A Model-Based Pose Estimation Framework: From Single Insects To Multiple Hands

Dissertation

zur Erlangung des akademischen Grades des Doktors der Naturwissenschaften (Dr.rer.nat.) an der Universität Konstanz im Fachbereich Informatik und Informationswissenschaft

vorgelegt von

Le Duan

Universität Konstanz

Konstanz 2019
Advisor
Prof. Dr. Oliver Deussen, University of Konstanz, Germany

Reviewers
Prof. Dr. Oliver Deussen, University of Konstanz, Germany
Prof. Dr. Dietmar Saupe, University of Konstanz, Germany

Date of Submission
January 14th, 2019

Date of Defense
May 20th, 2019
Robust pose estimation for objects such as human hands and insects plays a central role in human-computer interaction, virtual reality, augmented reality and behavioral studies of animals such as bees. Estimating poses of insects and hands are challenging because they suffer from (self-)similarities, (self-)occlusions, a varying number of targets, etc. Moreover, multi-instance pose estimation is more difficult because it requires the correct detection of all instances while also precisely localizing the corresponding target landmarks.

This thesis focuses on pose estimation problems for insects and multiple hands from single images. We present a series of frameworks based on the pictorial structure model in a progressive manner, from the first framework that can only solve part of the insect pose estimation challenges to the last framework that is able to estimate poses of multiple hands. This thesis first introduces the first framework for detailed insect pose estimation from single images. It is an exemplar of the random forest-based pose estimator with data-driven regularization. Given the landmark candidate positions predicted by random forests, a pictorial structure model-based data-driven approach is proposed to select landmark positions. Targeting the drawbacks of our random forest-based method, we present the first convolutional neural network-based framework for bee pose estimation. This framework utilizes the powerful representation capability of convolutional neural networks, is demonstrably robust and accurate, and is effective even for images that might be challenging for human annotators. Since human fingers and bee extremities share some similarities, algorithms for one object can be extended to serve the other. With slight modification in the problem formulation and the advantages of Mask R-CNN, we present the first multihand pose estimation framework based on a pictorial structure that allows detection of multi-hand instances and localization of hand joints simultaneously. Finally, we discuss the limitations of our methods and propose new ideas for future work.
Zusammenfassung


Acknowledgements

I would not be able to finish my PhD without the help of many people. First and foremost, I wish to express my sincere gratitude and appreciation to my adviser, Prof. Dr. Oliver Deussen, for the opportunity to work in his group and for his patience, generous help and support at all levels. In addition, I would like to thank my second adviser, Prof. Dr. Dietmar Saupe, for reviewing this thesis and providing valuable feedback during my study.

Many thanks to Dr. Minmin Shen, who has supported and accompanied me in all my projects during my PhD study. Without her supervision, advice, and insightful suggestions, this work would not have been possible. Thanks to all my coauthors for the in-depth discussions and cooperation. Additionally, I would like to thank Jianwei Guo and Fan Tang for their support and the time they spent proofreading my papers. Thanks to Alexander Artiga Gonzalez for his help of translating the Abstract into German.

I would also like to give my special thanks to Ingrid Baiker, for her kindness, her support in many non-academic situations, and for always being ready to help me when I experienced trouble.

I want to thank my wife, Ying Zhou. Without her love, support, encouragement, and patience, it would not be possible to concentrate on my research.

Last but not least, I would like to thank my family: my parents and all other family members in China for supporting me morally and emotionally in my life.
## Contents

3.4.3. Results and discussion ........................................ 30
3.4.4. Software ...................................................... 35
3.5. Summary ......................................................... 35

4. Bee Pose Estimation From Single Images with Convolutional Neural Network ........................................ 37
   4.1. Overview ......................................................... 37
   4.2. Related Work ................................................... 39
      4.2.1. VGGNet ..................................................... 39
      4.2.2. GoogLeNet .................................................. 40
   4.3. Bee Pose Estimation ............................................. 41
   4.4. Methodology .................................................... 42
      4.4.1. Confidence map generation ................................. 42
      4.4.2. Confidence refinement ..................................... 43
      4.4.3. Localization ................................................ 45
   4.5. Data Preparation ................................................. 46
      4.5.1. Training data preparation ................................ 46
      4.5.2. Training label generation ................................. 47
      4.5.3. Test data pre-processing .................................. 47
   4.6. Experiment ...................................................... 47
      4.6.1. Implementation details ..................................... 48
      4.6.2. Self-comparisons ........................................... 48
      4.6.3. Comparison with state-of-the-art .......................... 51
   4.7. Summary ......................................................... 54

   5.1. Overview ......................................................... 55
   5.2. Related Work .................................................... 57
      5.2.1. Multi-human pose estimation ............................... 57
      5.2.2. R-CNN family ............................................... 58
      5.2.3. ResNet ...................................................... 60
   5.3. Multi-hand Pose Estimation ..................................... 61
   5.4. Mask R-CNN for Hand Pose Estimation ....................... 62
   5.5. Confidence Refinement ......................................... 63
      5.5.1. Tree prior approximation ................................. 63
5.5.2. Final localization ................................................. 66
5.6. Data Preparation .................................................. 67
5.7. Implementation details .......................................... 68
  5.7.1. Mask R-CNN ................................................... 68
  5.7.2. Tree prior approximation ................................... 69
5.8. Experiments ......................................................... 69
  5.8.1. Results and discussion ...................................... 70
5.9. Summary ............................................................ 76

6. Conclusion and Future Work ........................................ 78
  6.1. Conclusion .......................................................... 78
  6.2. Discussion .......................................................... 79
  6.3. Future work ........................................................ 80

7. Author’s Contributions .............................................. 82
  A. Appendix ............................................................. 84
    A.1. Shape Descriptors ............................................. 84

Bibliography ............................................................ 88
In this thesis, we present new insights into the research areas of insect pose estimation and human hand pose estimation, especially for the multi-hand pose estimation problem. We show that, generally, algorithms for one object can be extended to serve another object and even more instances. In the first part of the thesis, methods for insect pose estimation are developed step by step, from simple cases to more complex situations. In the latter part, methods for insect pose estimation are expanded to the task of estimating multi-hand poses. The thesis concludes with a discussion on challenges and future goals for object pose estimation.

1.1. Insect Pose Estimation

Insects constitute successful models for studying the neuronal mechanisms of learning and memory due to their remarkable learning abilities [62, 63, 83]. Under controlled conditions, insects can be restrained, and their behavior monitored as movements of body parts, such as their antenna and mouthparts. Among insects, honey bees are reported to have the highest and broadest range of learning abilities [63] since they are able to associate a food reward with different kinds of stimuli such as odors, colors, and visual patterns.

Harnessled bee individuals have been used primarily in studies on associative learning by classical conditioning, where an odor is paired with a sucrose reward. Whether a bee has learned the association is usually measured by the proboscis extension response (PER) [6]. Generally, odors do not evoke the PER of bees, the
1.1. INSECT POSE ESTIMATION

Figure 1.1: Example images of various bee poses. Three colors are used to mark different tips of body parts: green for the right antenna, red for the tongue and blue for the left antenna. (a) every body part is present; (b) part of the tongue is occluded by the right antenna; (c)-(e) some body parts are not visible; (f) the antennae may move backwards in some rare cases; (g) the bee is fed sugar water with a stick.

PER normally occurs only when the bee is fed the sucrose solution. During training, an odor is presented in close temporal association with the sugar water. Afterwards, the PER of a bee may be evoked either when the bee is stimulated with sugar water or with a previously conditioned odor.

Traditionally, PER is measured by human observers and recorded in the form of binary responses (0 or 1) to prevent subjective biases introduced by observers. This procedure is tedious and time-consuming. Moreover, binary measurement is not adequate to demonstrate individual differences in learning and memory performance. Additionally, the temporal patterns of antennae movement are important because they might change upon sensory stimulation and reveal the internal states of bees such as sleep and wakefulness. However, existing animal and insect tracking systems are not suitable, or at least do not perform well for this case, since they (1) require prior marking of the animal [44], (2) are only capable of tracking the whole insect body [101], (3) only work with slow and coherent motion [4], (4) track only one type of body part [65], and (5) require additional human intervention [85]. Methods that can automatically determine positions of insect keypoints with high accuracy would thus be of great value for behavioral entomologists.

Informally, the task of localizing the positions of insect body parts is defined as a pose estimation problem, where the pose of a bee is determined by the tip locations of two antennae and the proboscis. To understand the difficulty of the problem, Fig. 1.1 shows some examples of intricate bee behaviors that need to be tracked. To train the bees and study their responses under controlled stimulus conditions, videos of bees may be recorded under different light conditions (Fig. 1.1c,f) or with human interference such as feeding the bee sugar water with a stick (Fig. 1.1g).
1.1. INSECT POSE ESTIMATION

The bees' responses can take the form of rapid movements in body parts such as the antennae or mouthparts. Thus, estimating the pose of a bee suffers from the following challenges:

- **Occlusions and self-occlusions.** Insect body parts can be occluded by other objects when human interferences are involved (Fig. 1.1g). Unlike regular occlusion, in which the target is blocked by other objects, self-occlusions occur when some targets overlap themselves. Insect body parts are highly clustered due to their small sizes. Thus, self-occlusions are prevalent in insect body parts (Fig. 1.1b). In some extreme cases (e.g., Fig. 1.2), occlusion and self-occlusion occur at the same time, it is even difficult for humans to label occluded tips.

- **Similarities and self-similarities.** Insect body parts usually share similar shapes with each other and have a dark and textureless appearance. Moreover, our image data are a set of 2D videos that do not contain depth information. As shown in Fig. 1.2, sticks, bee body parts and the image background have similar visual appearances. It is challenging to localize tip positions of bee body parts in such a complicated situation and to distinguish different bee body parts.

- **Varying number of targets.** A varying number of body parts might be observed in consecutive video frames (e.g., the bee tongue does not appear in Fig. 1.1c, right antennae are absent in Fig. 1.1d-e and only a small portion of tongue is visible in Fig. 1.1e); thus, we have incoherent motion paths, and trajectories might have long tracking gaps. In such cases, motion cues only provide limited information for predicting the current pose. In addition, since the correlations between different body parts are generally weak, the presence of a certain body part cannot be inferred by the presence of other body parts.
1.2. Hand Pose Estimation

Robust hand pose estimation plays a central role in human-computer interaction, virtual reality (VR) and augmented reality (AR). For example, in some VR/AR applications, hand joint positions need to be localized to interact with virtual objects. In the scenario of video gaming, hand gesture recognition is required to enable the user to interact with the game. With the development of the Internet of Things [2], it will come as no surprise to see that in the future, people will be able to control everything in their smart homes with simple hand gestures.

In this thesis, the task of estimating hand pose is defined as locating 2D finger joint and/or tip positions (i.e., $x$ and $y$) in individual images or video frames. As shown in Fig. 1.4, similar to the bee pose estimation problem, estimating the pose of a hand is difficult because hand fingers exhibit (self-)similarities (Fig. 1.4b) and (self-)occlusions (Fig. 1.4d). Moreover, due to the change of viewpoint, this task also suffers from a varying number of targets (Fig. 1.4a,e). However, hand pose estimation is more challenging due to the following difficulties:

- **Large number of targets.** Estimating a hand pose requires localizing a greater number of joints/tips. Depending on the task, the target number of joints/tips can range from five [90] to nineteen [96]. In some cases, not only the hand joints and tips need to be localized, positions of palm keypoints are also of interest, i.e., totally twenty-six keypoints (Fig. 1.3).

- **Segmentation.** In contrast to the insect pose estimation problem, where the centroid of an insect is supposed to remain unchanged through the whole video, obtaining a hand segmentation from an image background is often the first step performed in most hand pose estimation algorithms [34, 67, 90, 93, 96, 107, 112]. Having the region of the hand, hand pose estimation methods are applied to each of them. However, it is difficult to obtain a clear segmentation since
1.3. SUMMARY OF CONTRIBUTIONS

(1) hands and arms are connected and have the same skin color, and (2) hand positions can vary significantly and rapidly. Thus, the performance of the hand pose estimation algorithm depends highly on the segmentation results.

- **Size and shape variance.** Assuming that all hand instances segmentation are obtained, it is still challenging to precisely localize all keypoints since the size and shape of each hand instance can be different (Fig. 1.4). As illustrated in Fig. 1.4(a) and Fig. 1.4(e), although there is only one fingertip of these two hand instances is visible, sizes and shapes of the two hands differ a lot.

Since estimating poses of hand and bee extremities shares some similarities, and the hand pose estimation problem can be viewed as a superset of insect pose estimation, it is tempted to apply the existing hand pose estimation algorithm for estimating bee pose. However, directly apply existing hand pose estimation algorithms to estimate bee pose may result in poor performance since the-state-of-the-art hand pose estimation methods do not focus on the **varying number of target** challenge, which is the most important part for insect pose estimation. Thus, a general algorithm that is efficient and effective for pose estimation of both objects is desirable. In addition, although much progress has been made in the area of hand pose estimation [34, 67, 90, 93, 96, 107, 112], **multi-hand** pose estimation (Fig. 1.5) is still mostly unsolved. A good solution, however, would provide more flexibilities and possibilities in many HCI applications.

1.3. Summary of Contributions

This thesis outlines contributions to the aforementioned challenges of insect/hand pose estimation. Those challenges are solved step by step, from the first framework...
that only focuses on some of the challenges of bee pose estimation to the last framework that is able to estimate multi-hand poses. The specific contributions are the following:

Figure 1.5.: Examples of NYU2Hands dataset. This dataset is created based on the popular single hand dataset NYU hand dataset[96]. Estimating multi-hand poses is difficult since it requires the correct detection of all hand instances while also precisely localizing the corresponding hand joints. As can be see from (a)-(i), hand positions as well as poses vary a lot. In this thesis, a general framework that can fulfill multi-hand instance detection and hand joint localization simultaneously is presented.

- A general framework for detailed insect pose estimation from single images. It is an exemplar of a random forest-based [10] pose estimator with data-driven regularization. Given landmark candidate positions predicted by random forests, a pictorial structure [29] (PS) model-based data-driven approach is proposed to adaptively weight and select precise landmark positions, incorporating the probable global structure of the anatomy to be estimated. The proposed framework mainly aims at dealing with the challenges of self-occlusion, self-similarity and a varying number of targets.

The author of this thesis was involved in developing the contribution of this
work, which includes generating new ideas about random forests for image segmentation, implementing code for experiments and writing text for part of the paper. The final results were published in


- A new convolutional neural network (CNN)-based framework for bee pose estimation. This is an extension of the previous work. The proposed framework utilizes CNNs to learn a mapping from the global structure and local appearance of a bee to its corresponding pose. A new CNN architecture based on VGGNet [88] is presented to detect tip candidates. The global structure of a bee is represented by features extracted from a fine-tuned GoogLeNet [92]. With the above mentioned idea, the pose of a bee is estimated by integrating both the detected local landmarks and the global constraints of the bee. The proposed framework is able to handle all challenges of bee pose estimation.

