Robust Real-Time Hands-and-Face Detection for Human Robot Interaction

by

SeyedMehdi (Sepehr) MohaimenianPour

B.Sc., Computer Vision, Université de Bourgogne, 2015
B.Sc., Computer Science, Amirkabir University of Technology, 2014

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2018

Approval

Name:	**SeyedMehdi (Sepehr) MohaimenianPour**
Degree:	**Master of Science (Computing Science)**
Title:	**Robust Real-Time Hands-and-Face Detection for Human Robot Interaction**
Examining Committee:	**Chair:** Dr. Angelica Lim Assistant Professor
	Dr. Richard Vaughan Senior Supervisor Associate Professor
	Dr. Greg Mori Supervisor Professor
	Dr. Yasutaka Furukawa External Examiner Assistant Professor
Date Defended:	**April 27, 2018**

Abstract

With recent advances, robots have become more affordable and intelligent, which expands their application domain and number of consumers. Having robots around us in our daily lives creates a demand for an interaction system for communicating humans' intentions and commands to robots. We are interested in interactions that are easy, intuitive, and do not require the human to use any additional equipment. We present a robust real-time system for visual detection of hands and faces in RGB and gray-scale images based on a Deep Convolutional Neural Network. This system is designed to meet the requirements of a hands-free interface to UAVs described below that could be used for communicating to other robots equipped with a monocular camera using only hands and face gestures without any extra instruments.

This work is accompanied by a novel hands-and-faces detection dataset gathered and labelled from a wide variety of sources including our own Human-UAV interaction videos, and several third-party datasets. By training our model on all these data, we obtain qualitatively good detection results in terms of both accuracy and speed on a commodity GPU. The same detector gives state-of-the-art accuracy and speed in a hand-detection benchmark and competitive results in a face detection benchmark. To demonstrate its effectiveness for Human-Robot Interaction we describe its use as the input to a novel, simple but practical gestural Human-UAV interface for static gesture detection based on hand position relative to the face. A small vocabulary of hand gestures is used to demonstrate our end-to-end pipeline for un-instrumented human-UAV interaction useful for entertainment or industrial applications. All software, training and test data produced for this thesis is released as an Open Source contribution.

Keywords: Human Robot Interaction, Deep Learning, Computer Vision, Object Detection, Gesture Recognition

Dedication

Dedicated,
to my inspiring parents,
and sister,

for being there for me,
for having faith in me,
and for all their support...

Acknowledgements

I owe my most profound gratitude to my senior supervisor, Professor Richard Vaughan. Without his continuous support, enthusiasm, mentorship, and beyond everything his patience, this work would hardly have been completed. I admire his immense knowledge and domain expertise for crafting new ideas; I always enjoyed the meetings we had, and appreciate his help in writing manuscripts. I also express my warmest gratitude to my supervisor Professor Greg Mori, for his guidance, encouragement, and support towards using the leading AI methods in this work.

I am deeply grateful to Dr. Mani Monajjemi and Dr. Shokoofeh Pourmehr, my best friends who helped me a lot on this journey. Mani, it was an honour for me to know you, work with you, learn from you, and follow your path in the past nine years.

I owe a significant debt to my parents for their unconditional support in any decision I have made in my life, and my sister for always being there for me. I would also like to offer my special thanks to Sara Daneshvar, for since she entered my life, she has made every single second of this journey more pleasant.

Lastly, It is a pleasure to thank my friends who shared the graduate student life with me. I am indebted to my many of colleagues at SFU Autonomy Lab to contributed to my research through their criticism, support and encouragement: Mani, Shokoofeh, Abbas, Jake, Jacob, Jack, Lingkang, Geoff, Pratik, Rakesh, Faraz, Bita, Amir, and Payam.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With recent advances in technology, autonomous robots are becoming more affordable and advance. Different robots are finding their ways in more and more areas of our everyday lives. Robots have been used in industry for a while now, Amazon uses Kiva robots that interact with employees in their warehouses, delivery robots are appearing in London and Washington, people use autonomous UAVs that can follow them and record their activities, self-driving cars are appearing in the streets, home help and health care companions for elderly and disabled people are being used in Japan. Robots have many more use cases, to name a few: museum tour and other guidance robots, agricultural robots, search and rescue robots, telepresence robots, socially interactive robots and many more.

A crucial component for social robots is the human factor: detecting and interacting with humans who require a robot's attention, or who want to modify a robot's behaviour [11]. With the rapid increase of robots presence in our community, having a universal and standard language to interact with them will be essential, not only for trained experts but also for ordinary people. The first step before designing a standard vocabulary for interacting with robots is having the means for transferring the information to the robot. This thesis investigates a simple, intuitive, and un-instrumented method for giving instructions to robots.

Humans frequently interact with each other using face engagement and hand gestures [70]. Karam [50] found hand gestures to be the most common communication gestures among humans and hence suggested them as the most natural gestures for Human-Computer Interaction. Therefore we consider hand gestures as an intuitive means for an un-instrumented user to command a robot. Several researchers have demonstrated Human-Robot interfaces that use either face detection, hand detection or both [85]. We present a fast and accurate detector that finds the hands and faces of multiple people in 2D images at frame-rate, which can be used directly as an input to an HRI system.

In this work, we designed a fast and robust hand and face detector that can detect and locate all the hands and faces of multiple people in a single monocular camera image with state-of-the-art accuracy and speed. Our system uses a scalable CNN model that could be

resized for a speed/accuracy trade-off based on YOLOv2 [87]. We provide the first integrated hands and faces detection dataset for the Human-Robot Interaction (HRI) community. We also designed a novel, simple, but effective method for static gesture detection based on hand position relative to the face, and a small vocabulary of hand gestures for interaction between an un-instrumented human and a situated Unmanned Aerial Vehicle (UAV). Using this vocabulary, we implemented a real-world, end-to-end, close-range interaction system by which an un-instrumented user can take-off, manoeuvre and land a UAV by gestures alone with a single on-board camera.

We evaluate our implementation with well-known hand- and face-detection benchmarks, and by real-world experiments where we determine the hands/faces detection and gesture recognition accuracy versus human-robot distance in various environments under different illuminations and backgrounds.

Chapter 2

Literature Review

In this work, we present an end-to-end system for un-instrumented Human-Robot Interaction, consisting of several parts. The primary focus of our work is an accurate detection of the interaction partner and their hands and faces to translate them into high-level commands which then will be executed by the robot or provide an appropriate response. In this chapter, we survey the works on major components of our system. We need to review and select the best possible solution detector for detecting hands and faces in the monocular camera images and then use these detections for HRI purposes. The closest works to our simultaneous hands and faces detector are multi-modal systems for very close range Human-Computer Interaction [10, 33, 6, 5].

2.1 State of The Art in Object Detection

Detecting objects in an image is a very challenging area even for *deep learning*. The problem is not just about finding "what" exists in the image, it is also about "where" the object is. This brings up many issues, for instance, the object recognition part needs to be solved invariant to image transformations whereas the localization of the recognized object in the image needs to recover those ignored transformations.

Furthermore in the literature, things get more complicated. It is not just about selecting rigid bounding boxes around objects of interest in the images anymore; it becomes about recovering objects spacial positions, which is their 3D position and orientation in the world based on a fixed frame. Most of the current state-of-the-art object detection systems are base on finding the enclosing bounding boxes in the image frame.

Researchers have examined many approaches for this crucial and challenging topic in computer vision, most on the same principle. They involve extracting features from the image, followed by finding the features that match the object of interest. The part of the image corresponding to those features is then selected as the location of the object.

We can divide these approaches into two classes: (i) the now-classical algorithms which do not use Convolutional Neural Networks (CNN) as the feature extractor, (ii) algorithms which take advantage of CNNs.

2.1.1 Classical algorithms

In this section, we briefly describe a few major classical algorithms that are widely used and had the most impact on the literature.

Viola-Jones

The earliest widely used classical method is *Viola-Jones* [110]. The Viola-Jones object detection framework (proposed in 2001) is the first object detection framework to provide competitive object detection rates in *real-time*. Although this method was adapted and trained to detect a variety of object classes later on by other researchers, it was primarily motivated by the face detection problem. They use *Haar-like* features to extract those similar properties common to all human faces. Then an image representation called the *integral image* evaluates rectangular features in constant time, which makes it fast enough to be used in real-time.

The object detection framework employs a variant of the learning algorithm AdaBoost to both select the best features and to train classifiers that use them. This algorithm constructs a "strong" classifier as a linear combination of weighted simple "weak" classifiers. The authors showed that a cascade of gradually more complex classifiers achieves better detection rates, compared to a single very complex classifier. The evaluation of the strong classifiers generated by the learning process can be done quickly, but it is not fast enough to run in real-time. For this reason, the strong classifiers are arranged in a cascade in order of complexity, where each successive classifier is trained only on those selected samples which pass through the preceding classifiers. If at any stage in the cascade, a classifier rejects the sub-window under inspection, no further processing is performed on the sub-window, and the system continues to search the next sub-window. The cascade, therefore, has the form of a degenerate tree. In the case of faces, the first classifier in the cascade - called the attentional operator - uses only two features to achieve a false negative rate of approximately 0% and a false positive rate of 40%. The effect of this single classifier is to reduce by roughly half the number of times the entire cascade is evaluated. In cascading, each stage consists of a strong classifier. So all the features are grouped into several stages where each stage has a certain number of features. The job of each stage is to determine whether a given sub-window is definitely not a face or could be a face. A given sub-window is immediately discarded as not a face if it fails in any of the stages.

Discriminatively Trained Part Based Models

Discriminatively Trained Part based Models (DPM) [28] is another well-known method which is an algorithm between generative and discriminative models. First, given an image, build a pyramid for any scale. Then, use different root filters and part filters to get responses. By combining these responses in a star-like model and computing cost function, train classifiers by latent SVM.

The hardest part for object detection is that there are lots of variances, such as that arising from illumination, viewpoint, non-rigid deformation, occlusion, and intraclass variability. The deformable parts model is trying to capture these variances. It assumes each object is constructed by its parts. Thus, the detector will first find a match by coarser (at half resolution) root filter and then uses its part models to fine-tune the result. It uses *Histogram of Oriented Gradients* (HOG) features on pyramid levels before filtering and linear SVM for finding the different parts' locations of an object.

This method is still a widely-used algorithm that one can add other creative methods to, like combining segmentation and DPM [29].

Integral Channel Features

Integral Channel Features (ICF) [22] introduces new features by combining multiple registered image channels computed by linear and non-linear transformations. The author later argues that having a feature, like HOG, computed at one scale, allows the corresponding feature at higher and lower scales to be approximated by re-weighting [21]. Integrating those features and training using AdaBoost in a cascade way can achieve good accuracy on detecting pedestrians. Although it is a simple algorithm and works well on detecting rigid bodies like pedestrians, it may be hard to use in more general cases.

These transformations (*i.e.* colour, gradient, edge, gradient histogram, difference-of-Gaussian, thresholding, and the absolute value of Gaussian) are translationally invariant and only need to be calculated once. Training those features by cascading AdaBoost classifier, the whole method is fast and efficient on pedestrian detection. It is also empirically proved that most of the parameters in those transformations are not crucial.

