavrila and Larry S Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *Proceedings cvpr ieee computer society conference on computer vision and pattern recognition*, pages 73–80. IEEE, 1996.

[GGOEO$^+$18] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.

[GLYT17] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 5, 2017.

[GPAM$^+$14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[GPKT10] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real time motion capture using a single time-of-flight camera. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 755–762. IEEE, 2010.

[GSK$^+$11] Ross Girshick, Jamie Shotton, Pushmeet Kohli, Antonio Criminisi, and Andrew Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *2011 International Conference on Computer Vision*, pages 415–422. IEEE, 2011.

[GWC$^+$17] Hengkai Guo, Guijin Wang, Xinghao Chen, Cairong Zhang, Fei Qiao, and Huazhong Yang. Region ensemble network: Improving convolutional network for hand pose estimation. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 4512–4516. IEEE, 2017.

[HH96] Tony Heap and David Hogg. Towards 3d hand tracking using a deformable model. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 140–145. Ieee, 1996.

[HLVDMW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[Hog83] David Hogg. Model-based vision: a program to see a walking person. *Image and Vision computing*, 1(1):5–20, 1983.

[Hol97] Eun-Jung Holden. *Visual recognition of hand motion*. PhD thesis, Citeseer, 1997.

[Hri] Dionissios T. Hristopulos. *Random fields for spatial data modeling : a primer for scientists and engineers*. Advances in Geographic Information Science. Springer, Dordrecht.

[HS85] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

[HSK$^+$12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[HSKMVG09] Henning Hamer, Konrad Schindler, Esther Koller-Meier, and Luc Van Gool. Tracking a hand manipulating an object. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1475–1482. IEEE, 2009.

[HVZM$^+$12] Antonio Hernández-Vela, Nadezhda Zlateva, Alexander Marinov, Miguel Reyes, Petia Radeva, Dimo Dimov, and Sergio Escalera. Graph cuts optimization for multi-limb human segmentation in depth maps. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 726–732. IEEE, 2012.

[HWLX15] Li He, Guijin Wang, Qingmin Liao, and Jing-Hao Xue. Depth-images-based pose estimation using regression forests and graphical models. *Neurocomputing*, 164:210–219, 2015.

[HZCP04] Xuming He, Richard S Zemel, and Miguel Á Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[IMBJGK18] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. Hand pose estimation via latent 2.5 d heatmap regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 118–134, 2018.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[JDV$^+$17] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 11–19, 2017.

[KC12] Shu Gang Kang and Shiu Hong Choi. *Multi-agent based beam search for real-time production scheduling and control: method, software and industrial application*. Springer Science & Business Media, 2012.

[KH95] James J Kuch and Thomas S Huang. Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration. In *Proceedings of IEEE International Conference on Computer Vision*, pages 666–671. IEEE, 1995.

[KKKA12] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *European Conference on Computer Vision*, pages 852–863. Springer, 2012.

[Kor96] Richard E Korf. *Artificial intelligence search algorithms*. Citeseer, 1996.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Kul59] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1959.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[LBD$^+$90] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[LBOM12] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[LCS$^+$19] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019.

[Lie05] Cheng-Chang Lien. A scalable model-based hand posture analysis system. *Machine Vision and Applications*, 16(3):157–169, 2005.

[LK95] Jintae Lee and Tosiyasu L Kunii. Model-based analysis of hand posture. *IEEE Computer Graphics and applications*, 15(5):77–86, 1995.

[LMSO03] Shan Lu, Dimitris Metaxas, Dimitris Samaras, and John Oliensis. Using multiple cues for hand tracking and model refinement. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–443. IEEE, 2003.

[LMSR17] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.

[LRB$^+$16] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[LWH02] John Lin, Ying Wu, and Thomas S Huang. Capturing human hand motion in image sequences. In *Workshop on Motion and Video Computing, 2002. Proceedings.*, pages 99–104. IEEE, 2002.

[LWH04] John Y Lin, Ying Wu, and Thomas S Huang. 3d model-based hand tracking using stochastic direct search method. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pages 693–698. IEEE, 2004.

[LZ15] Dongnian Li and Yiqi Zhou. Combining differential evolution with particle filtering for articulated hand tracking from single depth images. *Int J Signal Process Image Process Pattern Recognit*, 8(4):237–248, 2015.