The author of this thesis developed the main idea of this work, implemented code for experiments, and took the lead in writing the following papers:


DUAN, L., SHEN, M., Quek, B., CUI, S. AND DEUSSEN, O. **A Pictorial Structure Model-based Framework For Bee Pose Estimation.** In *Pattern Recognition Letters.* (Submitted.)

- A novel framework based on a PS model to estimate 2D multi-hand poses from depth images. Unlike most existing single-hand pose estimation algorithms that are either subject to strong assumptions or depend on a weak detector to segment the human hand, the proposed framework utilizes a specific CNN architecture, i.e., Mask R-CNN [40], to detect multi-hand instances and localize hand joints simultaneously. Two multi-hand datasets, dexter2Hands and
NYU2Hands (Fig. 1.5), which are based on the popular single-hand datasets dexter1 [90] and NYU hand pose [96], respectively, are presented.

The author of this thesis took the lead in the project, including algorithm design, code implementation, experimental design and writing the text for the paper. All findings were published in


In this thesis, text from the aforementioned publications are used without citation marks.

1.4. Other Publications

The author of this thesis also contributed in the following publications. These papers are not part of the thesis.

• GUO, Z., KEE, C., DUAN, L., DEUSSEN, O. AND ZHOU, Y. Real-time detection of structural landmarks in corneal topography with single shot detector. In Biomedical Signal Processing and Control. (To be submitted.)


1.5. Structure of the Thesis

This thesis introduces a sequence of works based on the PS model for object pose estimation. In the following chapter, we briefly review the most related literature and show the state of the art in the field of human body pose estimation, single hand pose estimation, and animal tracking.

In Chapter 3, the single-image insect pose estimation framework is introduced. The proposed framework presents the interplay between Random Forest (RF) and PS model, and part of insect pose estimation challenges are solved. We extend our previous work with the discriminative capability of CNNs in Chapter 4 to address the limitations of our RF-based method. In Chapter 5, the framework for multihand pose estimation from single depth image is presented. We show that, with a proper choice of CNN architecture and problem formulation, our algorithm for single insect pose estimation can also be applied to the task of estimating poses of multiple hands. In addition to Chapter 2, we also review some closely related works in Chapter 4 and Chapter 5. Finally, Chapter 6 concludes this thesis by summarizing the presented contributions and discussing the limitations of the presented methods, detailed ideas for future work are also discussed.
CHAPTER 2

Literature Review

2.1. Overview

As mentioned in Chapter 1, the estimation of insect and human hand poses shares many similar ideas, and existing hand pose estimation methods mostly originated from human pose estimation frameworks. In this chapter, we review some closely related methods. Note that a full review of all the literature is outside of the scope of this thesis; we limit our discussion to prior works that 1) have similar ideas to ours, 2) use the same techniques, 3) can be used to improve the performance of our algorithms and 4) are in the same research direction. Since our algorithms are all based on pictorial structures (PS) [29], we present a brief introduction of the PS model and its applications in Section 2.2. Random forest-based algorithms are discussed in Section 2.3. In Section 2.4 and Section 2.5, we review frameworks that apply a convolutional neural network (CNN) for human and hand pose estimation. Some related generative and tracking-based methods are introduced in Section 2.6 and Section 2.7.

2.2. Pictorial Structure for 2D human pose estimation

The PS model is a general method for object detection. A typical PS model assumes that an object can be decomposed into a collection of parts with constraints between certain pairs of parts. A common way to represent a PS model is in terms of an acyclic graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ correspond to the $n$ parts and
2.2. PICTORIAL STRUCTURE FOR 2D HUMAN POSE ESTIMATION

$E$ is the set of edges connecting pairs of parts. A particular instance of the object is given by a configuration $X = \{x_1, x_2, ..., x_n\}$, where each $x_i$ specifies the location of part $v_i$. Given an image $I$, the distribution $P(X|I)$ measures the likelihood of a specific pose given an image $I$, and is modeled as a posterior probability that can be written as

$$p(X|I) \propto p(I|X)p(X). \quad (2.1)$$

where $P(I|X)$ measures the likelihood of seeing a specific image given a particular pose. The distribution $P(X)$ corresponds to a kinematic tree prior, which measures the prior probability that an object is at a particular position and is supposed to encode probabilistic constraints on part relationships and capture the unified global structure of objects in the training data. $P(X|I)$ is often computed via the maximum a posteriori (MAP) estimation.

Before the advent of convolutional neural networks, the PS model was the mainstream in the research domain of estimating 2D human poses from images. The original PS model approach presented in [29] is based on a simple appearance model that requires background subtraction, which is inappropriate in most cases, and the spatial relations of body parts are modeled by a tree model. Though tree-structured models allow for efficient inference, they often suffer from double counting; for example, given the position of the torso, two arms are localized independently and may respond to the same image region. Since then, many sophisticated extensions have been proposed by including stronger part detectors and/or more complex body part relationship models [3, 21, 49, 73, 74, 81, 82, 89, 91, 95, 102, 106].

As identified in [74], the most popular part representation is based on discriminatively trained orientation invariant appearance templates [3, 82] that are composed of histogram of oriented gradients [20] or shape context features [64]. These appearance models are extended by either generalizing to mixtures of part appearance templates [95, 106], or including new types of features such as skin or background models [81], part segmentation features [102], object contours [81] and rotation invariant features [74]. Another important part of the PS model is the encoding of spatial part relationships. Hierarchical models [91, 95] employ a hierarchical tree structure to represent the relationships between parts at different scales and sizes.
These models assume that larger body parts (e.g., arms, legs) can often have significant image structures that may be easier to detect and could be used to infer the location of smaller body parts (Fig. 2.1a). Loopy models [21, 49] introduce loops to augment the tree structure with additional edges that capture more complex relationships among parts such as symmetry, occlusion and long-range relationships (Fig. 2.1b). These methods require approximate inference during both learning and testing. In [73], a poselet conditioned PS model was presented in which the tree pose prior is computed according to the input image.

The traditional PS model approach for human pose estimation has some limitations; for example, part templates are not perfect and may be sensitive to noise, and the kinematic tree model can only be used to infer a fixed number of joints. For tasks such as bee or hand pose estimation, which suffer considerably from occlusions and similarities, the PS model is not suitable. However, the idea that the object pose is estimated by combining local appearance scores and part relationships forms the basis of our algorithms throughout this thesis.

### 2.3. Forest based pose estimator/landmark detector

Random forests (RF) [10] describe an ensemble of decision trees trained independently on a randomized selection of features. The authors in [87] used RF to recover the human pose from a single frame by classifying each pixel into one of 38 different
2.3. FOREST BASED POSE ESTIMATOR/LANDMARK DETECTOR

Body parts. Body part centers are located by mean shift [15]. A similar approach has been applied to estimate hand poses in [50]. In [33], Hough forests have shown that objects can be effectively located by training the RF regressors to predict the position of a point relative to the sampled region and then running the regressors over a region and accumulating votes for a likely position. In these studies, the final estimation is found as the center of the densest vote mass by mean shift. If the objects are occluded or missing on the test image, the candidates of the target point returned by the trained RF regressor will contain outliers. Therefore, it is crucial to remove such outliers to guarantee a reasonable estimation under the condition of occlusion. Following that, the authors in [39] presented a framework that uses a regression forest to vote for occluded body joints. In [28], the authors assumed that the pose parameters stored in a single leaf would follow a Gaussian distribution and discarded the leaves with high variances by simple thresholding when performing mean shift. Similarly, the hand pose estimator in [94] was refined by a mean shift-based method to recover poorly detected joints when they were occluded or missing.

In addition to the occlusion problems that our input images have, there are also less informative visual features for disambiguating objects with a similar appearance. As indicated in [26], landmark detectors based on classification may produce highly interchangeable responses due to very similar local appearance patterns of different anatomical body (sub)parts. Further disambiguation is required that incorporates the global structure of objects. Existing works on medical image analysis for the localization of landmarks exploit global landmark relations represented by either repetitive anatomical patterns [26] or shape models [12] for regularization.

In our task of insect pose estimation, however, such stable part relationships are not applicable; using RF to directly regress tip positions could lead to large pixel errors. Moreover, the typical RF regression approach can only handle cases of occlusion, but it is not applicable to infer nonexisting tips. In Chapter 3, we propose a method to learn the global landmark relation from training data based on a PS model to address the aforementioned issues.
2.4. CNN-based human pose estimation

Currently, using CNNs is the state-of-art method for the task of human pose estimation. The authors in [99] first fine-tuned the CNN proposed in [52] for directly regressing human body joints. However, as noted in [47, 94], mapping the image directly to 2D/3D joint positions is highly nonlinear and could result in poor generalizability. Instead of regressing joints directly, the authors in [98] adopted the confidence map representation of body keypoints to improve the localization accuracy. Since the human body is highly structured, relationships among body joints can be considered to improve the pose estimation result. One common approach to improve the estimation result is to integrate the graphical model into a network. In [98], the keypoint relationship was estimated by a Markov random field (MRF)-based spatial model. In [16], a CNN for predicting confidence maps of the human body joints was proposed. In that approach, relationships between body joints are taken into account, and the features of each joint are first learned individually and then refined by integrating the features of neighboring joints. The authors in [105] combined CNN with expressive part mixture models to refine the output.

Another way to combine human body part relationships with network design is to use a multistage network architecture. In this strategy, the results of the current stage are enhanced and regularized by the results of the previous stage. In [66], a network architecture called stacked hourglass that can better capture the spatial body part relationships was introduced. This work was further extended in [17] with a multicontext attention mechanism. In [11], a cascade network architecture that first detects body parts and then regresses body keypoints was proposed. The authors in [103] designed a sequential CNN architecture that directly operates on belief maps from previous stages.

Because movements of bee body parts are independent, algorithms for human pose estimation cannot be directly applied in this case. For example, the location of the right antenna tip cannot be determined by the location of the tongue, and the presence of a tongue cannot be inferred by the presence of antennae. Moreover, for tasks aiming at estimating only the fingertips of the human hand, state-of-the-art human pose estimation frameworks are also not suitable. Motivated by the existing human pose estimation algorithm, our bee pose estimation framework presented
2.5. CNN-based hand pose estimation

Similar to human pose estimation, CNN has also been widely used for estimating hand poses. The authors in [97] first presented a CNN architecture for predicting the confidence maps of hand joints, where the 3D hand pose is recovered from estimated 2D confidence maps. This work was subsequently extended in [34] by projecting the depth image of the hand onto three orthogonal views and estimating the 3D hand pose from multiview confidence maps to solve the problem of lacking 3D information in the 2D confidence maps. In [67], a multistage CNN was presented that enforces a prior on hand poses to directly predict hand joint positions, where the hand pose is then iteratively refined by a feedback loop. Unlike [67, 97], which directly output joint locations, [111] used an intermediate model layer to learn hand model parameters, for which joints are predicted based on the learned model. The authors in [35] presented a 3D CNN that regresses 3D hand joint positions directly from the depth image, the same authors extended their work in [36] by proposing a method that takes a point cloud as input and outputs pointwise estimations.

The aforementioned methods assume that the hand has already been detected. Commonly, a hand is detected either by assuming that it is the closest object to the camera appearing in images [67, 68] or by a pixel-level classification random forest [97]. Recently, some frameworks that can detect a hand and estimate its pose were presented. In [112], a three-stage approach that estimates 3D hand poses from regular RGB images was proposed. In that approach, the hand is first located by a segmentation network that serves as input to another network for 2D hand pose estimation. The final 3D hand joint positions are localized by combining the estimated 2D positions and the 3D pose prior information. Similarly, the authors in [13] presented a framework that can detect a hand and estimate the 3D hand pose simultaneously from depth images.

Existing hand pose estimation methods focus on estimating a single hand from images. By contrast, our framework presented in Chapter 5 can simultaneously fulfill multihand instance detection and hand joint localization. Moreover, as noted in [78],
2.6. GENERATIVE MODEL-BASED OBJECT POSE ESTIMATION

the PS model appears rarely in the hand pose estimation literature; we introduce
the PS model into this research direction for the first time.

2.6. Generative model-based object pose estimation

From a machine learning point of view, existing algorithms can be classified into two
categories: discriminative and generative methods. Discriminative methods model
the dependence of target values on observed variables. In other words, the method
learns the underlying regulation encoded in the training data automatically and
outputs a target value when seeing the input data. Thus, this category of methods
depends highly on the quality of the training data.

Generative methods, in contrast, assume that data points are generated from
some models. Target values are estimated by fitting the generated hypotheses into
observed values using optimization algorithms. In [25], an algorithm based on a par-
ticle filter was proposed to track human poses. This work was extended in [32] with
a multilayer framework, wherein the first layer pose is estimated from a silhouette
and color data, and the results are refined in the second layer by local optimization.
Similarly, the authors in [69] used particle swarm optimization (PSO) to track single
and interacting hands from RGB-D images. In [5], interacting hands were tracked
by a combination of detected salient points and energy minimization. The authors
in [61] presented a real-time system based on physical constraints to track the hand
pose.

Generative approaches require complex model calibration and are sensitive to
model initialization. Given sufficient computational resources, the performance of
model-based tracking algorithms can be highly accurate. In contrast, discriminative
methods do not require explicit specification of the object model and motion con-
straints and are fast and tracking-free. However, the performance of discriminative
methods is heavily correlated with the quality of the training data. Though in this
thesis we only focus on two discriminative methods (i.e., RF and CNN) for object
pose estimation, a direction for future work could be combining discriminative and
generative methods to achieve the superior benefits of both.
2.7. Animal tracking

Prior to our work, state-of-the-art methods for animal behavior analysis were mostly tracking-based [22]. In [9], a Kalman filter-based system that could track the animal and human kinematics was presented. However, Kalman filters are only suitable for tracking linear motion, but the motions of humans and animals are nonlinear [76]. Subsequently, particle filtering was used in some insect tracking algorithms to maintain the identity of objects throughout a whole video sequence [7, 54, 101]. Particle filtering also has some drawbacks, as explained in [71], it is often only effective for short tracking gaps, and the search space becomes significantly larger for long gaps. Similarly, data association techniques that were applied in [4] also cannot address the tracking gaps. To develop a more efficient algorithm, some studies have incorporated higher-level attributes that characterize specific insect motion into a learning diagram. In [30], overlapping larvae are separated by assigning object labels to each pixel, given user-annotated examples of encounters of two larvae as boundary conditions. For modeling occluded spatial temporal regions, dozens of examples of encounters of two larvae need to be selected. A behavioral model was proposed in [101] by first abstracting local motions and then modeling the behavior as a dynamic model on such local motions. However, the Markov model used in [101] for behavior limits its applications to some of the latest behavioral studies that require multitarget tracking, because the number of parameters to specify the transition and observation models is exponential in the number of moving objects. In [85], a track linking-based framework was proposed; it first detects bee body parts within bounding boxes, and the tip of each part is further estimated in subsequent bounding boxes. Unlike our fully automated approach presented in Chapter 3 and Chapter 4, the method in [85] requires user interactions for rectifying probable erroneous hypotheses in some frames.
3.1. Overview

Automated image-based tracking and pose estimation has received increasing interest from both the biology and computer science community, as its developments enable remote quantifying and understanding individual behavior that was previously impossible [22]. Therefore, automatic insect tracking techniques have been a research topic in biological image analysis [7, 8, 51]. The movements of harnessed insects’ body parts, such as antennae or mouthparts, provide information for studying insect behavior. Motivated by the latest behavioral studies in biology [45], we aim to localize the landmarks as the tips of body parts (e.g. a bee’s antennae or tongue shown in Fig. 3.1) to provide detailed pose information.