2.1.2 Convolutional Neural Network based approaches

The current trend of using Convolutional Neural Networks for object recognition and detection in images started in 2012 when AlexNet [53] won the ILSVRC 2012 by a large margin from other competitors using classical algorithms [90]. AlexNet is a very efficient GPU implementation of the old LeNet [56] combined with ReLU nonlinearity as well as data augmentation and dropouts for reducing overfitting. AlexNet proved the effectiveness of using CNNs in solving computer vision problems and opened a new era in the field by kicking off the glorious comeback of CNNs. Since then, CNNs have been the primary focus of

object detection research, with the orthodox architecture being a large, multi-layered CNN, often pre-trained as an image classifier, combined with a method for creating bounding box proposals.

Inspired by AlexNet, Girshick *et al.* proposed Regions with CNN features (R-CNN) [32] by combining region proposals generated using selective search [109] and AlexNet's CNN for classifying those selected region proposals. Simultaneously, OverFeat [93] tackled the object detection problem by also using AlexNet and CNNs. OverFeat, instead of finding the region proposals separately, uses AlexNet to extract features at multiple evenly-spaced square windows in the image over multiple scales of an input image.

R-CNN and OverFeat represent the two first competing approaches of object detection in 2D images: either classify the regions proposed by another method or classify a fixed set of evenly spaced square windows in the image. The first mechanism uses region proposals that fit better to the objects of interest in the image than grid-like evenly spaced candidates in OverFeat. On the other hand, the second approach takes advantage of convolution operation to quickly regress and classify objects in the image in a sliding window fashion and is two orders of magnitude faster than R-CNN.

Later on, MultiBox [26] ended the aforementioned competition by introducing the *Prior Boxes* and *Region Proposal Networks* (RPN) ideas. After the dramatic improvements in object recognition demonstrated by R-CNN, most object detection works focused on improving the post-classification ranking alone while considering the proposal generation part solved by making use of salience-based object localization, in particular Selective Search, or intensively trying to classify evenly spaced patches of the image. MultiBox, instead, showed that CNNs could be used for region proposal generation purposes, and showed that object detection could be seen as a regression problem. They trained a network in a class agnostic manner which could generate a small number of bounding boxes as object candidates. Since then, most of the state-of-the-art methods in object detection [26, 87, 88, 104, 105] have a set of anchor (*A.K.A* "prior" or "default") boxes, generated based on a set of sliding windows or by clustering ground-truth boxes, from which bounding box regressors are trained to propose regions that contain salient objects that could fit better to the actual objects. Afterwards, the new competition became between direct classification in region-free methods such as YOLO [86] and SSD [61], and refined classification approaches in region-based methods like Faster R-CNN [88] and Mask R-CNN [36].

Terminologies

Before surveying works on CNN based object detection, we will provide some terminologies that will frequently be used in the rest of this section.

- **Bounding Box Proposal**, also known as *region of interest*, *region proposal*, and *box proposal* is a rectangular region of the input image that potentially contains an

object of interest inside it. These proposals could be generated by some heuristic search such as Objectness [2], Selective Search [109], Category Independent Object Proposals [25], Constrained Parametric Min-Cuts (CPMC) [12], and Multi-scale Combinatorial Grouping [3]; Or by Region Proposal Networks (RPN) which are being used widely in state-of-the-art object detectors.

A bounding box can be represented as a 4-element vector. Either storing its two corner coordinates $\{(x_0, y_0), (x_1, y_1)\}$, or more commonly storing its centroid, *width* and *height* $\{x, y, w, h\}$ in the image frame. A bounding box is usually accompanied by a *confidence* score of how likely the box contains an object.

The difference between two bounding boxes is usually measured by the L_2 distance of their vector representations. w and h can be log-transformed before the distance calculation as suggested in R-CNN [32].

- **Intersection Over Union** (IOU) or *Jaccard Similarity Coefficient* is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets as shown in Equation 2.1 and illustrated in Figure 2.1.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.1}$$

- **Region Proposal Network** (RPN) is a *fully convolutional* network that simultaneously predicts *object bounds* and *objectness scores* at each position. An RPN is trained end-to-end to generate high-quality region proposals. A work by Cireşan *et al.* in which they detect mitotic cells by applying a CNN to a regularly spaced square crops [14], was one of the pioneers in using CNNs for region proposals.

Meta-Architecture

Most of the state-of-the-art CNN-based object detectors follow almost the same principle for architecture design. They all have the first part in common, which is applying some convolutional *feature extractor* to the input image to obtain high level features. Then based on which category the detector belongs to, they either do the object classification and box regression in a single network (in region-free methods such as YOLO [86, 87] and SSD [61]), or use two separate networks, one for generating salient objects region proposals (box regression and objectness scoring), then feeding these regions to refined classifier for box refinement and object classification (in region-based approaches such as Faster R-CNN [88] and R-FCN [18]).

From the very first implementation of the CNN-based object detectors, a pre-trained CNN model was used either for sole classification purposes on cropped patches of image

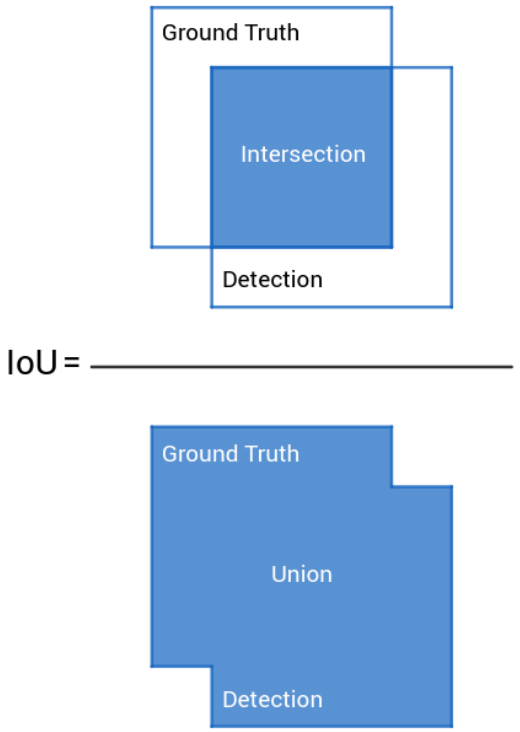


Figure 2.1: Intersection over Union or Jacard Index is a metric that measures the similarity between two bounding boxes

[53, 32, 31] or as the high-level feature extractor in all the newer models and the refined classifiers in region-based models. The choice of feature extractor is crucial as the number of parameters and types of layers directly affect memory, speed, and performance of the detector [43]. The first networks of this type such as R-CNN [32], OverFeat [93] and Fast R-CNN [31] used **AlexNet** for this purpose which was the only available CNN-based image classifier. As the community developed newer, more complex and accurate classification models, researchers started to use them to increase the accuracy and speed of detectors. Some of the well-known classifiers that are widely used in detectors are: **VGG-16** [94] which is a powerful, accurate classifier and **ResNet** [38] models by Microsoft Research group both of which have won many competitions such as ILSVRC and COCO 2015. **Inception v2** [46], based on which GoogLeNet [103] set the state-of-the-art in the ILSVRC 2014 classification and detection challenges, as well as its successor **Inception v3** [106] by Google Research group. Both of the Inception networks employed *Inception units* which made it possible to increase the depth and the width of a network without increasing its computational budget. The recent **Inception Resnet** [102], combines the optimization benefits conferred by *Residual connections* with the computation efficiency of Inception units. **Darknet-19** [87] is built off of prior works on network design as well as common knowledge in the field. Inspired by VGG models, it uses mostly 3×3 filters and doubles the number of channels after every pooling step [94]. Following the *Network in Network* work, they use global average

pooling to make the predictions, as well as 1×1 filters to compress the feature representation between 3×3 convolutions [60]. **MobileNet** [40] model has been shown to achieve the VGG-16 level accuracy on Imagenet with only 1/30 of the computational cost and model size. MobileNet is designed for efficient inference in various mobile vision applications. Its building blocks are depthwise separable convolutions which factorize a standard convolution into a depthwise convolution and a 1×1 convolution, effectively reducing both the computational cost and the number of parameters.

While both R-CNN [32] and Fast R-CNN [31] relied on external proposal generators, as mentioned before, recent works have shown that it is possible to generate the box proposals using CNNs as well [87, 104, 88]. In these works, it is typical to have a collection of boxes overlaid on the image at different spatial locations, scales, and aspect ratios that act as anchors. A model is then trained to make two predictions for each anchor: (i) a discrete class prediction for each anchor, and (ii) a continuous prediction of an offset by which the anchor needs to be shifted to fit the ground-truth bounding box [43].

Papers that follow the anchors' method, then minimize a combined classification and regression loss. For each anchor a, they first find the best matching ground-truth box b. If such a match can be found, they call a a *positive anchor* and assign it (i) a class label $y_a \in \{1...K\}$, and (ii) a box encoding $\phi(b_a; a)$, vector encoding of the box b with respect to the anchor a. If no match is found, they call a a *negative anchor* and set the class label to $y_a = 0$. If for the anchor a they predict the box encoding $f_{loc}(\mathcal{I}; a, \theta)$ and the corresponding class $f_{cls}(\mathcal{I}; a, \theta)$, where $\mathcal{I}$ is the image and θ the model parameters, then the loss for a is measured as a weighted sum of a location-based loss and a classification loss:

$$\begin{aligned}\mathcal{L}(a, \mathcal{I}, \theta) =& \alpha \cdot \mathbb{1}[a == Positive] \cdot \ell_{loc}(\phi(b_a; a) \\ & - f_{loc}(\mathcal{I}; a, \theta)) \\ & + \beta \cdot \ell_{cls}(y_a, f_{cls}(\mathcal{I}; a, \theta))\end{aligned} \tag{2.2}$$

where α and β are weights balancing the localization and classification losses. To train the model, Equation 2.2 is averaged over the anchors and minimized with respect to the parameters θ.

The choice of anchors has significant implications both for the accuracy and the computation. The main idea of using anchors was from the first MultiBox paper [26]. Anchors were called *box priors* by the authors in this paper and were generated by clustering the ground-truth boxes in the dataset. In more recent works, anchors are generated by tiling a collection of boxes at different scales and aspect ratios regularly across the image. The advantage of having a regular grid of anchors is that predictions for these boxes can be written as tiled predictors on the image with shared parameters (i.e., convolutions) and are reminiscent of the traditional sliding window methods as in Viola-Jones face detection

[110]. Faster R-CNN [88] and the second implementation of MultiBox [104] which called these tiled anchors *convolutional priors* were the first papers to take this new approach [43].

In the remainder of this section, we will survey the history of CNN based methods for object detection.

Region-Based Methods

These methods divide the object detection task into two stages. In the first stage, a dedicated module (a salient object detector or an RPN) is responsible for generating the region proposals. Then region-wise sub-networks are responsible for classifying and refining the generated regions as the detection bounding boxes. Some of this method's canonical works are:

- **R-CNN**
  R-CNN [32] was one of the pioneers in object detection that made use of CNNs and is the first region-based detector. It is a natural combination of heuristic region proposal method and CNN feature extractor. R-CNN consists of three modules.

  The first module generates around 2000 category independent region proposals for each input image. These proposals define the set of candidate detections. They use the Selective Search [109] method in its "fast mode" to generate about 2000 proposals for detection. They wrap each proposal to a 227×227 pixels image and forward propagate it through the second module to compute features.