[Mat13] MatSoft. human-hands male basemesh. https://www.turbosquid.com/3d-models/free-basemesh-human-hands-3d-model/770920, retrieved 2019-07-22, 2013.

[MBS$^+$18] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–59, 2018.

[MEN$^+$18] Jameel Malik, Ahmed Elhayek, Fabrizio Nunnari, Kiran Varanasi, Kiarash Tamaddon, Alexis Heloir, and Didier Stricker. Deephps: End-to-end estimation of 3d hand pose and shape by learning from synthetic depth. In *2018 International Conference on 3D Vision (3DV)*, pages 110–119. IEEE, 2018.

[MES18] Jameel Malik, Ahmed Elhayek, and Didier Stricker. Structure-aware 3d hand pose regression from a single depth image. In *International Conference on Virtual Reality and Augmented Reality*, pages 3–17. Springer, 2018.

[MKO13] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics based 3d skeletal hand tracking. In *Proceedings of Graphics Interface 2013*, pages 63–70. Canadian Information Processing Society, 2013.

[MN06] Zhenyao Mo and Ulrich Neumann. Real-time hand pose recognition using low-resolution depth images. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1499–1505. IEEE, 2006.

[MNT04] Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff. Methods for non-linear least squares problems. 2004.

[MS10] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010.

[MYCML18] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5079–5088, 2018.

[NHH15] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[NM65] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[NOTA18] Vassilis C Nicodemou, Iason Oikonomidis, Georgios Tzimiropoulos, and Antonis Argyros. Learning to infer the depth map of a hand from its color image. *arXiv preprint arXiv:1812.02486*, 2018.

[NR02] Claudia Nolker and Helge Ritter. Visual recognition of continuous hand postures. *IEEE Transactions on neural networks*, 13(4):983–994, 2002.

[NSMO96] Kenichi Nirei, Hideo Saito, Masaaki Mochimaru, and Shinji Ozawa. Human hand tracking from binocular image sequences. In *Proceedings of the 1996 IEEE IECON. 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, volume 1, pages 297–302. IEEE, 1996.

[NTTC05] Ramanan Navaratnam, Arasanathan Thayananthan, Philip HS Torr, and Roberto Cipolla. Hierarchical part-based human body pose estimation. In *BMVC*, 2005.

[NWNT17] Natalia Neverova, Christian Wolf, Florian Nebout, and Graham W Taylor. Hand pose estimation through semi-supervised and weakly-supervised learning. *Computer Vision and Image Understanding*, 164:56–67, 2017.

[NWTN14] Natalia Neverova, Christian Wolf, Graham W Taylor, and Florian Nebout. Hand segmentation with structured convolutional learning. In *Asian Conference on Computer Vision*, pages 687–702. Springer, 2014.

[OB80] Joseph O'rourke and Norman I Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):522–536, 1980.

[OH99] Hocine Ouhaddi and Patrick Horain. 3d hand gesture tracking by model registration. In *Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging*, pages 70–73, 1999.

[OK94] Jun Ohya and Fumio Kishino. Human posture estimation from multiple images using genetic algorithm. In *Proceedings of 12th International Conference on Pattern Recognition*, volume 1, pages 750–753. IEEE, 1994.

[OKA11] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BmVC*, volume 1, page 3, 2011.

[OWL15] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015.

[PA17] Paschalis Panteleris and Antonis Argyros. Back to rgb: 3d tracking of hands and hand-object interactions based on short-baseline stereo. *Hand*, 2(63):39, 2017.

[PC15] Pedro O Pinheiro and Ronan Collobert. Weakly supervised semantic segmentation with convolutional networks. In *CVPR*, volume 2, page 6. Citeseer, 2015.

[PCMY15] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1742–1750, 2015.

[PMTS$^+$13] Gerard Pons-Moll12, Jonathan Taylor13, Jamie Shotton, Aaron Hertzmann14, and Andrew Fitzgibbon. Metric regression forests for human pose estimation. BMVC, 2013.

[POA18] Paschalis Panteleris, Iason Oikonomidis, and Antonis Argyros. Using a single rgb frame for real time 3d hand pose estimation in the wild. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 436–445. IEEE, 2018.