In contrast to most existing works that aim to estimate the center of mass (position), detecting the detailed body posture and position of appendages (pose) is more challenging. Due to (self-)occlusions, (self-)similarities and varying numbers of targets, most existing tracking/pose estimation algorithms are not applicable to our task. As shown in Fig. 3.1, the insect is fed with sugar water by a stick, and it may respond by extending its tongue. It is required to infer the presence of the tongue before localizing it. Similarly, an antenna may be absent when it moves above the
3.2. Insect Pose Estimation

head (Fig. 1.1d) or may suffer from heavy motion blur (Fig. 1.1e).

Addressing the aforementioned issues, we present a unified framework that incorporates the geometric model as a prior and utilizes the RF model to estimate the possible positions of a body part with pixel precision. Under this framework, the maximum a posteriori (MAP) estimation is used as the respective landmark position.

![Example images of various bee poses. Three colors are used to mark different tips of body parts: green for the right antenna, red for the tongue and blue for the left antenna. (a) every body part is present; (b) part of the tongue is occluded by the right antenna; (c)-(e) some body parts are not visible; (f) the antennae may move backwards in some rare cases.](image)

The main contributions of the proposed insect pose estimation framework are as follows:

- To the best of our knowledge, the proposed framework is the first method for obtaining detailed insect pose estimations from a single frame.

- Our method is an exemplar of a random forest-based pose estimator with data-driven regularization. Given the landmark candidate positions predicted by the random forests, we propose a data-driven approach to adaptively weight and select precise landmark positions, incorporating the probable global structure of the anatomy to be estimated.

- We provide a software with a graphical user interface for biological research.

### 3.2. Insect Pose Estimation

Following the standard Pictorial Structure (PS) model [29] setting, we assume that an insect can be decomposed into a set of parts. The pose of an insect is defined as \( X = \{x_1, x_2, ..., x_N\} \), where the state of part \( n \) is defined as \( x_n = \{y_n, t_n\} \), \( N \) is the number of insect body parts. In the case of bee pose estimation, \( N \) is three, i.e., a bee is composed of left antenna, tongue and right antenna. \( y_n = \{x_n, y_n\} \) is
3.3. Combined Landmark Position Proposals

To map the image $I$ of a bee to the corresponding pose $X$ is a difficult problem in computer vision; our framework imposes constraints to the solution space based on the position of the bee body part tip in our defined coordinate system as shown in Fig. 3.2(a). Since positions of bee head might be changed in different experiment trials, the additional coordinate system is presented for the purpose of extracting the information of the relative position of each object to the bee head. $t_n = \{0, 1\}$ is the state indicating the presence of part $n$. In some tasks, bounding boxes of insect body parts may also be of interest. The state of a part is defined as $x_n = \{y_n, t_n, \theta_n, s_n\}$, where $\theta_n$ is the absolute orientation of part bounding box and $s_n$ is the scale of the part. Note that $y_n$ and $\theta_n$ might be redundant for the task of bee pose estimation. However, they may provide different information in more complex situations (Fig. 3.2b).

Our primary goal is to detect the presence of insect body parts as well as the tip locations of present body parts; thus we mainly focus on introducing the method for insect pose estimation. However, with slight modification, our presented framework can be extended for determining the absolute orientation of part bounding box and the scale of the part. To simplify the problem, we take the bee pose estimation problem as the example and the bee head centroid (denoted as $x_0$), mouth center and head contour are assumed to be known.

Figure 3.2: (a) The coordinate is defined by setting the centroid of the insect head as the origin $o$, the line from the origin towards the mouth center as the $x$-axis. The head centroid and the mouth center are assumed to be known. (b) Example of ant pose estimation. Bounding boxes of ant body parts as well as tip positions need to be detected.
two cues: local appearance and global structure. We use the RF model to predict the class label \( c \) \((c = \{0: \text{background}, 1: \text{right antenna}, 2: \text{left antenna}, 3: \text{tongue}\})\) at pixel precision based on the patch appearance (Fig. 3.3a,b,d). Additionally, we propose a method based on the PS model to learn the global structure (Fig. 3.3a,c,e); the overall framework is shown in Fig. 3.3. We do not make any specific assumption about the anatomical model of an insect’s head (which was done in [85]); instead, we use a common assumption that holds for generic objects: the same types of insects have similar appearances when they are in similar poses. The global structure of an insect is represented by its silhouette, which is a \( d \)-dimensional datapoint \( f \in \mathbb{R}^d \). Based on this assumption, it is expected that these data points will lie on or near a low-dimensional manifold, in which the neighborhood of each data point is preserved. The likelihood of the global structure of the unknown pose is learned by the nearest neighbor (NN) method and used as a constraint to regularize the mapping estimated by the RF model.

![Figure 3.3: Data flow of our RF-based framework. (a) Input image; (b) RF classifier. (c) KNN search. (d) RF classification result. (e) Probability mass of KNN result. (f) Final tip localization.](image)

Given the image \( I \), the posterior probability \( p(X|I) \) is computed as

\[
p(X|I) \propto p(I|X)p(X) \tag{3.1}
\]

where \( p(I|X) \) is the likelihood of the image evidence given a particular pose, and \( p(X) \) corresponds to a tree prior according to the PS model. Both terms are learned from the training data. Specifically, we propose a method to adaptively construct
3.3. COMBINED LANDMARK POSITION PROPOSALS

Figure 3.4: Bee body part graph. Black circle: bee head centroid. Green circle: Right antenna. Red circle: tongue. Blue circle: Left antenna. (a) Complete graph of bee body parts. (b) Tree-structured graph of bee body parts.

the graph for the insect geometric model.

3.3.1. Geometric model

Typical PS models assume that an object can be decomposed into parts connected with pairwise constraints that define the prior probability of part configurations. As we aim to estimate the tips of body parts appended to the insect’s head, we first build a complete graph (Fig. 3.4a) to model the pairwise relations between tips as well as each tip $x_n$ and the centroid of the head $x_0$. The weights of the edges are assigned by the average Euclidean distance between each pairs computed from training data. Then, the PS model of a bee $G = (V, E)$ (Fig. 3.4b) is learned by computing the minimum spanning tree of the complete graph. $V = \{v_0, v_1, ..., v_N\}$ corresponds to head centroid and $N$ bee body parts, where the location of each $v_n$ is specified by $x_n$. The resultant $E$ is the set of pairs of each tip and the centroid, and the centroid is the root. According to Fig. 3.4(b), the prior $p(X)$ which encodes probabilistic constraints on part configurations can be factorized as

$$p(X) = p(x_0) \prod_{n=1}^{N} p(x_n | x_0).$$  \hspace{1cm} (3.2)

Assuming that part likelihood is conditionally independent [3], $p(I|X)$ can be factorized as

$$p(I|X) = \prod_{n=1}^{N} p(I|x_n),$$  \hspace{1cm} (3.3)

where $p(I|x_n)$ is the likelihood of part $n$. 

22
3.3. COMBINED LANDMARK POSITION PROPOSALS

Substituting Eq.(3.2) and Eq.(3.3) into Eq.(3.1), Eq.(3.1) can be factorized as

\[ p(X|I) \propto p(x_0) \prod_{n=1}^{N} p(I|x_n)p(x_n|x_0). \]  

(3.4)

Although we model the global structure in the form of Eq.(3.2), it is different from typical PS models that assume all parts are present. As we focus on pose estimation rather than detecting the insect head, \( x_0 \) is assumed to be known. To represent the global structure of a pose, we extract a feature vector \( f \in \mathbb{R}^d \) for each image combining five types of silhouette features: the edge histogram descriptor (EHD) [31], the geometrical feature descriptor (GFD) [56], the shape signature histogram descriptor (SSHD) [108], the Fourier descriptor (FD) [109] and the Hu moments descriptor (HuMD) [43]. The dimensionality of \( f \) is \( d = 88 \); thus, no dimensionality reduction is required. Details of the shape descriptors can be found in Appendix A. For a new visual input represented by silhouette features \( f \) of image \( I \), we find its \( K \) neighbors in the visual feature space and construct the neighborhood \( K \) in the pose space.

As the nodes of landmarks of \( G \) are all leafs (Fig. 3.4 b), and the likelihood of individual landmarks is conditionally independent, the node representing \( x_n \) will be removed if the frequency of the presence of \( x_n \in K \) is lower than a threshold value \( \tau \), i.e.,

\[ t_n = \begin{cases} 
1 & \text{if } \sum_{i=0}^{K} t_{ni} > \tau \\
0 & \text{otherwise.} 
\end{cases} \]  

(3.5)

where \( t_{ni} \) indicates the presence of part \( n \) in \( K \). Since \( x_n = \{y_n, t_n\} \), \( x_0 \) is known, Eq.(3.6) can be factorized as

\[ p(X|I) \propto \prod_{n=1}^{N} (p(I|y_n))^{t_n} (p(y_n))^{t_n} \]  

(3.6)

where \( p(I|y_n) \) is the likelihood of position based on the local appearance of \( y_n \).

Assuming that \( y_n \) follows a multimodal Gaussian distribution, the model pa-
3.3. COMBINED LANDMARK POSITION PROPOSALS

Parameters \((\mu_y^n, \Sigma_y^n)\) are learned from the training samples in \(K\). \(p(y_n)\) is computed as

\[
p(y_n) = \mathcal{N}(y^n | \mu_y^n, \Sigma_y^n).
\]  \(3.7\)

3.3.2. Random forest based classifier

We use an RF model to compute the part likelihood \(p(I|y_n)\). The RF is an ensemble of randomized decision trees. It takes image patches as input and assigns class labels to those inputs. Let \(q(y_i) = \{q_1(y_i), q_2(y_i), ... q_F(y_i)\}\) denote a feature cuboid of an image patch centered at \(y_i\), where \(q_k\) are the extracted visual features with fixed size, \(F\) is the total number of feature channels, and \(y_i\) is the image coordinate. In our case, we use 7 (i.e., \(F = 7\)) feature channels, namely, RGB color values and the first- and second-order horizontal and vertical derivatives. Taking bee images for example, each data point in the dataset \(D\) is an image patch sampled as follows. We uniformly sample the patches with centroids located along the contour (see the light blue contours in Fig. 3.5) as examples of corresponding class \(c\), and the patches with centers inside or outside the contour as examples of the background. The dimension of a patch is \(m \times m \times 7\), where \(m\) is the patch size. A class label \(c\) is assigned to every data point. As shown in Fig. 3.5a, a class label \(c \in [0, N]\) of each patch (colored square) is the index of its closest tip (colored circle) along the contour: \{1: right antenna, 2: left antenna, 3: tongue\} and 0 as the background. A similar sampling strategy is applied for ant images, as shown in Fig. 3.5c, \(N = 3\) for bee and \(N = 2\) for ant. We classify the left or right antenna to balance the distribution of classes since the samples of the tongue are much fewer than the left antenna or right antenna. According to the width of the insect antenna (which is around 10 pixels), we set \(m\) of a patch to 16 since this size of patch provides a good balance between discriminability and repeatability.

To train the forest \(R\), we follow the framework presented in [10]. For each non-leaf node of a tree, a randomly generated binary test is used to split the training data into two sets. The binary test \(\phi\) is defined as:

\[
\phi = q_k(u) - q_k(v) > \tau,
\]  \(3.8\)
where $q_k$ indicates the feature channel, $u$ and $v$ are two positions within the patch boundaries, and $\tau$ is a threshold. A patch is sent to the right child when it satisfies the test $\phi$; otherwise, it is passed to the left child. During training, a set of test $\{\phi^m\}$ is proposed with random values of $k$, $u$, $v$ and $\tau$. The optimal binary split $\phi^*$ of each non-leaf node is selected by finding the $\phi$ giving the largest information gain:

$$
\phi^* = \arg \max_\phi \left( H(U) - \sum_{s \in \{l, r\}} \frac{|U_s(\phi)|}{|U|} H(U_s(\phi)) \right),
$$

(3.9)

where $H$ is the Shannon entropy, such that $H = -\sum_{c \in C} p(c) \log(p(c))$. $p(c)$ is the proportion of samples of class $c$ in the current node. $U$ denotes all patches in the current node and $U_s$ ($s \in \{l, r\}$) denotes the subset of patches that is passed to the left/right child node.

The forest $R$ is constructed with each leaf $L_t$ created when the maximum depth is reached, or a minimum number of patches remain. Each leaf stores the patches from $D$ that end here. Each patch of a given test image $I$ passes down a tree and ends in a leaf $L_t(y_i)$, which gives the class probabilities $p_t(c|q(y_i))$. A class label $c$ is assigned to each pixel with the highest $p(c|q(y_i))$, which is computed as:

$$
p(c|q(y_i)) = \frac{1}{T} \sum_{t=1}^{T} p_t(c|q(y_i)),
$$

(3.10)

where $T$ is the number of trees.

Having the class label $c$ of a patch, we set $p(c|q(y_i)) = 1, \forall c = 1, 2$ since the RF model may not correctly differentiate the left or the right antenna. The value of $p(y_i)$ is set to $p(c|q(y_i))$. 

Figure 3.5.: The image sizes of Dataset A, B and C are (a) $275 \times 235$, (b) $350 \times 320$ and (c) $415 \times 420$ in pixels, respectively. The patch size (denoted by a square) is $16 \times 16$ pixels. The images have been scaled for better visualization.
3.3.3. Final landmark localization

According to Eq. (3.6), the posterior probability of configuration $p(X|I)$ is computed as the product of $p(I|y_n)$ and $p(y_n)$. To localize a landmark, a simple strategy by selecting the pixel with the highest probability may fail due to the outlying pixels. Instead, we use mean shift [18] with a flat kernel to find the modes of probability mass for each part and assume that the seeking result lies in the connected component that contains the landmark. Finally, the landmark is estimated as the ending pixel within the connected component.

3.4. Experiments

As noted in [85], the state-of-the-art tracking algorithms do not perform well when applied to our task. In this experiment, as our method combines the strength of the traditional regression forest [33] and the pictorial structure model, it is compared with the methods directly applying these two concepts.