  The second module is a CNN that extracts a fixed length features vector from each proposed region. They use the Caffe [48] implementation of the AlexNet [53] model to extract a vector of 4096 features from the region. Regardless of the size or aspect ratio of each arbitrarily shaped region from Selective Search, they wrap all pixels of the region with 16 pixels of padding in a tight bounding box of 227×227 pixels. Features are computed by forward propagating the wrapped image into the AlexNet network.

  Finally, in the third module, a set of class-specific linear SVMs are responsible for classifying the category of each region based on the extracted features from the previous layer. The SVM weights matrix is $4096 \times N$, where N is the number of classes. They pre-trained the model on the ImageNet as a sizable auxiliary dataset using image-level annotations for which the bounding boxes were not available, followed by domain-specific fine-tuning on the Pascal dataset to adapt the CNN module to the new detection task for which annotated data was scarce at that time. Figure 2.2 shows the system overview for R-CNN.

- **SPP-net**
  The main idea behind the Spatial Pyramid Pooling networks (SPP-net) [37] was using

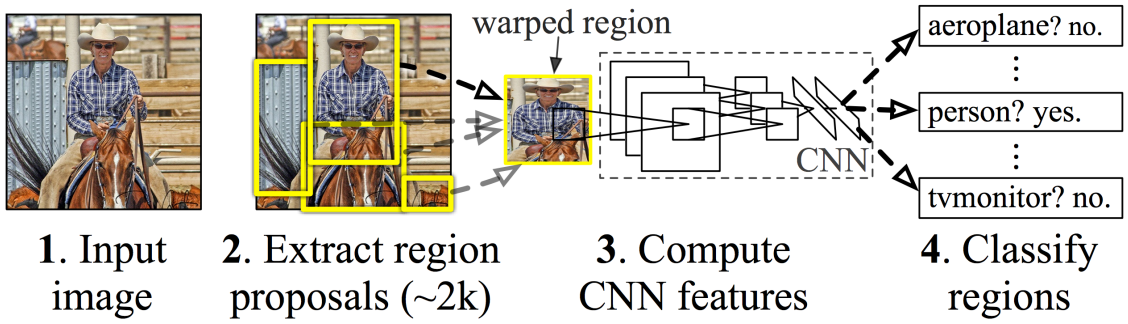


Figure 2.2: R-CNN [32] (i) takes an input image, (ii) extracts about 2000 bottom-up region proposals, (iii) computes features for each proposal using CNN, and then (iv) classifies each region using class-specific linear SVMs (© 2014 IEEE)

an extension of a Bag-of-Words model called *Spatial Pyramid Matching* [54] as the pooling layer in the network in order to eliminate the determined input size constraint and generate a fixed length output regardless of the input size. Convolutional layers in a CNN do not bound the input size, operate in a sliding-window manner and output the feature maps representing spatial arrangement of activations. On the other hand, the fully-connected layers need to have a fixed size input by definition. That is why all the CNN works prior to SPP-net (and most of them up to today) require a fixed input image size which is generated either via cropping or wrapping the image to the network's input size as shown in Figure 2.3. However, the cropped region may not contain the entire object, while wrapping contents may result in unwanted geometric distortions. Recognition can be compromised due to this contents loss or distortions. Besides, a pre-defined scale may not be suitable when object scale varies.

SPP, unlike the sliding window pooling, can generate fixed-size outputs regardless of the size of the inputs. It also uses multi-level spatial bins while the sliding window pooling uses only a single window, which makes it robust to the object deformations and gives it the ability to pool features extracted at variable scales.

This paper also introduced the idea of *multi-size* training, which was adopted later in the YOLOv2 [87], our baseline model. For a single network to accept variable input sizes, they approximated it by multiple networks that share all the parameters, while each network was trained using a fixed input size.

Another big difference between SPP-net and R-CNN was running the Convolutional layers for feature extraction on the image only once, while in R-CNN they repeatedly applied the deep convolutional layers to the raw pixels of thousands of wrapped regions per image. In this method, the feature maps from the entire image are computed only once on the input image, regardless of the number of object proposals generated from Selective Search [109]. In the SPP-net features are extracted for a proposal by max-

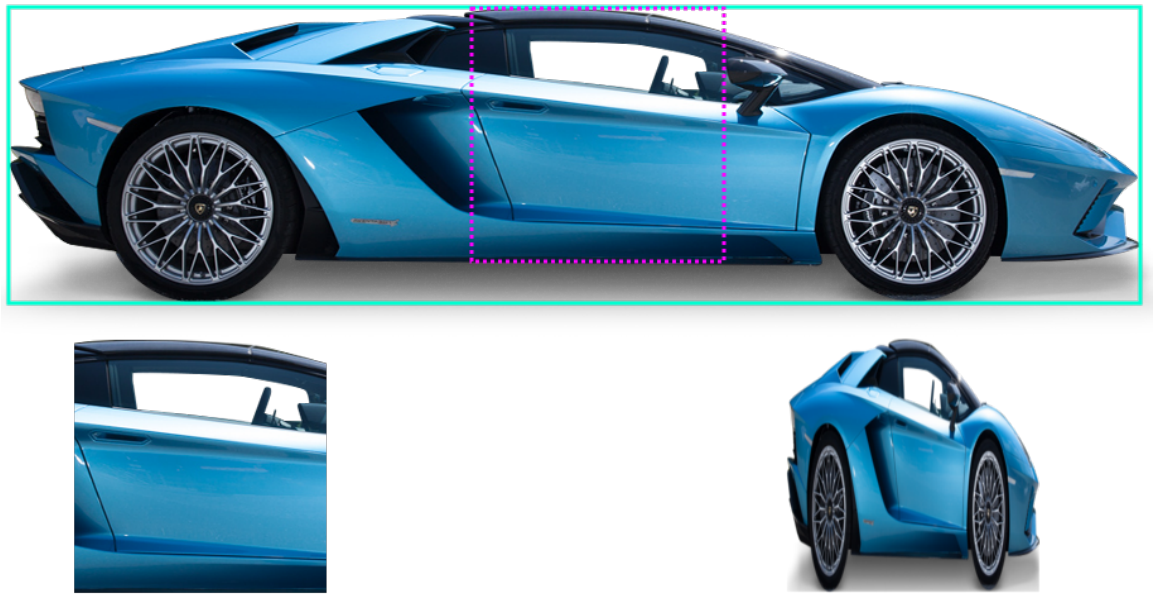


Figure 2.3: Most CNN methods accept fixed size images as input. These images are either generated via cropping the required size region from the original image (left) or wrapping the region in a fixed size window (right)

pooling the portion of the feature maps inside the proposal into a fixed-size output. It then uses the Spatial Pyramid Pooling (SPP) to pool multiple output sizes into a fixed-length representation. This representation is used for training the object classifier and box regressors. Pooling features from a shared global feature volume rather than pushing all the image crops through a full CNN like R-CNN provide two orders of magnitude speed up. Although SPP operations are differentiable, the authors did not do that and used ZFNet [123] trained only on the ImageNet without any fine-tuning.

- **Fast R-CNN**

  Following enhancements made to R-CNN by SPP-net, the R-CNN author embraced these ideas to speed up the R-CNN and add more contributions in their new method called Fast R-CNN [31]. Both previous methods used multi-stage training pipelines which involved extracting some features, fine-tuning a network with log loss, saving all the features for each bounding box proposal on the disk (which takes lots of time and memory space), using those features for training the SVMs, and finally fitting the bounding box regressors. In Fast R-CNN the author introduced single-stage training using multi-task loss. This end-to-end single-stage training made the training process faster and simultaneously could update all the network layers; also no additional memory was used to save the features on disk.

  Beside the trainable feature extraction layers, he replaced the SPP layers of SPP-net with new layers called Region of Interest (RoI) pooling. A Fast R-CNN network takes

as input an entire image and a set of bounding box proposals generated from Selective Search [109]. After extracting the features using the first several convolutional and max-pooling layers of the network (the feature extractors), for each object proposal, an RoI pooling layer extracts a fixed-length feature vector. RoI pooling is a special SPP where here only one pyramid level is used.

The final features vector after passing a sequence of fully-connected layers branches into two sibling output layers. One produces Softmax probability estimates for each object class is known by the network plus one catch-all "Background" class. The other one outputs four real numbers for each of the known classes that encode refined bounding box position for one of the known objects.

- **Faster R-CNN**
  Faster R-CNN [88] is a technique that merges the convolutional features of the full-image network with the detection network, thereby simultaneously predicting the boxes and objectness scores. The detection network (RPN) is trained end-to-end in an alternating fashion with the Fast R-CNN network; its objective is to produce good region proposals. In other words, Faster R-CNN is Fast R-CNN [31] with the heuristic region proposals replaced by RPN inspired by MultiBox [26].

  In Faster R-CNN, the RPN is a small three-layered CNN (3×3 Conv $\rightarrow 1 \times 1$ Conv $\rightarrow 1 \times 1$ Conv) looking at the global feature volume extracted by the fifth convolutional layer of the feature extraction network in sliding window fashion. The most prominent similarity between the MultiBox RPN and Faster R-CNN RPN is the usage of prior boxes (called "anchor boxes" in the Faster R-CNN) that are designed to be translation invariant and predicted from the top layer feature map. Each sliding window has nine anchor boxes relative to its receptive field (3 scales $\times$ 3 aspect ratios). The RPN does the bounding box regression and box confidence scoring for each anchor box. The whole pipeline is trainable by combining the loss of the box regression, box confidence scoring, and object classification into one common global objective function. Note that here the RPN looks at a small input region and anchor boxes hold both the centeral location and the box size, which are different from MultiBox's and YOLO's [86] design.

- **RFCN**
  Region-based, Fully Convolutional Networks (R-FCN) [18] is another approach for accurate and efficient object detection. In contrast to the aforementioned region-based detectors that apply a costly per-region subnetwork hundreds of times, this detector is fully convolutional with almost all the computation shared on the entire image. To achieve this goal, they propose position-sensitive score maps to address a dilemma between translation-*invariance* in image classification and translation-*variance* in object detection. For example, translation of an object inside a candidate box should produce

meaningful responses for describing how well the candidate box overlaps the object. This method can thus naturally adopt fully convolutional image classifier backbones, such as ResNets [38] and GoogLeNets [103].

Given the proposal regions (RoIs), the R-FCN architecture is designed to classify the RoIs into object categories and background. In R-FCN, all learnable weight layers are convolutional and are computed on the entire image. The last convolutional layer produces a bank of k^2 position-sensitive score maps for each category and thus has a $k^2(C+1)$-channel output layer with C object categories (+1 for background). The bank of k^2 score maps corresponds to a $k \times k$ spatial grid describing the relative positions. For example, with $k \times k = 3 \times 3$, the 9 score maps encode the cases of {top-left, top-center, top-right, ..., bottom-right} of an object category.

R-FCN ends with a position-sensitive RoI pooling layer. This layer aggregates the outputs of the last convolutional layer and generates scores for each RoI. Unlike R-CNN [32], R-FCN's position-sensitive RoI layer conducts selective pooling, and each of the $k \times k$ bin aggregates responses from only one score map out of the bank of $k \times k$ score maps. With end-to-end training, this RoI layer shepherds the last convolutional layer to learn specialized position-sensitive score maps.