[Pow64] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964.

[PT94] Francisco J Perales and Juan Torres. A system for human motion matching between synthetic and real images based on a biomechanic graphical model. In *Proceedings of 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 83–88. IEEE, 1994.

[QSW$^+$14] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1106–1113, 2014.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[RK94a] James M Rehg and Takeo Kanade. Digiteyes: Vision-based hand tracking for human-computer interaction. In *Proceedings of 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 16–22. IEEE, 1994.

[RK94b] James M Rehg and Takeo Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. In *European conference on computer vision*, pages 35–46. Springer, 1994.

[RKK09] Javier Romero, Hedvig Kjellström, and Danica Kragic. Monocular real-time 3d articulated hand pose estimation. In *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pages 87–92. IEEE, 2009.

[RKSI$^+$14] Grégory Rogez, Maryam Khademi, JS Supančič III, Jose Maria Martinez Montiel, and Deva Ramanan. 3d hand pose detection in egocentric rgb-d images. In *European Conference on Computer Vision*, pages 356–371. Springer, 2014.

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[Roh94] Karl Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image understanding*, 59(1):94–115, 1994.

[ROL18] Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Domain transfer for 3d pose estimation from color images without manual annotations. In *Asian Conference on Computer Vision*, pages 69–84. Springer, 2018.

[SBB94] Michael Shapiro, Charles Bouman, and Calvin F Bagley. A multiscale random field model for bayesian image segmentation. Technical report, CONSTRUCTION ENGINEERING RESEARCH LAB (ARMY) CHAMPAIGN IL, 1994.

[SBIK16] Nikolaos Sarafianos, Bogdan Boteanu, Bogdan Ionescu, and Ioannis A Kakadiaris. 3d human pose estimation: A review of the literature and analysis of covariates. *Computer Vision and Image Understanding*, 152:1–20, 2016.

[SBS02] Hedvig Sidenbladh, Michael J Black, and Leonid Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *European conference on computer vision*, pages 784–800. Springer, 2002.

[SFB$^+$11] Jamie Shotton, Andrew Fitzgibbon, Andrew Blake, Alex Kipman, Mark Finocchio, Bob Moore, and Toby Sharp. Real-time human pose recognition in parts from a single depth image. 2011.

[SISB04] Leonid Sigal, Michael Isard, Benjamin H Sigelman, and Michael J Black. Attractive people: Assembling loose-limbed models using non-parametric belief propagation. In *Advances in neural information processing systems*, pages 1539–1546, 2004.

[SJC08] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[SK99] Jakub Segen and Senthil Kumar. Shadow gestures: 3d hand pose estimation using a single camera. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages 479–485. IEEE, 1999.

[Ski98a] Steven S Skiena. *The algorithm design manual: Text*, volume 1, pages 251–253. 1998.

[Ski98b] Steven S Skiena. *The algorithm design manual: Text*, volume 1, pages 254–258. 1998.

[SKR$^+$15] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3633–3642. ACM, 2015.

[SKS01] Nobutaka Shimada, Kousuke Kimura, and Yoshiaki Shirai. Real-time 3d hand posture estimation based on 2d appearance retrieval using monocular camera. In *Proceedings IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 23–30. IEEE, 2001.

[SKS12] Min Sun, Pushmeet Kohli, and Jamie Shotton. Conditional regression forests for human pose estimation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3394–3401. IEEE, 2012.

[SLMN11] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011.

[SOT13] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *Proceedings of the IEEE international conference on computer vision*, pages 2456–2463, 2013.

[SP97] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[SRAN$^+$19] Adel Saleh, Hatem Rashwan, Mohamed Abdel-Nasser, Vivek Singh, Saddam Abdulwahab, Md. Mostafa Kamal Sarker, Miguel Garca, and Domenec Puig. Finseg: Finger parts semantic segmentation using multi-scale feature maps aggregation of fcn. 02 2019.

[SSKM98] Nobutaka Shimada, Yoshiaki Shirai, Yoshinori Kuno, and Jun Miura. Hand gesture estimation and model refinement using monocular camera-ambiguity limitation by inequality constraints. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 268–273. IEEE, 1998.