3.4.1. Datasets and evaluation metric

In this experiment, our method was evaluated on three challenging datasets of individual insects (i.e., bees and ants) during a behavioral experiment, among which two datasets of individual bees were recorded in different light conditions or other experimental settings. For example, Dataset A contains images from a video recording a bee in different trials of experiments, while Dataset B shows various bees in different trials. The image data come from our biological partner:

- Dataset A (bee): 5633 training images, 2788 testing images
- Dataset B (bee): 3625 training images, 9003 testing images
- Dataset C (ant): 215 training images, 238 testing images

The spatial resolution is 39 pixels per $\mu$m for bee images and 22 pixels per $\mu$m for ant images. More details about the three datasets are shown in Fig. 3.5. The ground truth (the tip position of present parts) of all images is labeled by human annotators.
As our method directly estimates the position of landmarks, it does not produce bounding box (BB) hypotheses. Some popular pose estimation metrics, such as average precision of keypoints (AFK) [106], require ground truth BBs for evaluation and thus are not suitable for our method. The results of our method are compared with the two aforementioned methods as well as ground truth landmark positions, which are manually annotated by a human. We compute the rate of false positives (FPs) and false negatives (FNs) for inferring the presence of each landmark to validate the adequacy of our geometric model, e.g., a FP of the tongue indicates that the tongue is inferred to be present while it is actually absent. The accuracy of localization is measured by the average Euclidean distance in the pixels between the results and the ground truth.

3.4.2. Implementation details

For learning the geometric model, we found $K = 100$ nearest neighbors to construct $K$ in Section 3.3.1. Parameters number of trees $T$ and tree depth $d$ are determined by 10-fold cross-validation. The pixel error is used as the evaluation metric. Given a specific value of $T$, e.g., $T = 6$, the pixel error is computed as averaging the experiment results of $\{(T = 6, d = 3), ..., (T = 6, d = 8)\}$. $T$ is set to the value that provides the lowest pixel error. The value of $d$ is determined by similar strategy. According to our experiment results (Fig. 3.6 and Fig. 3.7), we construct 10 trees for the RF and each tree has the depth of 5. Each tree converges until the maximum tree depth is reached or the amount of remaining patches is less than 50. For experiments of bee, 12000 (3000 patches for each class) randomly selected patches are used to train a tree. The bandwidth of the meanshift kernel is set as 0.05 for all images.

The complexity of the algorithm is measured by the processing time. For constructing a random forest, we use the code from the Hough forest [33]. Using a MATLAB implementation of our method, testing takes approximately 4 seconds per frame on an Intel i7 machine with 8 GB RAM. It takes approximately 1 second for the RF classification and 3 seconds for computing Eq. (3.6) and the final landmark localization.
3.4. EXPERIMENTS

(a) Left antenna

(b) Tongue

(c) Right antenna

Figure 3.6: Number of trees.
3.4. EXPERIMENTS

(a) Left antenna

(b) Tongue

(c) Right antenna

Figure 3.7.: Depth of trees.
3.4.3. Results and discussion

For quantitative evaluation, Table 3.1 shows the average position error (pixels) in three datasets. The average position errors of each part are 10.2, 5.3 and 18.0 pixels. These errors are rather small compared to the size of an insect head (shown as Fig. 3.5). The position error on Dataset C is larger than the others because the images have more severe motion blur (e.g., the right antenna in Fig. 3.11f), and the exact positions of the landmarks are ambiguous.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>position error (pixels)</th>
<th>FN (%)</th>
<th>FP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10.2</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>5.3</td>
<td>3</td>
<td>0.2</td>
</tr>
<tr>
<td>C</td>
<td>18.0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

To validate the advantage of our method over RF and the typical PS model, we compare the three methods on Dataset A, which is the most challenging dataset due to the complex background. The typical PS model in Eq. (3.1) assumes that all landmarks are present, and the spatial relations between the parts are learned from all the training samples. As shown in Table 3.2, this method produces a high FP rate when inferring the presence of the tongue. With the learned geometric model in Eq. (3.2), our method achieves a significant improvement over both the PS model and the RF in terms of both the localization precision and the ability to disambiguate different objects.

<table>
<thead>
<tr>
<th>Methods</th>
<th>L. antenna</th>
<th></th>
<th></th>
<th></th>
<th>Tongue</th>
<th></th>
<th></th>
<th></th>
<th>R. antenna</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pos. error</td>
<td>FN(%)</td>
<td>FP(%)</td>
<td>pos. error</td>
<td>FN(%)</td>
<td>FP(%)</td>
<td>pos. error</td>
<td>FN(%)</td>
<td>FP(%)</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>10.2</td>
<td>4</td>
<td>0</td>
<td>14.9</td>
<td>11</td>
<td>0</td>
<td>8.4</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>17.6</td>
<td>2</td>
<td>0</td>
<td>18.4</td>
<td>4</td>
<td>13</td>
<td>16.7</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RF</td>
<td>24.5</td>
<td>2</td>
<td>0</td>
<td>26.3</td>
<td>26</td>
<td>44</td>
<td>25.3</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3.8 - 3.10 show the localization errors of each landmark from Dataset A for a more detailed discussion. We show nine image sequences where either the RF or the PS method produces large localization errors, while our method achieves very low position errors in most of the frames.

We visualize more results in Fig. 3.11 and Fig. 3.12 to discuss the advantage of our method in more detail. As shown in Fig. 3.11a and Fig. 3.11e, our method successfully localizes the two antennae even in complex backgrounds, while the RF regressor fails to distinguish the left antenna from the background noise and thus produces high position errors. In addition, the RF regressor may incorrectly recognize
3.4. EXPERIMENTS

Figure 3.8: Quantitative results by our method (green), RF (blue) and PS model (red).
3.4. EXPERIMENTS

(a) Test sequence $A_2$ (Left antenna)

(b) Test sequence $A_2$ (Tongue)

(c) Test sequence $A_2$ (Right antenna)

Figure 3.9: Quantitative results by our method (green), RF (blue) and PS model (red).
3.4. EXPERIMENTS

(a) Test sequence $A_3$ (Left antenna)

(b) Test sequence $A_3$ (Tongue)

(c) Test sequence $A_3$ (Right antenna)

Figure 3.10: Quantitative results by our method (green), RF (blue) and PS model (red).
an antenna as the tongue, indicating a high FP rate for the tongue in Table 3.2. In contrast, our method is able to disambiguate the tongue and the right antenna, even when they are very close to each other (see Fig. 3.11a).

Figure 3.11.: Qualitative results by our method (the first row), RF regressor (the second row) and RF classification (the bottom row).

Fig. 3.12 shows the advantage over typical PS models in inferring the number of landmarks present. Without inferring the presence of a landmark, it will be localized at some position. A naïve approach for rejecting the potentially incorrect position is that a landmark falling inside the region of the insect head will be rejected, e.g., the tongue has been rejected in Fig. 3.12a and Fig. 3.12d. However, the method cannot deal with more general cases as shown in Fig. 3.12b and Fig. 3.12e. Moreover, the spatial relations between parts learned from all the training data provide minimal information in our case, since the possible positions of the antennae in all training images are nearly uniformly distributed. As shown in Fig. 3.12a and Fig. 3.12d, the left antenna is incorrectly located in the right antenna. In contrast, our method is capable of inferring the absence of a landmark by the learned geometric model.

To validate that the localization precision of our method is comparable to human annotators, we also compare the pixel error between two annotators and our method. The means and standard deviations of the Euclidean distance between the landmark positions of human annotator A versus the results of our method (blue), human annotator B versus our method (green) and between only the human raters
3.5. Summary

In this chapter, we presented a new PS model-based algorithm exploiting local appearance and global geometric structure of an insect to infer its pose from a single image. Our method is a data-driven approach to incorporate geometric constraints.

We also developed a graphical user interface for biological research. As shown in Fig. 3.14(a), once a video is selected for processing, the bee head centroid and mandible must be marked so that the software can automatically transform the bee to the standard position and scale. Detection results are displayed in a user-friendly way, and users are allowed to modify the detection results as they want (Fig. 3.14b). The proposed software was used by G. Galizia for his study on olfactory research with bees.

3.4.4. Software

Figure 3.12: Qualitative results by our method (the first row), PS model (the second row) and RF classification (the bottom row).

(red), are illustrated in Figure 3.13. The results show that the estimation of our method is comparably accurate to human annotators in most cases.
Figure 3.13: Comparison between human annotators and our method: The means and standard deviations of the Euclidean distance between the landmark positions of human annotator A vs. the results of our method (blue), human annotator B vs. our method (green) and between human raters only (red).

Figure 3.14: Software GUI.
The model parameters are learned from the training data. Our method addresses the issue of interchangeable estimations by using only the RF model for landmark detection and presents a good interplay between the RF and PS model. The performance of our method has been validated on three large challenging datasets of different types of insects and has achieved comparable position accuracy to that of human annotators for the frames containing no stick. The proposed method requires a relatively small number of training data and is fast both in training and testing.
4.1. Overview

In the previous chapter, we described a random forest (RF)-based framework for insect pose estimation. It utilizes RF as the classifier and represents the global structure of insects as geometric features (e.g., edge histogram descriptor). As mentioned in Chapter 1, bees to be analyzed may be fed sugar water with a stick. When the testing image frames contain no stick, the RF-based algorithm performs at a level of accuracy comparable to that of human annotators. However, as shown in Fig. 4.5, the RF classifier is unable to reliably distinguish the stick from the bee’s body parts since they have a similar visual appearance. Moreover, the geometric features cannot fully represent the bee’s global structure in a manner that makes such disambiguation possible (Fig. 4.2).

Recently, deep convolutional neural networks (CNNs) have gained considerable prominence in computer vision. CNN-based approaches such as GoogLeNet [92] and VGGNet [88] have shown notable performance compared to other methods that have been applied to pose estimation [16] and image classification [52]. In addition, the optimization of model architectures [42], ease of algorithm training [55], and relatively modest computational hardware requirements [110] help make CNNs suitable
4.1. OVERVIEW

Figure 4.1.: Limitations of the RF-based framework 1. RF classifier cannot distinguish bee body part from the stick since they have similar appearance. (A) Input image. (B) RF output

Figure 4.2.: Limitations of the RF-based framework 2. Geometric features used in the RF-based framework may not be able to fully represent the bee’s global structure, these images have similar features

In this chapter, to address the challenges in bee pose estimation as well as the limitations of our RF-based framework, we present a PS model-based framework that utilizes CNN for bee pose estimation. In this unified framework, we formulate the problem as one involving the localization of landmarks of bee body parts. As shown in Fig. 4.3, the proposed framework seeks to learn a mapping from the global structure and local appearance of a bee to its corresponding pose. This is done along two parallel processing pathways. The first pathway involves a new VGGNet-based CNN architecture that is designed for localizing multiple landmarks (Fig. 4.3b-c). In the second pathway, we utilize a fine-tuned GoogLeNet to extract the global features of the bee (Fig. 4.3d) and approximate the bee pose prior (Fig. 4.3e-f) based on the features. By integrating both the detected local landmarks and the global constraints of the bee, the pose of the bee can then be estimated. The overall framework is shown in Fig. 4.3.
4.2. RELATED WORK

Figure 4.3: The flowchart of the proposed framework. (a) Input image; (b) Modified VGGNet (see Fig. 4.6 for details); (c) Confidence map; (d) Fine-tuned GoogLeNet; (e) KNN search; (f) Probability mass of KNN result; (g) Final tip localization.

4.2. Related Work

Since our presented framework unitizes VGGNet and GoogLeNet, we give a brief introduction of these two popular network architectures in this section.

4.2.1. VGGNet

VGGNet [88] was proposed by the Visual Geometry Group (VGG), Oxford, and it received second place in the Large Scale Visual Recognition Challenge (ILSVRC) 2014 [80]. Instead of using large kernel-sized filters (e.g., $7 \times 7$ or $5 \times 5$), VGGNet employs a stack of $3 \times 3$ kernel-sized filters followed by max pooling. For a certain receptive field (the effective area size of the input image on which the output depends), multiple stacked smaller size kernels are better than the one with a larger size kernel because 1) multiple nonlinear layers increase the depth of the network, which enables it to learn more complex features, and 2) they have fewer parameters. The overall network architecture is shown in Fig. 4.4. When the values of $N_1, N_2$ and $N_3$ are set to three, the net is called VGG-16, which means that this net has 16 weighted layers. Another popular variation is the VGG-19 net, where $N_1, N_2$ and $N_3$ are set to four. The authors in [88] also concluded that increasing the network depth could achieve better performance.

Due to the relatively straightforward structure, VGGNet is widely used in
other applications as the basic model. Notable examples include human pose estimation [16], object detection [37, 77] and image segmentation [60, 79].

4.2.2. GoogLeNet

GoogLeNet was the winner of ILSVRC 2014. Unlike other popular CNN architectures [52, 88] that are composed of stacked convolution layers followed by one or more fully connected layers, GoogLeNet introduced a new structure named the ‘inception module’ to reduce parameters and increase the learning ability. The number of parameters of each layer is reduced by decreasing the number of filters in each layer; we refer the reader to [92] for more details. Another advantage of the inception module is that it allows multiscale image feature learning at the same time. As shown in Fig. 4.5a, image features are captured by parallel kernels with different sizes; thus, the image details are kept while larger image areas are covered. The main structure of GoogLeNet is shown in Fig. 4.5b. It consists of 3 successive convolutional layers followed by a stack of 8 inception modules. Only one fully connected layer is involved in the network, and the number of trainable layers is 22.
4.3. Bee Pose Estimation

Following Section 3.2, we assume that a bee can be decomposed into a set of parts, and the pose of a bee is defined as $X = \{x_1, x_2, ..., x_N\}$, where the state of part $n$ is defined as $x_n = \{y_n, t_n\}$, $N$ is the number of insect body parts. $y_n = \{x_n, y_n\}$ denotes the position of a tip in the image coordinate system and $t_n = \{0, 1\}$ denotes the state indicating the presence of part $n$.