- **Mask R-CNN**
  Mask R-CNN [36] is the biggest revolution in region-based CNN object detectors that extends the Faster R-CNN work to provide instance segmentation of objects in the image as well as providing the RoIs by adding a branch for predicting class-specific object masks, in parallel with the existing branch for bounding box recognition. Figure 2.4 shows sample outputs from Mask R-CNN.

  Adding the segmentation layers to the original Faster R-CNN architecture without any modifications, the Mask R-CNN authors realized that the regions of the feature maps selected by RoIPool layers were slightly misaligned from the regions of the original image. Since image segmentation requires pixel-level specificity, unlike bounding boxes, this naturally led to inaccuracies. Since RoIPool is not designed for pixel-to-pixel alignment between network inputs and outputs, Mask R-CNN replaces it with RoIAlign, which uses bilinear interpolation to compute the exact values of the input features at each sub-window instead of RoIPooling's max-pooling.

Region-Free methods

These methods treat the object detection as a single shot problem, straight from image pixels to the bounding boxes. The main advantage of these methods is high efficiency.

- **OverFeat**
  Although AlexNet [53] demonstrated an impressive localization performance in the

Figure 2.4: Results of Mask R-CNN [36] on the COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (© 2017 IEEE)

ImageNet 2012 [90] localization challenge, there has been no published work describing their approach, which makes OverFeat [93] the first paper to provide a clear explanation of how CNNs could be used for the localization task.

OverFeat uses a modified version of AlexNet model to extract features in multiple evenly-spaced square windows in a sliding-window fashion, over multiple scales on an input image. Even with this, however, many viewing windows may contain a perfectly identifiable portion of an object (say, the head of a dog), but not the entire object, nor even the centre of the object. This leads to decent classification but poor localization and detection. Thus, the idea behind OverFeat was to train the system to not only produce a distribution over categories for each window but also to produce a prediction of the location and size of the bounding box containing the object relative to the window. Then it would select a bounding box containing the object by accumulating the evidence for each category at each location and size.

For localization purposes, OverFeat replaced the classifier layers with a regression network and trained it to predict the object bounding boxes at each spatial location and scale. It then combined the regression predictions, along with the classification results obtained at each location. To generate the object bounding box predictions, they simultaneously run the classifier and regressor networks across all locations and scales. Since these two networks share the same feature extraction layers, only the final regression layers need to be recomputed after computing the classification network. The output of the final Softmax layer for a class c at each location provides a score of

confidence that an object of class c is present in the corresponding field of view. Thus a confidence is assigned to each bounding box. The predictions then get combined via a greedy merge strategy applied to the regressor bounding boxes. The final prediction is given by taking the merged bounding boxes with the maximum class scores. This is computed by cumulatively adding the detection class outputs associated with the input windows from which each bounding box was predicted.

- **MultiBox**

  As mentioned before, MultiBox [26, 104] is not an object detector but a CNN-based region proposal solution. The main argument in that work was that domain agnostic proposal generation had a principal drawback that the proposals come unranked or with very weak rankings, making it hard to trade the quality for running time. A better way of ranking the proposals is to cut down their number at generation. In the ideal case, we will be able to achieve high coverage with very few proposals, which is one object proposal per object in the image. This can improve not only the running time but also the quality, as the post-classification stage would need to handle fewer potential false positives. Having a strong proposal ranking function also provides a way to balance the speed/accuracy trade-off by simply selecting an appropriate threshold.

  The original MultiBox approach [26] directly produces bounding box coordinates and avoids linear scaling in the number of classes by making class-agnostic region proposals. The main contribution of this work was defining the object detection as a regression problem to the coordinates of several bounding boxes for the first time. In addition, for each bounding box the model predicts a confidence score of how likely the box contains an object. It popularized the ideas of region proposal networks and prior boxes, proving that a CNN can be trained to propose better region proposals than heuristic approaches. Since then, the heuristic methods have been fading out, being replaced by region proposal networks (RPN).

  The second version of MultiBox in 2015 [104] shows a comparable efficiency improvement compared to SPP-net [54] by drastically reducing the number and improving the quality of the region proposals via RPNs, which also associates a confidence score to each proposal. In this work, some ideas introduced by recent works since the previous version such as YOLO [86] and Faster R-CNN [88] are adopted.

- **YOLO**

  You Only Look Once (YOLO) [86] is a direct development on MultiBox. All of the object detection works prior to YOLO repurpose the CNN classifiers to perform object detection. Instead Redmon *et al.* frame object detection as a regression problem to spatially separate the bounding boxes and associated class probabilities. In this work, a single end-to-end CNN predicts the bounding boxes and class probabilities directly from the input image in a single forward pass.

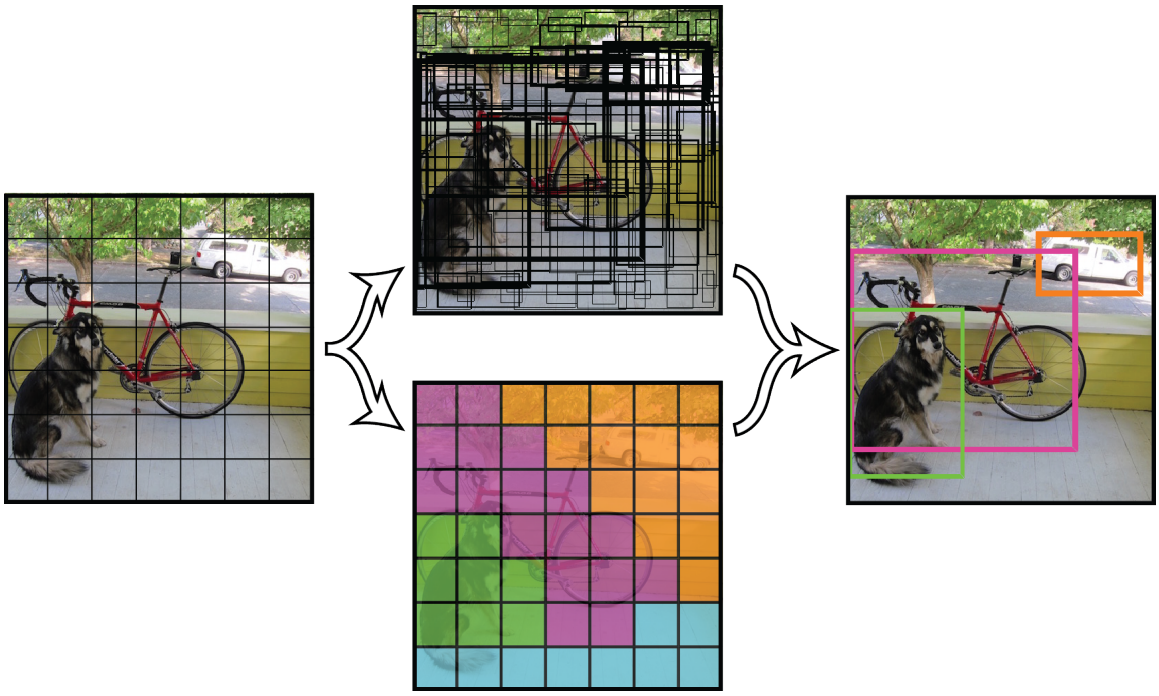


Figure 2.5: YOLO [86] models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts β bounding boxes, confidence for those boxes, and C class probabilities (© 2016 IEEE)

YOLO turns MultiBox from an RPN to an object detection method by adding a Softmax layer parallel to the box regressor and box classifier layers to directly predict the object class. YOLO divides the image into an $S \times S$ grid; if the centre of an object falls into a grid cell that cell is responsible for detecting the object. Each grid cell predicts β bounding boxes and a confidence score for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Each bounding box consists of 5 predictions: x, y coordinates of bounding box centroid relative to the grid cell, w, h the bounding box dimensions relative to the whole image, and finally the confidence prediction represents the IOU between the prediction box and the ground truth. Each grid cell also predicts C conditional class probabilities of $Pr(Class_i|Objectness)$. At the test time multiplying the conditional class probability for each grid by its confidence gives class-specific confidence scores for each box. The final output of the network is an $S \times S \times (\beta * 5 + C)$ tensor as shown in Figure 2.5.

YOLO was the first CNN-based object detector that could run in real-time. The base YOLO model with 24 convolutional layers can run at 45fps on an NVIDIA Titan X with 63.4mAP on the Pascal VOC which was more than twice the accuracy of the state-of-the-art in real-time object detection at that time, 30Hz PDM [91] with 26.1mAP on the same dataset. The authors introduced an even faster model called

Fast YOLO which can run at 155fps on the same GPU with 52.7mAP on the Pascal VOC, which was a direct competitor to 100Hz DPM [91] with 16.0mAP. Although YOLO is fast and accurate, it is limited in that each grid cell can only contain one object by construction, with the grid being quite coarse.

- **SSD**
  Single Shot MultiBox Detector (SSD) [61] is the direct competitor to YOLO. SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. In other words, SSD leverages the Faster R-CNN [88] RPN, using it to directly classify objects inside each prior box instead of just scoring the objectness. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes; this improves the diversity of prior boxes' resolutions by running the RPN on multiple convolutional layers at different depth levels.

  SSD was slightly faster than original YOLO with better accuracy, however, YOLOv2 wins the competition both in speed and accuracy.

2.2 State of The Art in Hand Detection

Hand detection and hand pose recovery are essential problems because of the importance of hands in communication and many other applications. Vision-based hand tracking and hand detection are amongst the most challenging tasks to accomplish in the modern human-computer interfaces. Although hand pose and gesture recognition are receiving an increasing amount of attention in the computer vision community, current hand tracking approaches often assume that the initial hand location is known to the classifier, thereby skipping the more difficult step of initial hand detection for tracker initialization or error recovery [96].

Furthermore, since low cost depth cameras recently became available, the focus has shifted from object detection and tracking in monocular videos, to image analysis using the more featureful depth maps. However, due to RGB-D imaging dependency on the infrared, depth cameras based on time of flight or structured light often perform poorly in the direct sunlight, limiting their applicability to indoor applications with controlled lighting conditions and distance. Additionally, RGB-D cameras are heavier, more power hungry, and the amount of data they transmit is enormous compared to monocular images.

As opposed to rigid or non-articulated objects such as faces, human hands lack clear discriminative features, making it difficult to design a robust hand detector based on low-level feature detection. Additionally, the human hand has 27 degrees of freedom, six of which represent translation and rotation of the wrist [24]. Due to high deformability, a hand's 2D silhouette can assume a wide variety of shapes, further complicating the robust

hand detection task. As a result, even the current state-of-the-art detection methods are unable to detect hands in video sequences robustly. Hence hand detectors are less successful than face detectors.

Moreover, current hand detection and tracking solutions all have limitations: some methods focus on constrained environments with a controlled background [97, 23, 8]. Other methods require the user to wear coloured gloves [113, 71] or use specialized hardware such as an RGB-D camera [45, 58, 63, 108] with limits on operational distance and environment. Many methods assume a uniform illumination because they use a simple skin colour or motion detection based segmentation.