[ST03] Cristian Sminchisescu and Bill Triggs. Estimating articulated human motion with covariance scaled sampling. *The International Journal of Robotics Research*, 22(6):371–391, 2003.

[SWL$^+$15] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 824–832, 2015.

[TBC$^+$16] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, et al. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics (TOG)*, 35(4):143, 2016.

[The83] Charles W Therrien. An estimation-theoretic approach to terrain image segmentation. *Computer Vision, Graphics, and Image Processing*, 22(3):313–326, 1983.

[TJCTK14] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3786–3793, 2014.

[TSLP14] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):169, 2014.

[TSSF12] Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 103–110. IEEE, 2012.

[TST$^+$15] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust articulated-icp for real-time hand tracking. In *Computer Graphics Forum*, volume 34, pages 101–114. Wiley Online Library, 2015.

[TYK13] Danhang Tang, Tsz-Ho Yu, and Tae-Kyun Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Proceedings of the IEEE international conference on computer vision*, pages 3224–3231, 2013.

[UB18] Aisha Urooj and Ali Borji. Analysis of hand segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4710–4719, 2018.

[UMIO03] Etsuko Ueda, Yoshio Matsumoto, Masakazu Imai, and Tsukasa Ogasawara. A hand-pose estimation for vision-based human interfaces. *IEEE Transactions on Industrial Electronics*, 50(4):676–684, 2003.

[WH99] Ying Wu and Thomas S Huang. Capturing articulated human hand motion: A divide-and-conquer approach. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 606–611. IEEE, 1999.

[WHH$^+$95] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *Backpropagation: Theory, Architectures and Applications*, pages 35–61, 1995.

[WLH01] Ying Wu, John Y Lin, and Thomas S Huang. Capturing natural hand articulation. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 426–432. IEEE, 2001.

[WN99] Stefan Wachter and H-H Nagel. Tracking persons in monocular image sequences. *Computer Vision and Image Understanding*, 74(3):174–192, 1999.

[WPVGY17] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[WPVGY18] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Dense 3d regression for hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2018.

[WYVG16] Chengde Wan, Angela Yao, and Luc Van Gool. Hand pose estimation from local surface normals. In *European conference on computer vision*, pages 554–569. Springer, 2016.

[XC13] Chi Xu and Li Cheng. Efficient hand pose estimation from a single depth image. In *Proceedings of the IEEE international conference on computer vision*, pages 3456–3462, 2013.

[YJLSHDY15] Ho Yub Jung, Soochahn Lee, Yong Seok Heo, and Il Dong Yun. Random tree walk toward instantaneous 3d human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2467–2474, 2015.

[YKC13] Tsz-Ho Yu, Tae-Kyun Kim, and Roberto Cipolla. Unconstrained monocular 3d human pose estimation by action detection and cross-modality regression forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2013.

[YSZ$^+$11] Hanxuan Yang, Ling Shao, Feng Zheng, Liang Wang, and Zhan Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823–3831, 2011.

[YY00] Masanobu Yamamoto and Katsutoshi Yagishita. Scene constraints-aided tracking of human body. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 151–156. IEEE, 2000.

[YY19] Linlin Yang and Angela Yao. Disentangling latent hands for image synthesis and pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9877–9886, 2019.

[YYK16] Qi Ye, Shanxin Yuan, and Tae-Kyun Kim. Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation. In *European conference on computer vision*, pages 346–361. Springer, 2016.

[YYS$^+$17] Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhant Jain, and Tae-Kyun Kim. Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4866–4874, 2017.

[Z$^+$03] Hanning Zhou et al. Tracking articulated hand motion with eigen dynamics analysis. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1102–1109. IEEE, 2003.

[ZB17] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *International Conference on Computer Vision*, volume 1, page 3, 2017.

[ZJC$^+$16] Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. 3d hand pose tracking and estimation using stereo matching. *arXiv preprint arXiv:1610.07214*, 2016.

[ZSZ$^+$16] Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei. Deep kinematic pose regression. In *European Conference on Computer Vision*, pages 186–201. Springer, 2016.

[ZTF11] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pages 2018–2025. IEEE, 2011.

[ZWZ$^+$16] Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. Model-based deep hand pose estimation. *arXiv preprint arXiv:1606.06854*, 2016.