We formulate the bee pose estimation problem as finding the maximum posteriori of a pose given an image $I$, i.e., $p(X|I)$, which can be approximated as

$$p(X|I) \propto p(I|X)p(X)$$ (4.1)

where $p(I|X)$ is the likelihood of the image evidence given a particular pose, and $p(X)$ corresponds to a kinematic tree prior according to the PS model. We present a CNN-based framework for solving Eq. (4.1).
4.4. Methodology

4.4.1. Confidence map generation

CNNs generally comprise several different types of layers and are efficient when applied to learning hierarchical feature representations of input data. General image features can be learned and captured by lower layers, with specific features and (perhaps high-level) information relevant to the input dataset acquired and represented by higher layers in the network. While the general architecture of a CNN comprises a number of fully connected layers followed by convolutional layers to capture the global information of an image, fully convolutional networks were introduced in [60] and proposed using convolutional layers in place of all the fully connected layers. For confidence map generation, we use a fully convolutional VGGNet [88] as the base model. In this model, all tips share the weights of lower layers, and the feature maps of individual tips and the background are learned from the corresponding layers. Layers for extracting feature maps of each tip as well as the background are independent, given that the movements of each tip are uncorrelated with those of others. The feature maps of each tip provide detailed information of the insect pose and tip appearance, which can be regarded as the response of filters for a specific tip of the whole image. The final confidence map of each tip is predicted from the corresponding feature map through a 1×1 convolution across the feature map. The confidence map has four channels; the first three channels correspond to the three tips, and the last channel corresponds to the background. The loss function is defined as:

\[
\text{Loss} = \sum_x \sum_y \sum_n g_n(x, y) \log \left( \frac{e^{p_n(x, y)}}{\sum_{n'} e^{p_{n'}(x, y)}} \right),
\]

where (x,y) defines a location on the confidence map, and \( n \in \{1, 2, 3, 4\} \) is the class label. \( g_n(x, y) \) is the ground truth label of position (x,y). \( g_n(x, y) = 1 \) if the class label of (x,y) is \( n \), otherwise \( g_n(x, y) \) is set to 0. \( p_n(x, y) \) is the probability of location (x,y) belonging to class \( n \).

Details of the modified VGGNet that we used are shown in Fig. 4.6. This is a VGG-16 network that was pretrained on ImageNet [24]. To preserve the high resolution of the confidence map, we adopt the approach presented in [16] to remove the last two pooling layers from the VGGNet, which is equivalent to a downsampling
4.4. METHODOLOGY

Figure 4.6: Details of the modified VGGNet in our framework. All the tips share weights up to \(7 \times 7\) conv layer, features for specific tips and background are learned from corresponding path.

of the image by a factor of eight (i.e., given an input image of size \(352 \times 352 \times 3\), the size of each confidence map channel will be \(44 \times 44\)). Resizing the confidence map to the size of the original input image is thus required; strategies for resizing will be introduced in sec. 4.5.3.

While confidence maps provide the probabilities of each tip position, the visual similarity of the tips and background could result in substantial noise. As shown in Fig. 4.7(a,b), although all tips of the antennae are correctly highlighted, false positives were detected in certain areas of the background and the bee’s head. In addition, we cannot detect the presence of a tip from the confidences map alone (Fig. 4.7c) if the correctness of confidence maps cannot be guaranteed. Motivated by the PS model by which the pose of an object can be inferred by combining part confidence maps, i.e., \(p(I|X)\), and tree prior \(p(X)\), we utilize GoogLeNet and present a conceptually simple method to approximate \(p(X)\).

4.4.2. Confidence refinement

GoogLeNet was originally trained on ImageNet and thus cannot fully represent the kind of image characteristics that are present in our bee image dataset. To transfer the representation of ImageNet to our dataset, we attempted to localize the two antennae as an auxiliary task. With fine-tuning, the Euclidean distance between the prediction and labels is minimized, and the refined GoogLeNet was then used for extracting features representing the whole image. In our case, the training image
4.4. METHODOLOGY

Figure 4.7: Examples of confidence maps of different tips. (a) Confidence maps of the right antenna tip. (b) Confidence maps of the left antenna tip. (c) Confidence maps of the tongue tip.

labels were the tip positions of the two antennae. To fine-tune the model, the lowest three convolutional layers ("conv1/7×7_s2", "conv2/3×3_reduce", "conv2/3×3") were kept invariant, and the remaining 8 inception modules were retrained on our dataset. The weights of the net were initialized by the model that was pretrained on ImageNet. The feature vector \( f \) (with 1024 dimensions) representing a whole image was extracted from the "pool5" layer.

If we assume that similar poses will have similar features, the data point of the test image should then lie close to the training images with similar poses in the feature space. As illustrated in Fig. 4.3(a-e), after the feature vector \( f \) of a test image was extracted from the fine-tuned GoogLeNet, the \( K \) nearest neighbors (KNN) of that feature were found in the feature space. The corresponding training subset \( S \) comprising the KNN result can be regarded as having a similar pose as the test image. Fig. 4.8 presents three examples of this process. Images in the first column are the input, and images between the two black bars are the results of KNN.

As figured out in Chapter 4, movements of bee body parts are independent,
4.4. METHODOLOGY

Figure 4.8: Examples of KNN results, the first column is the input images and the last column shows tips positions of neighbours, columns in the middle are the KNNs of the input image. Green crosses represent right antenna tips, blue crosses represent left antenna tip and red crosses are the tongue tips. (a) The first example. (b) The second example. (c) The third example.

the kinematic prior \( p(X) \) of a bee is approximated as

\[
p(X) \approx p(x_1, x_2, ..., x_N | f) = \prod_{n=1}^{N} p(x_n | f),
\]

where \( p(x_n | f) \) is the \( n \)th part prior of the bee based on the feature vector \( f \). By further assuming that tips locations (e.g., last column of Fig. 4.8) follow a Gaussian distribution, the tip probability is computed as

\[
p(x_n | f) = \mathcal{N}(y_n^m | \mu_n^m, \Sigma_n^m),
\]

where parameters \( \mu_n^m \) and \( \Sigma_n^m \) are learned from the training subset \( S \).

4.4.3. Localization

Given \( p(I|X) \) and \( p(X) \), the posterior probability \( p(X|I) \) is approximated by Eq. (4.1). We use mean shift [18] with a flat kernel to locate the tip position of each part. The absence of tip(s) can be inferred by the number of absent tips in \( S \), i.e., if the number of absent tips of tongue from \( S \) result is greater than a threshold \( \tau \), the tongue is deemed as absent from the input image. Fig. 4.8 shows how we infer the presence of the tongue. In Fig. 4.8(a) and Fig. 4.8(c), the KNN results contain no tongue, so the tongue is absent in these two images. Similarly, the tongue is present
4.5. Data Preparation

We collected three challenging datasets (A, B, and C) of individual bees from biologists who perform behavioral experiments. The three video image datasets were recorded under different light conditions and experimental settings. Dataset A contained 7886 type A images, Dataset B contained 12610 type B images, and Dataset C contained 1440 type C images. Samples images of the three datasets are shown in Fig. 4.9.

4.5.1. Training data preparation

We randomly selected 8705 images as training data, which contained frames from two types of bee videos, 5098 images of type A and 3607 images of type B. Note that the number of tips in those images is variable; in some images, the tongue might be missing, or certain tips might be absent or indiscernible due to blurred images or variations in the lighting conditions. To train the VGGNet, we split the images into a training set and a validation set. The training set contained 7224 images, and the remaining 1481 images were used as validation images. We applied rotation (from -165 to 180 degrees with a step size of 15 degrees) and flipping to the training set, which led to 347788 images in the final training set. All 347788 images were resized to $336 \times 336$.

As the number of images varied greatly across different types of bee behaviors and movements, we augmented the training data to balance the number of exemplars across types, and the final training set comprised 9109 images, all resized to $224 \times 224$, for fine-tuning the GoogLeNet.
4.5.2. Training label generation

To train the VGG-based network, localization of the bee tips was framed as a classification problem. Each pixel was assigned a class label tensor, and the training label for the input image had a size of $42 \times 42 \times 4$. The first 3 channels represented 3 tips ($c_1$: right antenna, $c_2$: left antenna, $c_3$: tongue) and the last channel represented the background. The values of the first three channels were all set to zero if the corresponding tip(s) were missing. Otherwise, locations $c_i(x, y)$ would be set to 1, where $i \in \{1, 2, 3\}$ and $(x, y)$ are the corresponding tip positions in the training images times $\frac{1}{8}$. For the last background channel, all values were set to 1 except the regions near the tips. Tip positions were set to 3, and the four neighboring pixels (left, right, up, down) were set to 2.

4.5.3. Test data pre-processing

As a preprocessing step, we scaled and translated the test image to align with the training images. For the test images of our modified VGGNet, each image was first padded to an equal width and height (Fig. 4.8c) if it was not square. The image was further resized to $368 \times 368$ if the size of the (padded) image was less than $350 \times 350$ or greater than $380 \times 380$. Given an input image of size $224 \times 180$, it would first be padded with zeros to have a size of $224 \times 224$. Then, the padded image would be further resized to $368 \times 368$, and thus, the resulting confidence map would be of size $46 \times 46 \times 4$. Each channel of the confidence map should be resized to have the same dimensions as the input image, and this was achieved in our experiment using bicubic interpolation. Input images of our fine-tuned GoogLeNet were directly resized to $224 \times 224$ without zero-padding.

4.6. Experiment

We conducted a self-comparison and a comparison with state-of-the-art methods on the test data of three datasets. Dataset A contained 2788 test images, Dataset B contained 9003 test images, and Dataset C contained 1440 test images. Images in Dataset B and C contained no tongue, and thus, we only compared the results for two antennae. Note that no image from Dataset C was involved in the training pro-
4.6. EXPERIMENT

cess; this dataset is mainly used to evaluate the generalization capability of different methods.

The averaged Euclidean distances in pixels between the results and the ground truth is employed as the metric to evaluate the performances of different methods. In addition, the false positive (FP) and false negative (FN) rates were computed to determine the adequacy of our method. A false negative, in this case, could be the missed detection of the right antenna when it was, in fact, present in the image, and a false positive could be the detection of tips from spurious regions in the image background. The localization accuracy was evaluated by the average Euclidean distance in pixels between the estimated tip positions and their true positions.

4.6.1. Implementation details

For the KNN search, the value of $K$ was set to 30, and the threshold $\tau$ for detecting absence of tip was set to 20. The bandwidth of meanshift kernel was set to 0.05 for all images.

We use the Caffe [48] implementation of VGGNet and GoogLeNet and all models are trained on a single NVIDIA GTX 960. The lower layers of our VGG-based net used pre-trained weights for initialization, and were then fine-tuned with an initial learning rate of 0.001. The weights of other layers were randomly initialized with an initial learning rate of 0.01. All the layers were fine-tuned together.

The base learning rate for fine-tuning the GoogLeNet was set to 0.005, and the learning step size was set to 32000. The maximum number of iterations was 5000000.

4.6.2. Self-comparisons

First, the results generated from the individual pathways in our framework are compared with our method to illustrate the need for two pathways. To explain why we chose GoogLeNet for KNN search, we also compare the performance of the fine-tuned GoogLeNet, the pretrained GoogLeNet, the fine-tuned VGGNet and the pretrained VGGNet in our framework.
4.6. EXPERIMENT

Figure 4.10: Hard cases for detection presences of tongue. (a) A bee with tongue absent. (b) A bee with tongue present.

For simplicity, we use **Ours** to denote the proposed framework, Conf to denote the upper pathway in our framework, KNN to denote the bottom pathway in our framework, Ours(PrG) to denote features extracted from the pretrained GoogLeNet, Ours(FV) to denote features extracted from the fine-tuned VGGNet, Ours(PrV) to denote features extracted from the pretrained VGGNet, L. Ante. to denote the left antenna, Ton. to denote the tongue, R. Ante. to denote the right antenna and Pos. Err to denote the position error in the tables.

**Two pathways vs. Individual pathway**

Table 4.1 shows the comparison results on the three datasets; we can see that our method produced fewer pixel errors than Conf and KNN for almost all tips. Though some FN rates of Conf were less than **Ours** (which are equal to KNN), our method has a better generalization capability (results on Dataset C). The reason for the higher tongue FN rates of our method than Conf is because when the tongue is not obvious (Fig. 4.12a), the KNN results are the images without the corresponding tip (Fig. 4.12c). In contrast, since our training data lacks visibility information, using only the confidence map could not distinguish the cases shown in Fig. 4.10; this is the reason for the high tongue FP rate of Conf on Dataset A.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pos. Err</td>
<td>FN(%)</td>
<td>FP(%)</td>
<td>Pos. Err</td>
</tr>
<tr>
<td>A</td>
<td>Ours</td>
<td>5.6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Conf</td>
<td>20</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>KNN</td>
<td>13.2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>Ours</td>
<td>10.2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Conf</td>
<td>24</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>KNN</td>
<td>21.7</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>Ours</td>
<td>8.2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Conf</td>
<td>26.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>KNN</td>
<td>19.3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Model selection

As shown in Table 4.2, applying different feature extraction methods in our framework (i.e., Ours, Ours(PrG), Ours(FV), and Ours(PrV)) had similar performances on Datasets A (Fig. 4.12c-f) and B. However, the comparison results on Dataset C indicated that fine-tuned GoogLeNet for feature extraction in our framework generalized better than other deep models.

Table 4.2: Self-comparison 2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Pos. Err</td>
<td>FN(%)</td>
<td>FP(%)</td>
</tr>
<tr>
<td>A</td>
<td>Ours</td>
<td>5.6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Ours(PrG)</td>
<td>5.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Ours(FV)</td>
<td>5.4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Ours(PrV)</td>
<td>5.4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>Ours</td>
<td>10.2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Ours(PrG)</td>
<td>9.26</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Ours(FV)</td>
<td>11.0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Ours(PrV)</td>
<td>10.8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>Ours</td>
<td>5.2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Ours(PrG)</td>
<td>6.8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Ours(FV)</td>
<td>24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Ours(PrV)</td>
<td>27.4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 4.11 shows a typical exemplar where pixel errors of Ours(PrG), Ours(FV) and Ours(PrV) are high (see table 4.2C), due to the presence of shadows with similar appearances to the antenna in the background, resulting in a poorer confidence map (Fig. 4.11b). The accuracy of the final output highly depends on the KNN result. If most tip positions of the KNN result do not lie close to the true tip position, this may lead to large pixel error (Fig. 4.11d-f). The KNN result of Ours(PrG) (Fig. 4.11d) is poorest for the right antenna because most right antenna tip positions lie too far away from the true positions; this is the reason why this setting produces the highest right antenna pixel error (54.3). Similarly, large pixel errors of left antenna of Ours(FV)(24) and Ours(PrV)(27.4) are due to the same reason.

Table 4.3: Comparison with state-of-the-art

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Pos. Err</td>
<td>FN(%)</td>
<td>FP(%)</td>
</tr>
<tr>
<td>A</td>
<td>CNN(Our)</td>
<td>5.6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RF(our)</td>
<td>13.8</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Shen(2015)</td>
<td>6.5</td>
<td>6.1</td>
<td>1.5</td>
</tr>
<tr>
<td>B</td>
<td>CNN(Our)</td>
<td>10.2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>RF(our)</td>
<td>11.6</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Shen(2015)</td>
<td>8.5</td>
<td>2.1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>CNN(Our)</td>
<td>5.2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RF(our)</td>
<td>11.6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Shen(2015)</td>
<td>5.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2nd Annot.</td>
<td>2.8</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
4.6.3. Comparison with state-of-the-art

The performance of our method is compared with Shen(2015) [85], our RF-based method on three datasets. RF(our) is used to denote the RF-based method in tables.