State-of-the-art hand detectors are mostly multi-modal methods that combine several different features. Skin colour is a substantial module in most methods, which makes them vulnerable to the illumination changes. Gloves and extreme differences in skin colour are also challenging for these approaches. Most of these methods are too slow [30, 4, 116] to be used for real-time purposes such as HRI. Spruyt *et al.* have designed a near-real-time multi-modal method for hand detection and tracking [95] which only works from very close range, in a controlled environment. The closest work to ours is [116] in which using R-CNN for hand detection makes it too slow for real-time purposes and [7] which focuses on very close range hands in egocentric first-person views. The authors in [83] used the same dataset to re-trained the YOLOv2 model for hand detection in an automatic groceries picking detection scenario.

2.2.1 Hand Detection in Monocular Camera Images

Kölsch and Turk [51] trained the Viola-Jones [110] method for hand detection in several different postures. While the authors reported a reasonable detection rate between 65.8% and 92.23%, their method is view dependent, and can therefore only be used to detect hands in a predefined pose.

Just *et al.* [49] replaced the Haar-like features in the Viola-Jones method with the Modified Census Transform (MCT) to describe the texture of the neighbourhood of each pixel. The MCT is a feature descriptor that encodes the local texture information of a pixel as an ordered set of binary comparisons of pixel intensities between all pixels in a 3×3 neighbourhood and the average intensity within this neighbourhood. Calculating these features involves a simple and efficient convolution. The resulting detector outperforms the [51] detector in terms of recognition rates, but suffers from the same problem: due to the rigid model that is represented by the boosted classifier cascade, only predefined hand postures can be recognized.

In a work by Ong and Bowden [77] a hand detection method proposed which consists of a tree of boosted detectors. While the head of the tree returns hand position hypotheses, the remaining cascades focus on specific hand postures. Although this approach seems to outperform the two previous methods when used on a dataset with a uniform background,

the system fails in the case of any changes to the background. Furthermore, due to the rigid model that is represented by the boosted cascades at each leaf-node of the tree, only a specific set of hand postures can be detected. Based on this idea Stenger *et al.* [99] proposed a rigid model based hand detection and tracking system using a hierarchical Bayesian filter. In this work an extensive database of hand templates is hierarchically clustered based on their Chamfer distance, detection is performed by searching the hierarchical template tree, and temporal consistency is enforced during the tracking, utilizing Bayesian filtering. However, due to the computational complexity of the Chamfer distance metric, this method is as slow as three frames per second.

Stenger also proposed another two-stage hand detector [98]. The first stage is based on the simple colour and motion cues, colour cue represents skin colour likelihood, whereas the motion cue represents the difference in the pixel intensity between the current frame and the previous frame in a video. In the second stage, the hypotheses made by the first stage are tested by a classifier. The second stage classifier is a nearest neighbour classifier in which edge orientation and skin colour likelihood are combined in a feature vector. A distance measure is then calculated between the feature vectors that describe the bounding box of a hand location hypothesis, and each entry of a template database that was generated during the training. Similar to the earlier discussed rigid model based approaches, this detector can only detect hands in specific postures.

Mittal *et al.* followed the two-stage hand detection idea [4] in a parts-based detector. As opposed to the earlier discussed rigid model based detectors, a parts based detector is able to cope with deformable objects by modelling the relative position of object parts, together with their expected variance. Their method uses a DPM, based on HOG features. In their first stage, they use the HOG features to generate hand location hypotheses, while the second stage confirms or rejects these hypotheses, based on the skin colour obtained from face detection. The resulting detector by this approach can detect hands irrespective of their posture, in unconstrained environments with random backgrounds and illumination. However, the whole detection process takes approximately two minutes for an image as small as 640×360 pixels and is therefore not suited for real-time HRI applications.

Leibe *et al.* followed the parts-based model idea and designed a hand detector that learns a class-specific Implicit Shape Model (ISM) [57]. This ISM is a codebook of interest points, obtained by Scale-Invariant Feature Transform (SIFT) features. Furthermore, an offset vector from the interest point to the object's centroid in the training data is stored for each codebooks' entry. These vectors are used to build a probabilistic model of the spatial configuration of the interest points. At runtime, interest point descriptors in the image are matched against the codebook, and matching entries cast a probabilistic vote for the centroid of the object. The resulting detector is robust against the object deformations and occlusions, but needs between 4-7 seconds of processing time for an image of size 320×240. This approach is referred to as a generalized Hough transform since votes are accumulated

after which maxima are located in this accumulator. While this generalized Hough based detector is able to cope with deformable objects, partial occlusion and complex backgrounds, the method requires computationally expensive codebook matching at runtime, limiting its applicability in real-time applications. Moreover, in order to generate the codebook, large-scale clustering problems need to be solved.

Gall and Lempitsky [30] continued the work on parts-based object detectors using random forests. A random forest is trained using image patches that are selected using a sliding window. For each image patch, an offset vector to the object centroid is stored, describing its relative location. During classification, new image patches are classified by each tree in the forest, and the offset vectors that are stored in the leaf-nodes of each tree are used to cast a probabilistic vote for the object's centre. Despite the impressive results, the random forest based detector is not scale and/or rotation invariant. In order to cope with different scales, several scaled versions of the image are classified sequentially, yielding high processing times. The complete detection system needs approximately 6 seconds to process a 720×576 image, and can only detect objects at four different scales within this frame rate. Additionally, the detector is not rotation invariant.

Spruyt *et al.* [95] adapted the previous work for a real-time, scale and rotation invariant detection method for hands in video sequences. This detection system consists of a cascade three-stage classifier and can detect hands independent of their posture. Similar to the Viola-Jones method, each classifier outputs a set of hypotheses about possible hand locations, which are then verified by the next stage. The first-stage classifier is a Random Hough Forest parts-based classifier. A region of interest detector is used to select blob-like structures in the image. These regions are classified as being part of a hand, or part of the background. The parts-based Random Hough Forest classifier takes a set of regions of interest as its input. The regions of interest are found by searching for blob-like structures in the image by detecting isotropic regions of high entropy and low self-similarity across different scales. The first-stage classifier is used as a high-recall, low-precision classifier. It detects almost all hands in an image, but also returns many false positive detections. To eliminate these false positive detections, a second-stage classifier which is a simple Linear Discriminant Analysis (LDA) classifier is trained that quickly rejects many false positive detections by classifying the resulting hand bounding box as a whole, instead of classifying the parts that make up the hand. Finally, a third-stage classifier which is a simple decision tree that was trained using 10-fold cross-validation and pruned afterwards is used to evaluate whether an object is indeed a hand, by examining its temporal behaviour during the past N frames. Although this complicated pipeline can run as fast as 14fps which makes it near-real-time, it only works from very close range, in a controlled environment and is dependent on the temporal information in the video sequence and is not robust to the camera's ego-motion, thus not useful for a camera on a moving robot.

Afterward, the researchers' attention was drawn toward CNN-based methods. They started to adapt and retrain different CNN-based object detectors with the small number of datasets available for hand detection. With other advances in technology such as augmented reality and autonomous driving cars, the need for robust and real-time hand detection increased. EgoHands [7] introduced a complex egocentric interactions hand detection dataset which is useful for developing methods for detecting human hands from the first-person view, mostly for augmented reality applications, which we cover more thoroughly in Section 4.2.6. The VIVA challenge [19] concentrates on detecting drivers' and passengers' hands from inside of the cars. This research considers issues in sensing, analysis, modelling, and prediction of the parameters associated with drivers, occupants, vehicle dynamics and vehicle surroundings as well as transportation infrastructure. We used the provided training data in the VIVA dataset as described in Section 4.2.5, and used their benchmarking tool to evaluate our system between state-of-the-art CNN based hand detectors covered in Section 5.1.

2.3 State of The Art in Face Detection

Face detection in RGB images is one of the most comprehensively explored areas in the literature using both classical and CNN-based methods. Face detection is a necessary first-step in face recognition and verification systems, with the purpose of localizing and extracting the face region from the background. It also has application in several areas such as HRI, face tracking for surveillance, facial behavior analysis, facial attribute recognition (*e.g.* gender/age recognition and assessment of beauty), face relighting and morphing, facial shape reconstruction, organization and presentation of digital photo-albums, content-based image retrieval, video coding, and intelligent Human-Computer interfaces. Unlike the previous two sections in this chapter, there are many comprehensive surveys on face detection and recognition in the literature [122, 34, 41, 39].

There are many available benchmarking tools and datasets for face detection. Face Detection Dataset and Benchmark (FDDB) [47] is a dataset of face regions designed for studying the problem of unconstrained face detection. This dataset consists of bounding box annotations for 21% of a bigger dataset [9] that we have labelled for our work and discussed in Section 4.2.7. The authors further provide comprehensive results on 78 methods for face detection evaluated using their benchmark tool[1]. WIDER [119] is another face detection benchmark dataset, of which images are selected from the publicly available WIDER dataset. This benchmark tool includes 32,203 images which are separated randomly to 40%/10%/50% data as training, validation and testing sets. We have not included the WIDER's training data in our training dataset. Although the authors do not provide the

[1] `http://vis-www.cs.umass.edu/fddb/results.html`

annotations for the test data, we use their benchmarking tool to evaluate our face detector and compare it to the other state-of-the-art methods on their validation data in Section 5.2. Hence we will not provide an exhaustive survey on face detection like previous two sections; instead, we will mention the two widely used face detectors in HRI community due to their fast response time.

The structure of faces makes them relatively easy to detect. Nevertheless, only a few methods achieve the combination of enough high-accuracy and real-time performance required for HRI. OpenCV[2] includes an implementation of the famous Viola-Jones [110] face detector, which is fast and widely used in robotics; however, it detects so many false positives. dlib[3] is another open source face detector which makes use of Deep Metric Learning [121] method and HOG features for face detection. These make it better than the OpenCV detector in terms of accuracy but slower. Most of these methods cannot be repurposed for hand detection.

2.4 State of The Art in Close-Range HRI

Embodied, world-embedded interactions occur directly between humans and robots through mechanical or sensor-mediated interfaces. A useful property of this type of interaction, in contrast to remote control interfaces, is that robots observe humans directly using their onboard sensing, so they may not need to localize themselves in a shared coordinate frame. Also, human users can walk and work among the robots, and are not tied to an operator station [81].

Yan *et al.* [115] define *vision-based, audio-based, tactile-based*, and *range-based* as four modal for close-range interaction between human and social robots. Our un-instrumented, embodied, gesture-based HRI system falls under the vision-based systems category.

Gestures are a common mode of interaction in vision-based systems. Mitra and Acharya [65] have provided a comprehensive survey on gestural communication in which hand gestures were cited as the most expressive and the most frequently used means of human-computer communication. A work by Waldherr *et al.* [111], one of the earliest attempts in using hand gestures for HRI, describes a "natural" gesture-based interface for the control of a trash-collector robot. The interface used a camera to track a person and recognize gestures involving arm motion. A fast, adaptive tracking algorithm enables the robot to track and follow a person reliably through office environments with changing lighting conditions. Looper *et al.* [62] used an active depth camera mounted on a mobile robot platform to perform human detection, person following, and static gesture detection for communicating the human user's intents to the robot.