The first two rows of Table 4.3 show the comparison results of the our CNN-based and RF-based methods on Dataset A, which is more challenging due to the complex background containing shadows and noise, making it difficult, even for human observers, to discern the presence of the tongue. For example, only a small part of the tongue is visible in some images (Fig. 1.1e), and in some cases, the antenna and tongue are overlapping. We can see that our methods produced fewer pixel errors than the RF-based method for all the tips and lower false negative rates for the antennae. Though the RF-based method is able to detect short tongues (Fig. 4.12k-l), which leads to a lower tongue FN rate than Ours, it also has a higher false positive (23%) rate.
4.6. EXPERIMENT

Figure 4.12: Illustration of high FN rates of our methods on dataset A. (a) Input image. (b) Confidence map. (b1) Detailed view of confidence map of tongue. (c) KNN result of Ours. (d) KNN result of Ours(Pr-G). (e) KNN result of Ours(FV). (f) KNN result of Ours(Pr-V). (g-j) Outputs of our method. (k) RF result. (k1) Detailed view of RF result of tongue. (l) KNN result of RF[84]. (m) Output of RF[84].

In Dataset B, the images without the stick have a clear background, while some shadows are present with the stick in some images. As shown in Table 4.3, our method performed better than the RF-based method in every metric except the false negative rate of Ours for the left antenna. The average pixel errors of both methods were rather small compared to the size of a bee head (120×160 pixels).

The comparison results on Dataset C show that our method generalized better than the RF-based method with an approximately 50% reduction in pixel errors on both objects (5.2 vs. 11.6 and 8.6 vs. 17.9). Moreover, we also compared two ground truths of Dataset C that were generated by two annotators. The last row of Table 4.3 shows that even the results generated by the two human annotators were different by more than 2 pixels. Fig. 4.13 shows a detailed view of the comparison results. We can see that in most frames, Ours was able to achieve comparable position accuracy to that of human annotators.
4.6. EXPERIMENT

(a) Left antenna.

Figure 4.13.: Comparison of Ours result with ground truth generated by annotator 1 (blue), annotator 2 (green), and the comparison of two ground truth (red). In most frames, Ours was able to achieve comparable position accuracy as that of human annotators.

(b) Right antenna.
Compared to our CNN-based method, Shen(2015) can provide slightly better performance on all three datasets. However, this method requires quite a lot of human interaction, and it takes around one minute to process an image, which is unfeasible for large datasets.

4.7. Summary

In summary, we presented a new algorithm based on the PS model for estimating a bee pose from a single image. The proposed framework utilizes the powerful representation capability of CNN to learn the mapping from the local appearance and global structure of a bee to its corresponding pose. Our method is able to localize a varying number of targets in complex backgrounds, especially for samples where the bee was being fed sugar water with a stick. The performance of our method has been validated on three challenging datasets of bees and outperforms the existing bee pose estimation algorithm. We evaluated feature extraction performances of four deep models and found that pretrained models can characterize data well and that fine-tuned models lead to a better generalization capability.
CHAPTER

Estimating 2D Multi-Hand Poses From Single Depth Images

5.1. Overview

In earlier chapters, we introduced two PS model-based frameworks for bee pose estimation. Since human fingers and bee extremities share some similarities, algorithms for one object can be extended to serve the other. In this chapter, we show that our idea can be generalized for hand pose estimation. Moreover, with slight modifications in the problem formulation and the advantages of Mask R-CNN [40], our proposed framework allows simultaneous detection of multi-hand instances and localization of hand joints.

Accurate hand pose estimation from depth images or videos plays an essential role in human-computer interaction, as well as virtual and augmented reality. As mentioned in Section 1.2, challenges with estimating hand poses can arise from self-similarity, self-occlusion, and large viewpoint variation. Although considerable progress has been made in this area [34, 67, 90, 93, 96, 107, 112], multi-hand pose estimation is still mostly unsolved. A good solution, however, would provide more flexibilities and possibilities in many HCI applications.

Compared to single-hand pose estimation, estimating poses of multiple hands from a single depth image is more difficult because it requires the correct detection of all hand instances while also precisely localizing the corresponding hand joints. A straightforward way to solve this problem is to follow the common two-stage strategy
that first uses a traditional method (e.g., a random forest [10]) to extract regions of an image that contain a hand object. Having these regions, single-hand pose estimation methods are applied to each of them. However, a general framework with more powerful detectors that can fulfill multi-hand instance detection and hand joint localization simultaneously could be more reliable and convenient in real-world applications.

Recently, convolutional neural networks (CNNs) have become a mainstream technique in computer vision tasks such as image classification [52], pose estimation [16, 35] and object detection [77]. In [40], a multitask learning framework named Mask R-CNN was proposed for simultaneous object detection and instance segmentation. Mask R-CNN is a generic multitask learning pipeline that can be generalized to multi-human pose estimation. Because minimal domain knowledge for human pose estimation is exploited, Mask R-CNN is not explicitly applicable to model joint relationships. Moreover, as noted in [14], keypoints might not be localized accurately in complex situations.

In this chapter, we propose a PS model-based framework to address the limitations of methods based on Mask R-CNN by refining the output from these networks with a learned global structure of the current hand pose during the test stage. The overall structure of our proposed method is shown in Fig. 5.1. Our framework is composed of two stages: first, Mask R-CNN is adopted to predict possible keypoint locations (Fig. 5.1c) and segments each hand from the given images (Fig. 5.1d). Then, we utilize the instance segmentation output of Mask R-CNN to approximate the pose prior of each hand (Fig. 5.1d-g) and add this constraint to the pose space. Finally, keypoint locations are estimated via combining local information and global constraints (Fig. 5.1f).

The main contributions of our work are the following:

- a new method for 2D multi-hand pose estimation from a single depth image.
- a PS model-based method to identify global structure constraints of a hand pose online and two ways to implement the method.
- two multi-hand datasets, dexter2Hands and NYU2Hands, which are based on the popular single-hand datasets dexter1 [90] and NYU hand pose [96], respec-
5.2. RELATED WORK

Figure 5.1: Example of how our method localizes joints of left thumb finger and right index finger. Given an input image (a), we first use Mask R-CNN (b) to detect bounding boxes, possible joint positions (c), and hand segmentations (d). Then, we extract global features (e) of each hand from Mask R-CNN and find hand poses similar to input hands from training data (f). Afterwards, we compute global constraints of input hands (g). Final hands joint positions are localized by combining the local information and global constraints (h).

5.2. Related Work

Since estimating human and hand pose share some similarities, in this section we first review some relevant multi-human pose estimation methods. Then the series of R-CNN frameworks are introduced, where the latest framework Mask R-CNN is served as the baseline of our research. Finally, we briefly introduce the popular CNN architecture ResNet, which is used as the CNN backbone in our experiments.

5.2.1. Multi-human pose estimation

In [27], a PS model-based framework was proposed for estimating the poses of multiple humans, but it relies on an additional human detector and simple geometric body part relationships. Similarly, the model proposed in [53] also requires a human detector for initial human hypotheses generation, and the estimation of keypoint positions and instances are divided into two stages. Unlike previous strategies that need to first detect people and subsequently estimate their poses, the method proposed in [75] utilizes CNN for body part hypotheses generation and is able to jointly solve the task of detection and pose estimation. This work was extended in [46] with stronger part detectors and more constraints in the problem formulation.
5.2. RELATED WORK

5.2.2. R-CNN family

R-CNN

In [38], a method named regions with CNN features (R-CNN) for object detection and semantic segmentation was proposed. R-CNN first utilizes a Selective Search [100] to generate a set of category-independent region proposals. These proposals are supposed to contain some objects and are resized into a standard size. Features of each region proposal are extracted from a CNN and are classified by an SVM. Additionally, bounding box regression is applied to those features to reduce the localization errors. Nonmaximum suppression is applied to those classified regions to obtain the final result. Though R-CNN achieves excellent object detection accuracy, it still has some drawbacks. (1) The method trains CNN, SVM and bounding box regressors separately. (2) Given a test image, each generated region proposal is fed into the CNN for feature extraction, so R-CNN takes 47 s for detection on a GPU. (3) The selective search algorithm is fixed, which could lead to poor region proposals.

Fast R-CNN

Fast R-CNN [37], which is the extension of R-CNN, solves some of the drawbacks of R-CNN. Similar to R-CNN, it still requires the selective search for region proposal generation. However, instead of extracting the features of every region proposal, Fast R-CNN first extracts the feature map for the whole image. Features of each region proposal are then extracted from the feature map and reshaped into a fixed size by the ROI pooling layer. The ROI pooling layer works by dividing the $h \times w$ ROI into an $H \times W$ grid of subwindows of approximate size $h/H \times w/W$ and then max-pooling the values. Each feature is fed into a sequence of fully connected layers that finally branches into two sibling output layers: one outputs softmax probabilities and one outputs per-class bounding box regression offsets. The loss function of Fast R-CNN is defined as:

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v),$$

(5.1)

where $L_{cls}(p, u) = -\log p_u$ is log loss for true class $u$ and $p$ is the class probability. $[u \geq 1]$ evaluates to 1 when $u \geq 1$ and otherwise 0, $\lambda$ is the hyper-parameter controls.
5.2 RELATED WORK

Figure 5.2: Faster R-CNN. Image from [77]

the balance between the two losses. The bounding box regression loss is computed as:

\[ L_{loc}(t^i, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t^i_i - v_i), \]  

(5.2)

in which

\[ smooth_{L_1} = \begin{cases} 
0.5x^2 & \text{if } |x| \leq 1 \\
|x| - 0.5 & \text{otherwise.} 
\end{cases} \]  

(5.3)

Faster R-CNN

Fast R-CNN achieves near real-time for object detection, when ignoring the time spent on region proposal. Therefore, Faster R-CNN [77], which uses a region proposal network (RPN) to replace the selective search, is proposed to handle the test-time computational bottleneck in its precedent. RPN is a network that can share weights with the Fast R-CNN detector. Given an input image, a network backbone of RPN and Fast R-CNN first extract the feature map of the entire image. Then, RPN adopts a sliding window approach to generate a set of proposals for each pixel of the feature map and predicts an objectness score (background or foreground) and bounding box position for each proposal. These bounding boxes with objects inside serve as the input to the Fast R-CNN detector. The overall structure of Faster R-CNN is shown in Fig. 5.2.
5.2. RELATED WORK

Figure 5.3: Mask R-CNN head architecture.

Mask R-CNN

Mask R-CNN is a generalization of Faster R-CNN that can be used for object instance segmentation and human pose estimation. It consists of two stages. In the first stage, candidate object bounding boxes are proposed by the RPN. In the second stage, the features of each candidate bounding box are extracted and classified, and bounding box regression, instance segmentation and key point detection are performed. Unlike the methods proposed in [19, 57, 72] whereby classification depends on mask prediction, Mask R-CNN applies a parallel strategy that can simultaneously solve tasks in stage two. The overall network architecture of Mask R-CNN contains a convolutional backbone used to extract features over the whole image and three parallel network heads: one for classification and bounding box regression, and two for the remaining tasks (Fig. 5.3).

5.2.3. ResNet

As mentioned in Chapter 4, increasing the depth of the network could lead to better performance. However, the authors in [41] determined that simply stacking more layers could cause the vanishing gradient problem. Instead, they introduced the residual module (Fig. 5.4) to handle this issue. The idea is that when increasing the network depth, the performance of the deeper network should not be worse than its shallower counterpart. This is guaranteed by the shortcut connection (Fig. 5.4 X
5.3. Multi-hand Pose Estimation

Mathematically, our objective is to estimate hand poses $\mathbf{P} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_M\}$ from a single image $I$, where $\mathbf{X}_i$ denotes the pose of an instance and $M$ is the number of instances in $I$. Following [3], we assume that a hand can be decomposed into a set of parts; the pose of a hand is defined as $\mathbf{X}_i = \{\mathbf{x}_i^n | 1 \leq n \leq N, \forall \mathbf{x}_i^n \in \mathbb{R}^3\}$, where the state of part $n$ is formulated as $\mathbf{x}_i^n = \{\mathbf{y}_i^n, t_i^n\}$. $\mathbf{y}_i^n = \{x_{i1}^n, y_{i1}^n\}$ is the position of the key point in image coordinate system and $t_i^n = \{0, 1\}$ denotes the state indicating the presence of part $n$.

We formulate the multi-hand poses estimation problem as finding the maximum posteriori of poses given an image $I$, i.e., $p(\mathbf{P} | I)$, which can be approximated as

![Residual module](image1)

Figure 5.4: Residual module.

![ResNet](image2)

Figure 5.5: ResNet.

identity). ResNet is now the most popular network architecture in deep learning, it consists of a stack of residual modules. The overall structure of ResNet is shown in Fig. 5.5. When $N$ is set to 16, 43 or 50, the net is called ResNet-50, ResNet-101 or ResNet-152, respectively.
5.4. MASK R-CNN FOR HAND POSE ESTIMATION

\[ p(P|I) \propto p(I|P)p(P), \]  
(5.4)

where \( p(I|P) \) is the likelihood of the image evidence given particular poses, and the \( p(P) \) corresponds to poses prior. We assume that all hands are independent for simplicity, Eq. 5.4 can be factorized as

\[ p(P|I) \propto \prod_{i=1}^{M} p(I|X_i)p(X_i), \]  
(5.5)

where \( p(I|X_i) \) is the likelihood of the image evidence given a particular pose, and \( p(X_i) \) corresponds to a kinematic tree prior according to the PS model, though this may not always hold when fingers of different hands are crossed. We propose a general framework based on PS model and utilize Mask R-CNN [40] to solve Eq. 5.5.

5.4. Mask R-CNN for Hand Pose Estimation

In this work, we use ResNet-50 [41] with feature pyramid network (FPN) [58] as the backbone to extract features from the entire image. For the network head, we follow the three-parallel-branches architecture presented in [40] whereby one branch is for bounding box classification and regression, one for instance mask prediction, and one for keypoint detection (Fig. 5.3). In general, given a training image, features of the entire image are first extracted by the ResNet-FPN backbone. Based on the features, RPN generates a set of ROIs. Each positive ROI is fed into three parallel branches of the network head: one branch for bounding box classification and the other two for the remaining tasks. The loss function is defined as

\[ L = L_{cls} + L_{box} + L_{mask} + L_{kpt}, \]

where the classification loss \( L_{cls} \) is the log loss over two classes (hand vs. background). The bounding box regression loss \( L_{box} \) is identical to Eq. 5.2. The mask loss \( L_{mask} \) is the binary cross-entropy loss over predicted hand mask and ground truth, and the key point mask loss \( L_{kpt} \) is the average cross-entropy loss over the predicted \( N \) joints and \( N \) ground truth points.

At test time, the Mask R-CNN keypoint head branch outputs the confidence maps of all joints. Fig. 5.6(a) shows an example of the confidence maps of left thumb finger joints and right index finger joints. Because relationships among hand joints
5.5. CONFIDENCE REFINEMENT

Confidence maps provide the probabilities of each joint position, which can be viewed as \( p(I|P) \) in Eq. 5.5. According to the PS model, the prior \( p(X_i) \) is supposed to encode probabilistic constraints on the part relationships and capture the unified global structure of objects in the training data. We present a conceptually simple method to approximate the tree prior \( p(X_i) \) and two methods to implement it.

5.5.1. Tree prior approximation

As illustrated in Fig. 5.1(d), masks predicted by the Mask R-CNN mask head capture global structures of hand instances, but they lack information on part relationships. 

are only implicitly learned during the training process, localizing key point positions via finding locations with maximum probabilities could lead to large pixel error. As shown in Fig. 5.6(b), the two joints of the left thumb finger are estimated incorrectly on the left index finger. Similarly, the joints of the right index finger are incorrectly predicted as the ring and little finger. Moreover, if we cannot guarantee the correctness of the confidence maps, they cannot be used alone to infer the presence or visibility of joints. Inspired by PS models by which the poses of objects can be estimated by combing global structure constraints (which encode part relationships) and part confidence maps, we utilize the output of the Mask R-CNN mask head to explicitly learn the kinematic structures of the hands. Learned kinematic structures are used to refine confidence maps of corresponding hands and infer the presence of joints.

Figure 5.6: (a) Confidence maps of left thumb finger joints and right index finger joints. (b) Mask R-CNN detection result.
Our idea is to find a training subset \( S_i \) that has a similar mask as the \( i \)th test hand mask; then, the kinematic tree prior that encodes the part relationships of the test hand can be learned from \( S_i \).

Before we introduce our method for determining \( S_i \), there is one critical question: Can we make masks comparable when they may have a different scale and size? In Mask R-CNN, the mask head branch would first predict a fixed size mask for each instance, and the predicted mask is further resized to have the true size of the corresponding instance. We reshape the fixed size mask into a feature vector so that every hand instance can be represented in a comparable form. This feature representation projects the instances to the feature space so that visually similar instances are close to each other. Feature vectors of all hand instances in training data are extracted by the same procedure and stored on disk for future use.

**Unsupervised learned tree prior approximation**

Given that the \( i \)th hand instance can be represented by a feature vector \( f_i \), we use a K nearest neighbors (KNN) search to find the features of the training images that lie close to \( f_i \) in the feature space, and \( S_i \) is composed of the corresponding training images. To learn \( p(X_i) \) from \( S_i \), for simplicity, we assume that all hand parts are independent, and the prior \( p(X_i) \) is approximated as

\[
p(X_i) \approx p(x_1^i, x_2^i, ..., x_N^i | f_i) = \prod_{j=1}^{N} p(x_j^i | f_i)
\]

where \( p(x_j^i | f_i) \) is the \( j \)th part prior of the \( i \)th hand instance based on the feature vector \( f_i \). Let \( \text{coord} = (x, y) \) denote the coordinate of a pixel in image, \( p(x_j^i | f_i) \) is computed as

\[
p(x_j^i | f_i) = \begin{cases} 1 & \|\text{coord} - \text{mean}_{S_i}^j\|_p \leq d \\ 0.5 & \text{otherwise} \end{cases}
\]

where \( \|\cdot\|_p \) is the Minkowski distance between two points and \( \text{mean}_{S_i}^j \) is the mean coordinate of the \( j \)th part in \( S_i \). \( d \) is a hyperparameter that adjusts the influence of \( p(X_i) \). We adopt this formulation because it allows faster computation than other common probabilistic distributions and it is mainly defined to refine joint confidence maps. However, in our formulation, we assume that all joints are independent,
5.5. CONFIDENCE REFINEMENT

Figure 5.7: KNN for hand instance kinematic prior approximation. A hand instance (a) is expressed by a feature vector $f_i$, training data with similar features are found out by KNN search (b). The kinematic structure of the hand instance are learned from those training data (c).

and joint relationships are implicitly preserved by the subset of training data in $S_i$. Fig. 5.7 shows an example of this process. The absence of joint(s) is inferred by the absent joints in $S_i$, e.g., if the number of absent tips of the ring finger from the $S_i$ result is greater than a threshold $\tau$, the ring finger tip is deemed as invisible for the $i$th hand instance.

Because the whole process needs to be repeated for every hand instance, the KNN-based tree prior approximation method is computationally heavy. Moreover, the features of the training data need to be stored, which may require a large amount of space. These limitations motivate us to find $S_i$ via other methods that require less computation and storage.

**Supervised learned tree prior approximation**

It is possible to use a supervised learning method to determine $S_i$, which should be faster than KNN, provided that a labeling method can be found that is able to distinguish different hand poses. In our framework, a hand instance is assigned a label $L = \{j_1, j_2, ..., j_N\}$, where the index of $j_i$ in the label vector indicates the joint name and $N$ is the number of joints. We first compute the distances between each hand joint and the origin. Those computed distances are stored in a vector $v$; then we sort $v$ in decent order. The value of $j_i$ is determined by the index of the corresponding joint plus one in sorted $v$. For example, if a sorted $v$ is of the form $v = \{\text{dist}(\text{joint}_2, \text{org}), ..., \text{dist}(\text{joint}_1, \text{org})\}$, where $\text{dist}$ is the function that computes the distance between two points, $\text{joint}_i$ is the coordinate of a joint and $\text{org}$ is the coordinate of the origin. The values of $j_1$ and $j_2$ are $N$ and 1 in the label
vector $L$. If a joint is not visible, the corresponding entry in $L$ is set to 0. In most cases, the presented labeling method is able to distinguish different hand poses and preserve joint spatial relationships, especially when we need to localize all the joints and tips of a hand.

The next step is to choose a proper classifier. We select random forest [10] (RF) because it is naturally designed for multiclass classification and it provides soft decision boundaries. Moreover, RF is able to handle high-dimensional input data efficiently, which allows fast computation at test time. Fig. 5.8 shows an example of how we use RF to predict the kinematic tree prior of a test hand. Feature vector $f_2$ of the test hand goes through all trees and falls into some leaf nodes (Fig. 5.8a). $f_2$ is assigned a label $l$ by RF, and we select training data with the same label $l$ (Fig. 5.8b), which is actually the training subset $S_l$. The kinematic tree prior (Fig. 5.8c) is estimated by Eq. 5.6. In practice, kinematic tree priors learned from each leaf node can be computed offline. It is only necessary to store joint coordinates and bounding box width and height, i.e., totally $N \times 2 + 2$ numbers, which requires much less storage space compared to our KNN method. The absence of joints or tips can be directly predicted by RF (entry in the label vector is 0).

5.5.2. Final localization

Given $p(I|P)$ and $p(X_i)$, the posterior probability $p(P|I)$ can be computed by Eq. 5.5. Joint locations are estimated by finding the image positions with the highest proba-
bilities. Note that both our tree prior approximation methods are able to detect the presence of joints; if Mask R-CNN fails to detect the $j_{th}$ joint of the $i_{th}$ hand, the position of the $j_{th}$ joint is estimated by $\text{mean}_i^j$.

5.6. Data Preparation

We generated two 2-hands datasets, dexter2Hands and NYU2Hands, based on the depth images of the popular single-hand datasets dexter1 [90] and NYU hand pose [96], respectively. For the dexter2Hands dataset, we randomly selected 2504 images from 3154 images in the dexter1 dataset as a training set, and the remaining 600 images were equally split into a validation set and a test set. Because the images of dexter1 only contained hands and the image size was relatively small ($320 \times 240$), images in the final training data of dexter2Hands are of size $640 \times 240$ and are generated by the concatenation of randomly selected left- and right-hand images from a (mirrored-)training set. The same processes were applied to generate the validation data as well as test data for the dexter2Hands dataset. In our experiments, the dexter2Hands training data contained 57404 images, validation data contained 14025 images and test data contained 9925 images. The keypoint number of a hand instance was 5, including thumb finger tip, index finger tip, middle finger tip, ring finger tip and little finger tip. Fig. 5.9(a) shows an example of the images in dexter2Hands. Hand masks of the dexter2Hands dataset were generated by setting pixel values of the hand object in each image to 1 and background to 0.

The processes used to generate the NYU2Hands dataset were similar, and we only used depth images from viewpoint 1. However, the image size of NYU2Hands was the same as NYU, which was $640 \times 480$. The training data and validation data of NYU2Hands were generated by copying the mirrored left side hand (in image coordinate) to be the corresponding right side hand. The keypoint number of a hand instance was 19, which are little finger tip (LT), little finger joint 1 (L1), little finger joint 2 (L2), little finger joint 3 (L3), ring finger tip (RT), ring finger joint 1 (R1), ring finger joint 2 (R2), ring finger joint 3 (R3), middle finger tip (MT), middle finger joint 1 (M1), middle finger joint 2 (M2), middle finger joint 3 (M3), index finger tip (IT), index finger joint 1 (I1), index finger joint 2 (I2), index finger joint 3 (I3), thumb finger tip (TT), thumb finger joint 1 (T1) and thumb finger joint 2 (T2), respectively.
5.7. IMPLEMENTATION DETAILS

5.7.1. Mask R-CNN

Training:

In our experiments, parameters of Mask R-CNN backbone are initialized by the ImageNet [23] pre-trained weights. Training depth images are converted into 3-channel images by replication. We train the model on 50K iterations for dexter2Hands and 60K iterations for NYU2Hands, starting from a learning rate of 0.002 and reducing it by 10 at 15K and 35K iterations. Models are trained on 4 Nvidia GTX 1080 GPUs. Each batch has 1 image per GPU and each image has 128 sample ROIs. Our code is implemented in TensorFlow/Keras based on the repository provided by [1]. Other implementations are identical as [40].
Inference:

At test time, the bounding box branch directly predicts bounding boxes of hand instances. The instance segmentation branch predicts a mask of size $28 \times 28$ and the key point mask branch outputs a $56 \times 56 \times N$ joint mask for each hand instance. $N$ is 5 for dexter2Hands and 19 for NYU2Hands. Those masks are further resized to the size of the bounding box, and binarized at a threshold $\tau$ to obtain the final detection result. $\tau$ is 0.1 for instance masks and 0.5 for key point masks. Instance threshold is chosen at a low value because we hope the estimated mask could cover hand finger tips. The feature vector of a hand instance is generated by reshaping the $28 \times 28$ mask into a vector of $1 \times 784$.

5.7.2. Tree prior approximation

For KNN search, we set $K = 10$ and threshold $\tau = 4$ for both datasets. For our RF approach, we use the RF implementation provided by [70] to construct a 10-tree RF and do not change other parameters. Each tree has a depth of around 30 and around 6000 leaf nodes. We choose Manhattan distance to compute part prior $p(x_j | f_i)$ in Eq. 5.7 since it is relatively fast and $d$ is set to 30 for Dexter2Hands dataset and 40 for NYU2Hands dataset.

5.8. Experiments

We evaluate our methods on the test data of Dexter2Hands and NYU2Hands. The results of our methods are compared with two versions of Mask R-CNN, i.e., keypoint only and keypoint & mask, as well as the ground truth joint positions. Mask R-CNN keypoint only indicates that the joint positions are localized via the finding positions of joint confidence maps with maximum probabilities. The Mask R-CNN keypoint & mask restricts the keypoints lying on the estimated masks. We employ two metrics to evaluate the performance of our proposed method. The first metric is the average Euclidean distance in pixels between the results and the ground truth. The second metric is the percentage of success frames in which all joint errors are below a certain threshold. In addition, we compute the false positive (FP) rate and the false negative (FN) rate to infer the presence of each joint to validate the adequacy of our methods.
In our experiments, we found that Mask R-CNN is able to correctly detect almost all hand instances, with fewer than 5 frames being incorrectly detected.

5.8.1. Results and discussion

Fig. 5.10 shows the comparison results of our methods and Mask R-CNN on the Dexter2Hands dataset. In all cases, we can see that our methods produce fewer pixel errors of each tip on each individual hand and both hands. Because the image background of this dataset is relatively clean, estimating the joint locations via the finding positions with maximum probabilities without constraint is noise sensitive. This is the reason for the large pixel errors in the Mask R-CNN keypoint-only method. As shown in Table 5.1, the average joint pixel error over all frames of our KNN method is 4.8, which is better than our RF method (5.7) and the Mask R-CNN keypoint & mask method (6.2). The fraction of good frames over a different threshold for each individual hand and both hands is shown in Fig. 5.11. For the left hand, our KNN method achieves the best good frame rate (82%) when the threshold is 10 pixels, while the good frame rate is 78% for our RF method and 77% for mask R-CNN keypoint & mask. Similarly, the performance of our KNN and RF methods outperform other methods on the right hand (Fig. 5.11b) and both hands (Fig. 5.11c).

Table 5.1: Quantitative evaluation on Dexter2Hands.

<table>
<thead>
<tr>
<th>Method</th>
<th>position error (pixels)</th>
<th>FN (%)</th>
<th>FP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our (KNN)</td>
<td>4.8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Our (RF)</td>
<td>5.7</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Mask RCNN (kpt &amp; mask)</td>
<td>6.2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Mask RCNN (kpt)</td>
<td>38.5</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Another advantage of our methods is that they are able to infer the presence of joint visibilities. Fig. 5.12 shows a typical example. Given an input image with the ground truth that only the middle fingers of both hands are visible (Fig. 5.12a), Mask R-CNN wrongly predicts that the pinky, ring, middle and index finger tips are visible on the left hand. Similarly, all fingertips are estimated to be overlapping on the right hand (Fig. 5.12b). Our methods successfully detect the presence of joints and correctly predict the visible joint positions. Both versions of Mask R-CNN produce FN rates of 2% and FP rates of 3%, while FN rate of our KNN and RF methods are 1% and 0%. The FP rate of our method is 1%.

We also compare our methods with Mask R-CNN on the NYU2Hands dataset,
5.8. EXPERIMENTS

Figure 5.10: Per-joint mean error distance in pixels on dexter2Hands.
5.8. EXPERIMENTS

(a) Left hand.

(b) Right hand.

(c) Both hands.