[2] `https://sourceforge.net/projects/opencvlibrary/`

[3] dlib, Author: King, DE, Access on: `http://dlib.net`

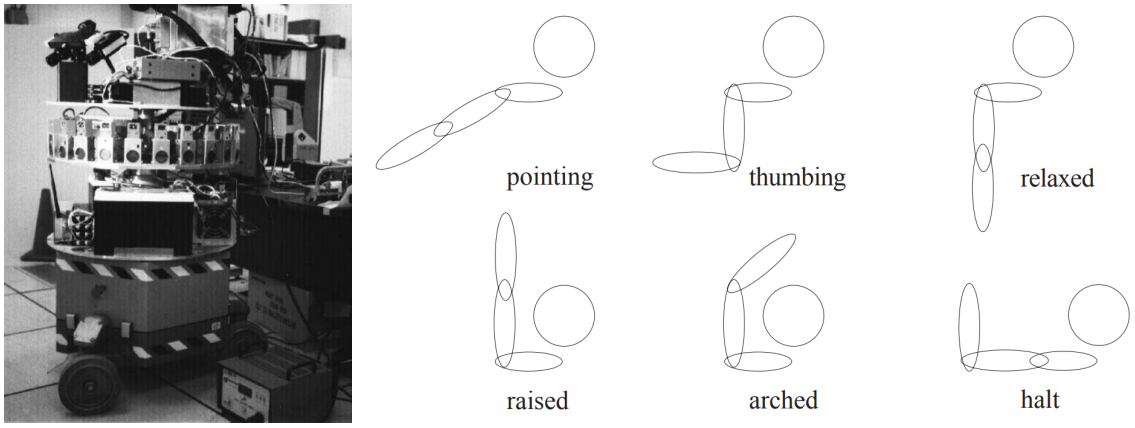
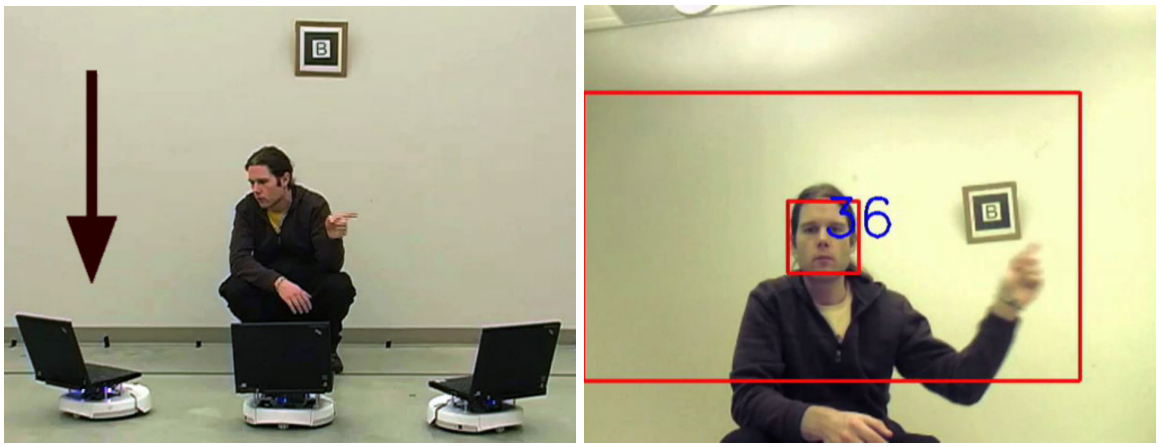


Figure 2.6: The complex active vision system developed by Kortenkamp *et al.* for extracting the human's 3D pose accompanied by the six gestures they managed to detect using their system [52] (© 1996 AAAI)

Kortenkamp *et al.* [52] presented a mobile robot with a 3D gesture recognition system. Using a coarse 3D model of a human to guide stereo measurements of body parts, the system was capable of recognizing six distinct gestures made by an un-instrumented human in an unaltered environment. They achieved this using a very complicated active vision approach, which focused the vision system's attention on small moving areas of space to allow for frame rate processing even when the person and/or the robot were moving. This system is shown in Figure 2.6.

Our research group has previously developed several vision mediated hands and faces engagement systems for selecting and commanding robots. Couture-Beil, for example, introduced a computer vision-based system that worked in real time to facilitate interactions between a single human and a multi-robot system [17]. As shown in Figure 2.7, a user first selects an individual robot from a group of robots by simply looking at it, and then commands the selected robot with a motion-based gesture. Robots estimate which robot the user is looking at by performing a distributed leader election based on the "score" of the detected frontal face. Based on this work, Milligan developed a system to select a group of robots by simply drawing a circle around them from his point of view with his un-instrumented hand [64].

Vision-based human detection and hand gesture recognition have always been crucial components in multi-modal HRI systems. The work by Steifelhagen *et al.* [100] is one of the earliest attempts for an integrated system which includes speech recognition and vision for color-based hand and face tracking to estimate pointing direction. Additionally, Perzanowski *et al.* [78] presented a multi-modal speech and gesture-based interface in which an active vision system was used to interpret pointing gestures as directional vectors, and to measure the distance between user's two hands. Following this work, Pourmehr *et al.* from our research group designed a multi-human multi-robot interaction system using a multi-modal



(a) (b)

Figure 2.7: (a) A user selects an individual robot by looking at it, and assigns it a task by waving his hand (b) A user-centric region is identified; the learned classifier uses optical-flow from the region to discriminate between gestures [17] (© 2010 IEEE)

system [82]. In that work the vision-based modality used Microsoft Kinect to detect humans that are reaching toward a robot- a specialization of pointing- to designate a particular robot for subsequent one-on-one interaction (Figure 2.8).

2.4.1 Gesture-based Situated Human-UAV Interaction

The rapid development of low-cost Unmanned Air Vehicles (UAVs) is enabling many valuable applications, and new industries are growing around them. In particular, small multi-rotor vehicles, which are relatively safe to operate around humans, provide a high degree of freedom that makes them rather more complicated to control than mobile robots. Although our system is a general purpose HRI system, un-instrumented, embodied, sensor-mediated human-UAV interactions are very good alternatives to conventional remote-control systems [70, 69]. Hence, we selected UAVs as the target platform to demonstrate our system.

There are not many practical works in the literature on close-range human-UAV interaction. A significant challenge in such systems is to perceive the commands issued by the user, which is the backbone of the problem that we attacked in this work. As we will show by surveying the literature, visual cues especially hands, head, and body gestures [65] are by far the most dominant means of communication.

Gesture detection can be done either through instrumenting the user or more naturally through embodied sensing [68]. Methods based on instrumentation usually rely on tangible interfaces such as gloves, special bodysuits, or marker-based optical tracking. Natural methods, on the other hand, rely on embodied sensors such as cameras to perform the detection. Robustness to body part occlusions as well as to viewpoint and intra-class variations are the most critical challenges for designing vision-based gesture recognition systems [16].

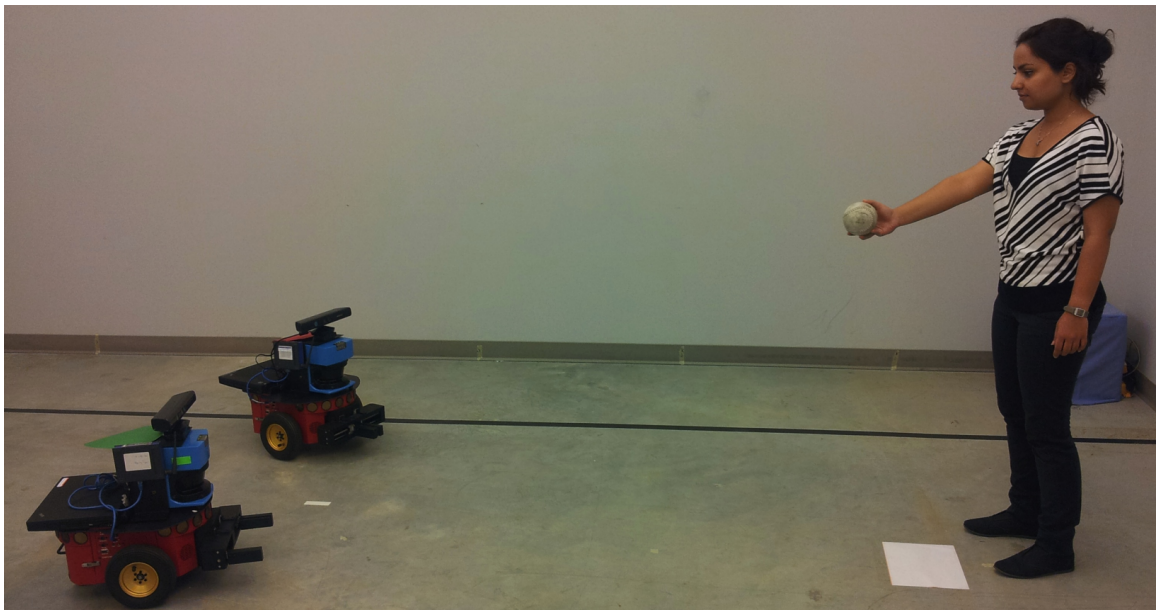


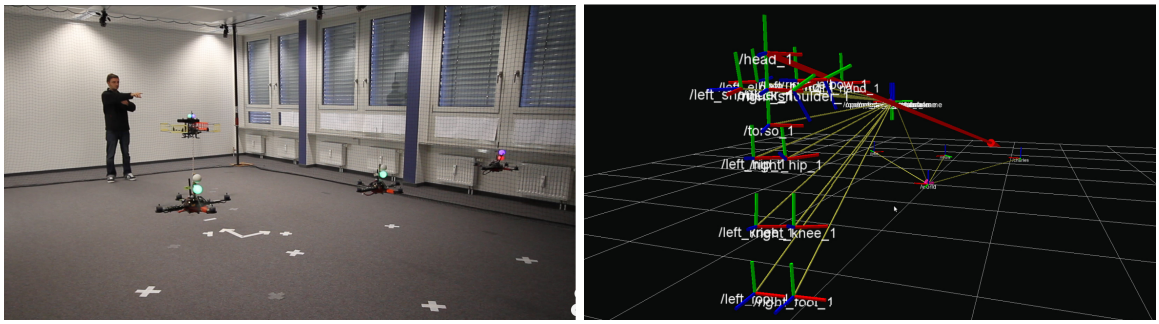
Figure 2.8: An un-instrumented person selecting one robot out of a group by offering it a ball (a modified pointing gesture) [82] (© 2013 IEEE)

Karam [50] found *hand gestures* as the most commonly used gestures among humans for communication. Hence he suggests hand gestures as the most natural gestures for human-computer interaction. Rautaray and Agrawal [85] identified *colour, shape, pixel values, 3D models*, and *motion* as the most common features used for vision-based gesture detection. There were some attempts to control the UAV using RGB-D cameras for skeleton detection and gesture recognition in a bench-top setting [20, 15, 80, 92], however, UAV embodied sensing systems are more useful, realistic, and natural.