Figure 5.11: Fraction of frames within distance on dexter2Hands.
Figure 5.12: Examples of our methods compared to Mask-RCNN on Dexter2Hands dataset. (a) Groundtruth. (b) Outputs of Mask-RCNN (with mask). (c) Outputs of our KNN method. (d) Outputs of our RF method.

which is more challenging since there are 19 joints on each hand. As shown in Fig. 5.13, our methods achieve fewer pixel errors than Mask R-CNN in all cases. The mean pixel errors of the left middle finger tip (MT in Fig. 5.13) of Mask R-CNN are 27.3 (keypoint only) and 22.4 (keypoint & mask), while the mean pixel errors of our methods for that joint are 11.2 (KNN) and 12.2 (RF). For the right hand, though on some joints (e.g., Fig. 5.13b: L1, RT, MT, etc.) Mask R-CNN keypoint & mask has fewer pixel errors than our RF method, the largest margin is on the right middle finger tip, which is 1.4 (11.1 vs 12.5). Table 5.2 shows the average position errors in pixels for the different methods. The mean joint pixel errors over all frames of our methods are 9.3 (KNN) and 10.1 (RF), which is better than Mask R-CNN keypoint & mask (12.4) and keypoint only (16.2). The proportion of good frames over different error thresholds is shown in Fig. 5.14, and we can see a clear order of performance of the four methods: our KNN method is better than our RF method, and the proposed methods outperform Mask R-CNN. The FN and FP rates of our methods are all 0%, while the FN rates of both versions of Mask R-CNN are 2%, and the FP rates are 0%. Some qualitative results for the NYU2Hands dataset are shown in Fig. 5.15. As seen, our proposed methods can better preserve the hand joint relationships and provide a more accurate estimation.

<table>
<thead>
<tr>
<th>Method</th>
<th>position error (pixels)</th>
<th>FN (%)</th>
<th>FP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our (KNN)</td>
<td>9.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Our (RF)</td>
<td>10.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mask R-CNN (kpt &amp; mask)</td>
<td>12.4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Mask R-CNN (kpt)</td>
<td>16.2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.2: Quantitative evaluation on NYU2Hands.

Runtime

The runtime of both versions of Mask R-CNN to process a test image of Dexter2Hands dataset is 0.45s on average. It takes 0.5s for our KNN method and 0.46s for our RF method. For the test image of NYU2Hands dataset, the averaged process time of
5.8. EXPERIMENTS

(a) Left hand.

(b) Right hand.

(c) Both hands.

Figure 5.13: Per-joint mean error distance in pixels on NYU2Hands.
5.8. EXPERIMENTS

(a) Left hand.

(b) Right hand.

(c) Both hands.

Figure 5.14: Fraction of frames within distance on NYU2Hands.
both versions of Mask R-CNN is 0.5s because the image size is two times larger than test images of Dexter2Hand and needs to locate more joints. The process time of our KNN method for the NYU2Hands dataset is around 0.85s per image, including 0.25s for the calculation of mean joint positions for each joint in KNN search result. Compared to our KNN method, our RF method is much faster because mean joint positions are already stored after training, which requires 0.55s to process a NYU2Hands test image.

5.9. Summary

We present a new algorithm based on the PS model for estimating 2D multi-hand poses from single depth images. The proposed framework utilizes Mask R-CNN to learn the mapping from the local information of joints and the global structures of hands to their corresponding poses. We formulate a new utilization of the segmentation output of Mask R-CNN and propose two ways to approximate pose priors of test instances. The estimated pose priors can be used to infer the presences of joints. Our method addresses the issues of interchangeable estimations by solely using Mask R-CNN for the detection of hand keypoints. We also present interplays between Mask
R-CNN and the PS model, as well as Mask R-CNN and random forests. The performance of our algorithm has been validated on two self-generated datasets with two hands, which can also serve as a baseline for future research.
6.1. Conclusion

In this thesis, we investigated the problem of estimating 2D object poses. We presented a general method based on a pictorial structure [29] (PS) model for insect and multihand pose estimation. Our algorithms jointly learn mappings from the local appearances and the global anatomical structures of objects to their corresponding poses. The proposed frameworks are able to localize a varying number of targets in a complex background.

Chapter 3 presented the first algorithm for estimating the insect pose from single-images. The proposed framework utilizes the discriminative power of the random forests (RF) model for landmark proposal generation. The global structure of an insect is represented by hand-crafted features, and we proposed a data-driven unsupervised approach to approximate the geometric prior for modeling various insect poses. The insect pose is determined by combining both global and local information, and our method achieves high precision of landmark localization when no human intervention is involved. This method requires relatively fewer training samples and allows fast computation.

In Chapter 4, the first convolutional neural network (CNN) based framework for estimating bee poses from single-images was proposed. Due to the powerful representation capability of CNN, our approach is demonstrably robust and accurate and is effective even for images that might be challenging for human annotators. Compared to our RF-based method, this framework is less noise-sensitive and exhibits
better generalization performance in unconstrained conditions.

Chapter 5 aimed at solving the task of estimating 2D multihand poses from single depth images. By utilizing Mask R-CNN, our previous bee pose estimation framework can be extended to estimate the poses of multiple and more complex objects and allows for simultaneous detection of multihand instances and localization of hand joints. In addition, we presented a supervised method for implementing the global structure approximation, which is more efficient.

6.2. Discussion

While our RF-based framework proposed in Chapter 3 achieved high accuracy for insect pose estimation when the images contained no stick, the RF classifier was unable to reliably distinguish the stick from the bee body parts, and hand-crafted geometric features were not able to fully represent the global structure of the bee. These drawbacks were addressed in Chapter 4 with the stronger discriminative power of CNN. However, training CNNs need a large number of training data. Due to the lack of training data, the performance of our CNN-based algorithm was not evaluated on other types of insects.

In our bee pose estimation frameworks, we assume that there is only one instance in the image and that the position of the insect is fixed. These constraints were removed by using the region proposal network (RPN) for instance detection in Chapter 5. However, our proposed methods still have the following limitations:

- Additional storage space. In our unsupervised tree prior approximation method, feature vectors of all training samples need to be stored, which requires a large amount of storage space. Though in our supervised approach the storage cost is significantly reduced, it will still be a burden for massive datasets.

- Limitation of generalization capability. Since our tree prior is computed from the training data, the performance of our method can be poor when the type of a specific test instance is never seen in the training data. For the same reason, our method is not scale and rotation invariant.

- Computationally slow. In our framework presented in Chapter 5, there are
two time-consuming parts, one is the RPN network, the other is the Mask generation head branch. Depending on the input image size, it takes 0.46-0.85 seconds for our multihand pose estimation algorithm to process one image, which is too slow for real-time applications.

6.3. Future work

Since the presented tree prior approximation method is the bottleneck of our algorithm, it is necessary to find other substitutions that are more efficient and effective. One direction is to utilize CNN to learn the global tree prior. The authors in [112] presented a network to learn the 3D pose prior based on incomplete and noisy confidence maps. Their idea could also be applied to our tasks. Taking the multihand pose estimation task as an example, given an input image (Fig. 6.1a), we can use a CNN (Fig. 6.1b) to generate instance bounding boxes and possible keypoint locations (Fig. 6.1c). The CNN could be a Mask R-CNN or other faster architecture. The learned keypoint proposals serve as the input to another CNN (Fig. 6.1d) for global structure estimation. Note that the keypoint proposals are often independently learned because this formulation leads to a better result [40]. The key idea of the PriorNet (Fig. 6.1d) is to jointly estimate the global structure of a hand instance based on keypoint confidence maps. If the global structure of an instance can be estimated in this way, our algorithm does not need to use the training data during the test stage and is rotation and scale invariant.

R-CNN-based detectors are all two-stage approaches, i.e., in the first stage, a sparse set of object proposal are generated, and in the second stage, a classifier is applied to those proposals. While R-CNN-based detectors achieve high object detec-
tion accuracy, it is computationally slow because the classifier needs to be applied to all object proposals. In contrast, one-stage detectors are directly applied on an image and output object detection result. Recently, a one-stage detector called RetinaNet [59] was proposed; it achieves similar performance as Faster R-CNN [77], and it is two times faster. Though RetinaNet is not able to predict keypoint locations, we believe it is possible to add another branch for keypoint proposal generation to RetinaNet with a similar idea to how Faster R-CNN was extended to Mask R-CNN.

Apart from the aforementioned solutions to the drawbacks of our proposed method, there is still plenty of room to improve the performance of our method. For example, temporal information could be involved in our insect pose estimation framework to improve the performance. Our single-frame algorithm first outputs keypoint proposals, and then, the relationships between neighboring frames can be used to remove noise. While our multihand datasets are created by mirroring existing single-hand datasets, the natural next step is to generate a real multihand dataset with accurate labeling that not only labels the joint position but also provides visibility information of occluded joints. Moreover, it is necessary to extend our 2D multihand pose estimation into 3D space for real-world applications. Based on our idea presented in Fig. 6.1, we can either directly use a 3D-CNN, or first use a 2D-CNN to produce 2D joint confidence maps and combine the learned 3D hand prior for final 3D hand pose estimation.
Author’s Contributions

This thesis presented a series of frameworks based on the pictorial structure model for estimating poses of insects and multiple hands from single images. Highlights of main chapters are as follows:

- In Chapter 3, the first framework for detailed insect pose estimation from single images was presented. The author of this thesis was involved in developing the contribution of this work, which includes generating new ideas about random forests for image segmentation, implementing code for experiments and writing text for part of the paper. The final results were published in


- Chapter 4 presented a new convolutional neural network (CNN)-based framework for bee pose estimation. The author of this thesis developed the main idea of this work, implemented code for experiments, and took the lead in writing the following papers:


  DUAN, L., SHEN, M., Quek, B., CUI, S. AND DEUSSEN, O. A Pictorial
Structure Model-based Framework For Bee Pose Estimation. In Pattern Recognition Letters. (Submitted.)

• In Chapter 5, a novel framework based on a PS model to estimate 2D multihand poses from depth images is presented for the first time. The author of this thesis took the lead in the project, including algorithm design, code implementation, experimental design and writing the text for the paper. All findings were published in


Other Publications

The author of this thesis also contributed in the following publications. These papers are not part of the thesis.

• GUO, Z., KEE, C., DUAN, L., DEUSSEN, O. AND ZHOU, Y. Real-time detection of structural landmarks in corneal topography with single shot detector. In Biomedical Signal Processing and Control. (To be submitted.)


A.1. Shape Descriptors

In this section, a brief introduction of the five shape descriptors used in Chap. 3 is given.

**Edge histogram descriptor**

Insect body parts' shapes reflect good linear edges in different orientations, thus the edge histogram descriptor (EHD) [31] is chosen as the shape descriptor. The EHD is a widely used method for shape detection and it represents the occurrence frequency of five categories of edges, i.e., vertical, horizontal, $45^\circ$ diagonal, $135^\circ$ diagonal and isotropic (no edge) in each sub-image [31]. The sub-image is defined as dividing the image space into $2 \times 2$ non-overlapping regions. Given an image of a bee, an edge detector is first used to detect the contour of the bee. Then each of the edge filters shown in Fig. A.1 is applied to each sub-image to check if there exists a corresponding type of edge, otherwise the sub-image is considered as containing no edge. The EHD of an image is computed by summing up occurrences of the five types of edges through all sub-images and the dimension of the EHD is five.

**Geometrical feature descriptor**

The geometrical feature descriptor (GFD) [56] is a 6D feature vector which stores the area, the perimeter, major axis length, minor axis length, eccentricity (length of
A.1. SHAPE DESCRIPTORS

<table>
<thead>
<tr>
<th>1 0</th>
<th>0 1</th>
<th>1 1</th>
<th>0 0</th>
<th>0 1</th>
<th>1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>0 1</td>
<td>0 0</td>
<td>1 1</td>
<td>1 0</td>
<td>0 1</td>
</tr>
</tbody>
</table>

(a) (b) (c) (d)

Figure A.1: (a) Vertical edge filter. (b) Horizontal edge filter. (c) $45^\circ$ diagonal edge filter. (d) $135^\circ$ diagonal edge filter.

major axis/length of minor axis) and circularity ($\text{perimeter}^2/\text{area}$) of a bee. This shape descriptor characterizes geometrical features which are important for detecting missing body parts of a bee.

Shape signature histogram descriptor

A shape signature (SS) [108] represents shape by a 1-D function derived from shape boundary points. In our experiments, the SS of a bee is computed simply by transforming the boundary points from the Cartesian coordinate system to the polar coordinate system. Note there are other popular SSs such as centroidal profile, complex coordinates and centroid distance. Since SSs that directly using boundary point information may be noise-sensitive, the shape signature histogram (SSH) of a bee is obtained by quantizing the angle in the polar coordinate system into $M$ bins. Following [86], $M$ is set to 45 in our experiments.

Fourier descriptor

Unlike the SSHD represents the shape of an object in the spatial domain, the Fourier descriptor (FD) [109] characterizes the global object structure in the spectral domain. The FD is normally obtained through Fourier transform on the centroid distance shape signature. Let $(x(t), y(t))$, $t = 0, 1, ..., N - 1$, denotes the boundary points of a shape, where $N$ is the number of boundary points. The centroid distance function of the shape is computed as

$$r(t) = \left(\left|x(t) - x_c\right|^2 + \left|y(t) - y_c\right|^2\right)^{1/2}$$  \hspace{1cm} (A.1)

where $(x_c, y_c)$ denotes the centroid of the shape, $x_c = \frac{1}{N} \sum_{t=0}^{N-1} x(t)$, and $y_c =$
\[
\frac{1}{N} \sum_{t=0}^{N-1} y(t).
\]

The discrete Fourier transform of \( r(t) \) is computed as

\[
a_n = \frac{1}{N} \sum_{t=0}^{N-1} r(t) \exp \left( -\frac{j2\pi nt}{N} \right), \quad n = 0, 1, ..., N - 1. \tag{A.2}
\]

where \( a_n \) is the Fourier transformed coefficients of \( r(t) \).

The normalized Fourier coefficients \( b_n \), which is rotation, scaling, translation and starting point invariant, is obtained by

\[
b_n = \frac{a_n}{a_0}. \tag{A.3}
\]

The FD of a shape is defined as \( FD = \{|b_n|, 0 < n \leq N\} \), where \(|b_n|\) is the magnitude of \( b_n \). Since \( r(t) \) are real values, thus only half of the \( b_n \) are distinct. In our experiments, \( N \) is set to 50 and the dimension of the FD is 25.

**Hu moment descriptor**

Image moments are certain particular weighted average of image pixels’ intensities [104]. Let \( I(x, y) \) denotes an image, image moments \( M_{i,j} \) are calculated by

\[
M_{i,j} = \sum_x \sum_y x^i y^j I(x, y). \tag{A.4}
\]

Similarly, the central moment of a gray-scale or binary image is defined as

\[
\mu_{i,j} = \sum_x \sum_y (x - \bar{x})^i (y - \bar{y})^j I(x, y). \tag{A.5}
\]

The Hu Moment descriptor (HuMD) [43], which is consisted of seven rotation, scaling and translation invariant moments, is computed as
A.1. SHAPE DESCRIPTORS

\[ HuMD[1] = \eta_{20} + \eta_{02} \]
\[ HuMD[2] = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2 \]
\[ HuMD[3] = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \]
\[ HuMD[4] = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \]
\[ HuMD[5] = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \]
\[ HuMD[6] = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \]
\[ HuMD[7] = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \]

where \( \eta_{ij} \) is obtained by
\[
\eta_{ij} = \frac{\mu_{ij}}{\mu_{001} + \frac{i+j}{2}}.
\]
Bibliography


91