In one of the earliest attempts to control UAVs using gestures [59], the authors designed a prototype environment in which a quad-copter that can carry a heavy payload (Asctec Pelican) carries a Microsoft Kinect sensor. This UAV hovers in front of the user and the skeleton tracking results obtained from Kinect's RGB-D data is used for gesture detection as illustrated in Figure 2.9a. The detected gestures next are interpreted into commands to control three Asctec Hummingbird quad-copters. The environment is equipped with a motion capture system for localizing (Figure 2.9b) and controlling the drones[4]. This work is very similar in nature to bench-top RGB-D based methods. However, this is the first demonstration of embodied RGB-D based sensing for Human-UAV Interaction.

Although Lichtenstern *et al.* [59], successfully demonstrated gesture recognition on board a UAV, the RGB-D sensor is almost stationary since the robot which carries the sensor is always hovering. Naseer *et al.* took one step further toward natural and embodied gesture

[4]Video demonstration: `https://youtu.be/oF3EcwNuO9Y`



(a) A user selects an individual robot by looking at it, and assigns it a task by waving his hand.

(b) A user-centric region is identified; a learned classifier uses optical-flow from the region to discriminate between gestures.

Figure 2.9: The prototyping environment of Lichtenstern *et al.* [59]. Motion capture system is used for localizing all the assets in the world (© 2012 ACM/IEEE)

recognition by using the same setup to do the maneuvers and follow a person [73]. They noticed that skeleton detection using a non-stationary RGB-D camera is not possible by utilizing the available skeleton detector (*e.g.* Microsoft and OpenAI). They used the UAV's movements to approximate and cancel out the ego-motion of the Asus Xtion Pro Live RGB-D sensor mounted on top an Asctec Pelican quad-copter. They also felt the need for a localization system in the world frame, for which they used a set of ceiling and wall mounted Augmented Reality markers. The person following was done by tracking the navigation waypoint positions generated by detected user's torso position in the world over time[5].

Using RGB-D sensors such as Microsoft Kinect imposes some limitations on embodied human-UAV interaction systems. They usually are big and more massive than small UAVs' payload, they consume much power, the generated data including depth information and point-clouds by these sensors are too big to be transmitted on wireless communication, and accompanying libraries require powerful on-board computing resources. They suffer from low range, resolution, and framerate in depth channel, they are only suitable for very close range and direct interactions, and also since these type of sensors rely on structured light or time-of-flight to estimate the depth information, their performance degrades a lot in an outdoor setting in the presence of sunlight. With recent advances in the technology of the RGB-D sensors, these limits are gradually lifted. For instance, Intel RealSense[6] technology made these sensors a lot smaller, faster, and more affordable. Nowadays there are drones such as Intel Aero[7] which are equipped with these sensors. There are also attempts to

[5]Video demonstration: `https://youtu.be/iaEKh4JYgqo`

[6]`https://www.intel.ca/content/www/ca/en/architecture-and-technology/realsense-overview.html`

[7]`https://software.intel.com/en-us/aero`

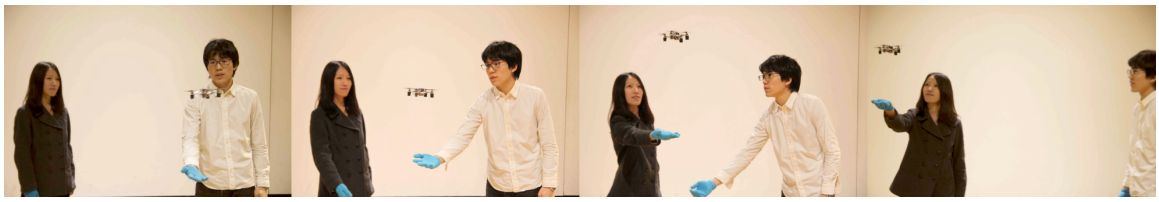


Figure 2.10: Passing the drone from one user to another by Miyoshi *et al.* [66] (© 2014 BCS)

solve the outdoor and direct sunlight problems using stereo-structured-lights (*e.g.* Intel RealSence R200, ZR300, and D400 series), or new time-of-flight technologies[8]. Nevertheless using monocular cameras on-board a UAV for vision-based human gesture and activity recognition is more alluring. These cameras are orders of magnitude cheaper, smaller, faster, and more accurate. Any small form factor consumer UAV is equipped with at least an HD camera that can stream stabilized video over WiFi.

Using a monocular camera for human-UAV interaction has also been explored by researchers. However, most of the state-of-the-art systems simplify the computer vision problem of detecting the human gestures by instrumenting the users with tangible devices or special clothing. The work by Miyoshi *et al.* [66] is an example of these systems. In this work, the authors designed a multi-modal (audio and vision) system, in which the UAV uses the audio modality to detect a whistle as *takeoff* and *land* commands. While hovering above the interaction partner's hand maintaining a fixed distance using its ultrasound distance sensor, it uses a downward installed monocular RGB camera to detect the hands covered with colour gloves. A basic colour segmentation onboard the UAV helps to detect the interaction partner and allows the user to control the UAV's movements.

Another similar work in which colour segmentation is used for detecting the user's hands is a work by Nagi *et al.* [71]. In this work, a group of networked AR-Drone quad-copters with onboard forward facing cameras transmit their video stream to an off-board computer for Viola-Jones-based [110] face detection ,and colour-segmentation-based hands and torso detection. Each UAV maintains a circular buffer of pair-wise relative distances between the centroid of the gloves and the jacket. This temporal estimation of the optical-flow of the detected human parts is averaged and normalized by each UAV. The gesture detection pipeline is only executed when this value is below a certain threshold.

To detect gestures, the system extracts 30 geometrical shape features from the detected glove blobs for the left and right hands which are distinguishable by their unique colour (*e.g.* convexity, aspect ratio, perimeter, *etc.*). A multi-class SVM is then used by each UAV to obtain a probabilistic decision vector that indicates the relative likelihood of each gesture in the vocabulary. Robots communicate their individually obtained vector with each other and

[8] `http://www.pmdtec.com/index.php`

Figure 2.11: Spatial arrangements of UAVs and the hand gestures vocabulary used by Nagi *et al.* [71] (© 2014 IEEE)

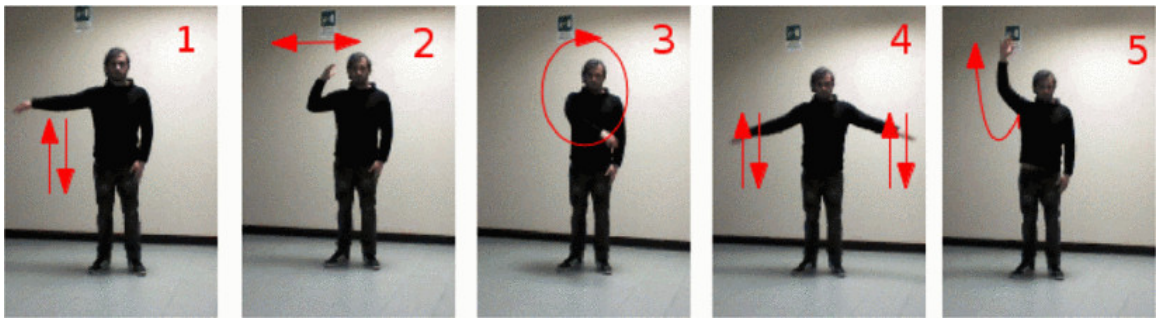


Figure 2.12: Motion-based hand gesture vocabulary designed by Costante *et al.* [16] for controlling and commanding a UAV (© 2014 IEEE)

run a distributed consensus algorithm to agree on the issued gesture by the human. Figure 2.11 shows this setup and a part of vocabulary used in this work. Most of the experiments presented in the original paper are performed on emulated and pre-recorded data[9].

Costante *et al.* [16] proposed a transfer learning approach for detecting and personalizing gestures for situated human-UAV interaction. They designed a vocabulary of five motion-based hand gestures illustrated in Figure 2.12. Then each user was asked to perform these gestures for a data gathering phase. Gestures are represented by encoding temporal variations of a Histogram of Optical Flows over regions of interest in the input frames. For each user, an SVM was trained to detect and distinguish these gestures. In the test time, a Viola-Jones [110] face detector was used to detect the interaction partner's face and Ahonen *et al.* [1]'s face recognition method to recognize the interaction partner and load the specific SVM for that user to classify their gestures.

The authors show that the proposed transfer learning based gesture recognition approach outperforms the linear classifiers that are trained either on the user's small training set or the whole or subset of the gesture database excluding the user's training set.

The most recent work in the literature is a work by Sun *et al.* [101], in which they have equipped a DJI Matrice M100[10] with a USB camera and an Intel® Core™i7 NUC

[9]Video demonstration: `https://youtu.be/G2tyV2USjG8`

[10]`https://www.dji.com/matrice100`

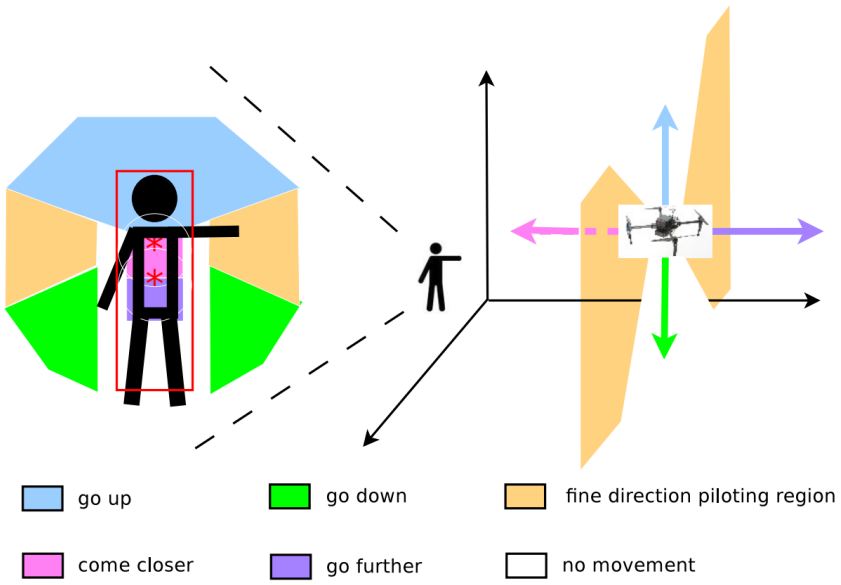


Figure 2.13: Sun *et al.* [101] used arm pointing gesture detection for piloting a UAV. They mapped the forward, lateral, and altitude movements of the UAV to the 2D image plane (© 2017 IEEE)

computer[11] to achieve a monocular gesture-based piloting system for aerial robots. They simplified the problem with some assumption: the user will always be visible for the aerial robot, there are no other people in the field of view of the UAV's camera, and to compensate for the long distance between the user and the aerial robot, camera motion, uncontrollable outdoor lighting conditions, *etc.*. They considered arm gestures instead of hand gestures. As they were not able to distinguish between the user's right and left hands, they first detect the user's torso and the 2D projection of her arms on the image plane using skin colour segmentation. Then command the UAV to fly in the direction pointed to by the user, with a velocity proportional to the length of the distance from the user's hand to the centre point between her shoulders.

For their gesture vocabulary design, they have projected 3 DOFs of the UAV's 4 DOFs (*i.e. forward, lateral,* and *altitude*) to a 2D image image plane as illustrated in Figure 2.13. Although this method looks promising, it suffers from environmental (*e.g.* illumination, background, *etc.*) changes. Additionally, because it relies entirely on arms' skin detection for gesture recognition, it is not robust for different skin colour tones, and the user has to wear short sleeves so that her arms are visible from far away.

Our research group has been working on natural, embodied, and un-instrumented human-UAV interactions for a while. Monajjemi *et al.* [70] were among the first to illustrate that one can command a flying UAV using only a monocular camera's RGB image. They introduced

[11] `https://www.intel.com/content/www/us/en/products/boards-kits/nuc.html`

Figure 2.14: Monajjemi *et al.* [70] showed that using vision-based methods such as face detection and calculating optical-flow in the image frame, one can command flying UAVs without any extra instrument (© 2013 IEEE)

the first un-instrumented commanding to flying robots using hand gestures. Similar to Nagi *et al.* [71], a team of multiple AR-Drones utilize their onboard forward-facing camera to detect human faces and to detect hand gestures. An OpenCV based face detection combined with a Kalman filter is used to robustly track a single human's face. The face tracker also reports the quality of the detected face or "face score". After detecting the user's face, two rectangular regions are considered around her face for Optical-Flow calculation. Dense Optical-Flow detection in any of the left and/or right regions indicates the corresponding hands' waving gesture[12].

The user can select and deselect each robot by a single-hand waving gesture (the right and left hand correspondingly), then use dual-hands waving gesture for commanding the selected robots to execute a task (*e.g.* takeoff, land, etc.). They handled the rapid ego-motion of the UAVs' camera by using a Kalman Filter to smooth the face position estimates. A nearest neighbour data association strategy is used to determine which detected face to use as the measurement input, using a Mahalanobis distance derived from the estimated covariance of the candidate faces. Figure 2.14 shows their setup and the calculated Optical-Flow for hand waving gesture detection.

This system formed the close-range interaction part of a bigger system. Monajjemi *et al.* used their system in the first demonstration of end-to-end far-to-near situated interaction between an un-instrumented human user and an initially distant outdoor autonomous UAV [67]. In this work, the user uses an arm-waving gesture as a signal to attract the UAV's attention from a very long range ($\approx 30m$). Once this signal is detected, the UAV approaches the user using appearance-based tracking until it is close enough to detect the human's face. Once in this close-range interaction setting, the user is able to use hand gestures to communicate her commands to the UAV using the previously mentioned system [70].

[12]Video demonstration: `https://youtu.be/xHH3GvZ52xg`

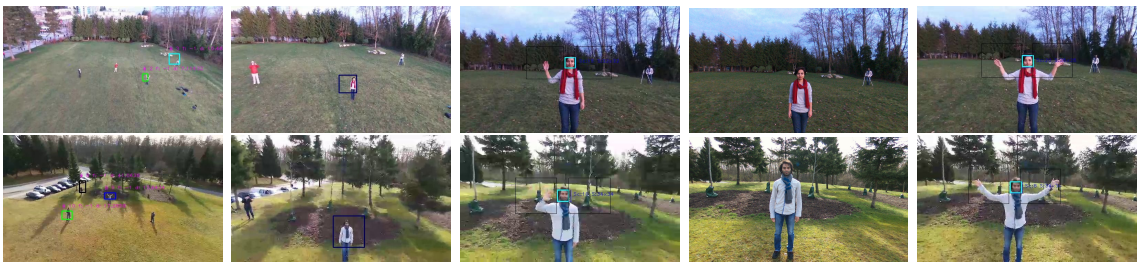


Figure 2.15: The flying robot's view from left to right: (i) The user initiates the interaction with the UAV using a dual-arm waving gesture in the presence of other humans (distance is $\approx 30m$) (ii) The UAV approaches the user using an appearance based tracker and a custom cascade controller (iii) The user asks the UAV to take a picture of her using a single hand waving gesture (iv) The resulting portrait (v) The user terminates the interaction by performing a dual hand waving gesture [67] (© 2016 IEEE)

Throughout the interaction, the UAV uses coloured-light-based feedback to communicate its intent to the user[13].

The gestures vocabulary for this work was limited to *single-hand* waving gesture for taking a selfie, and *dual-hands* waving gesture for terminating the interaction ("Bye Bye"). Although these gestures are robust and reliable, and the user does not need any particular instrument to interact with the system, but the gestures vocabulary is rather sparse. These gestures are limited to the *right*, *left*, and *dual* hands waving. Besides, calculating the Optical-Flow over a period decreases the response time of this system, and any frame loss in the video stream can give a hard time to the gesture detection system. Hence this thesis is about solving the mentioned dilemmas in close-range, embodied, un-instrumented interaction with robots.

2.4.2 Gestural Vocabulary for Human-UAV Interaction

Using gestural interfaces for communicating humans' intent to UAVs has become very popular recently. Monajjemi argues that the interaction interface should support natural interaction schemes, require naturalistic embodiment, and should preferably work when humans are not instrumented, *i.e.* no operator control unit is required, and the person carries no other dedicated equipment or clothing [68].

Peshkova *et al.* [79] discussed that for "natural" interaction with UAVs the gestures vocabulary should be *coherent*: which means there is a logical relationship between its components, and *intuitive*: that is the way it works corresponds to our expectations. They consider a gesture set to be *coherent* if all its gestures adhere to one and the same metaphor. This metaphor evokes a particular mental model that, in turn, defines a specific behaviour or, in the considered example, certain gestures. They additionally consider a gesture set to

[13]Video demonstration: `https://youtu.be/6kKuGH0B8XY`

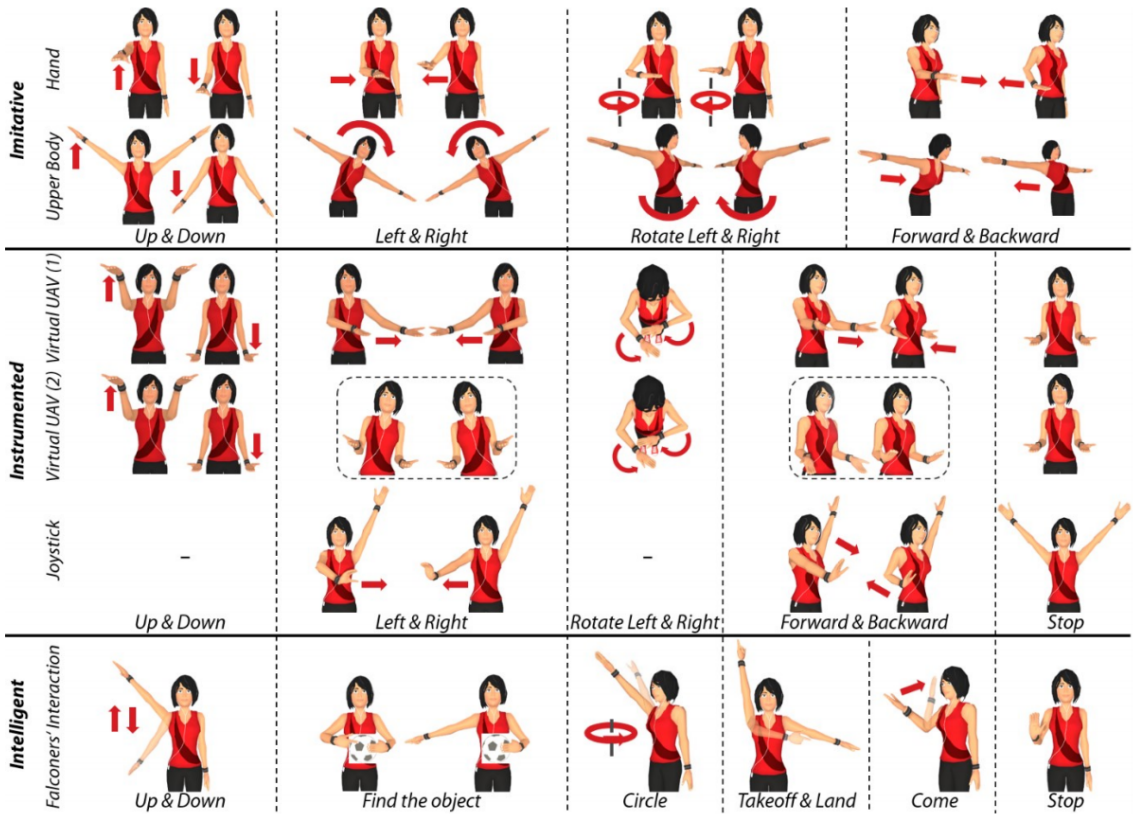
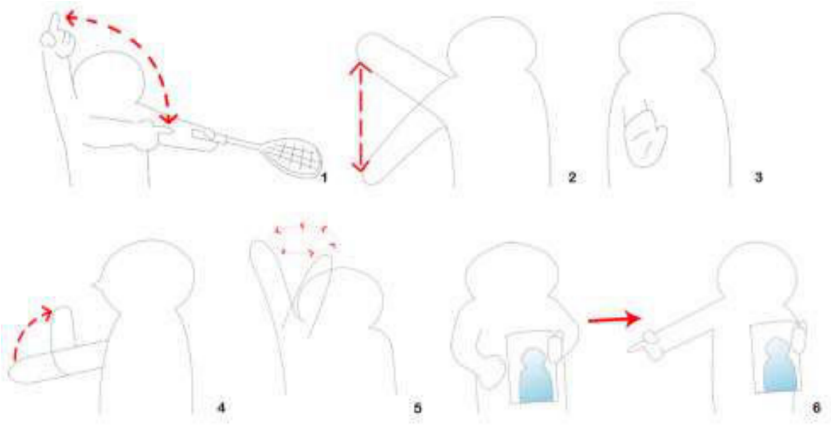


Figure 2.16: Examples of gesture sets (minor rows) for each of the three classes of mental models (major rows) for the specified commands (columns) in Peshkova *et al.* [79]'s survey (© 2017 IEEE)

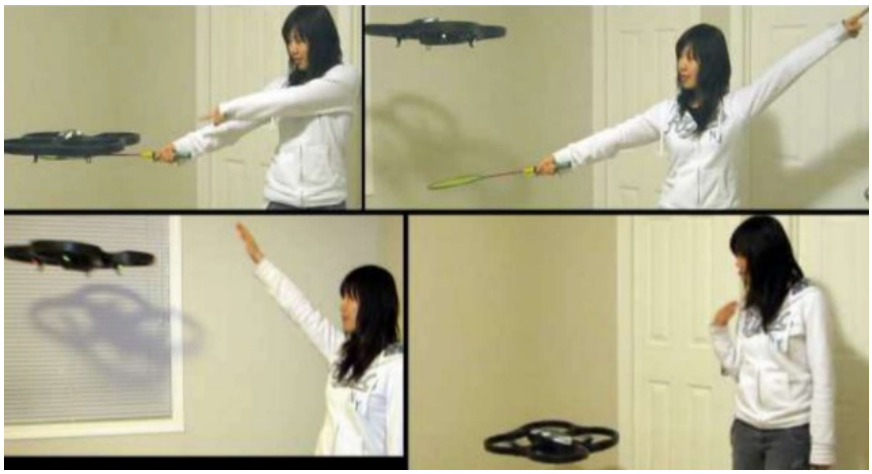
be *intuitive* if a single hint is enough to define all its gestures. In order to evoke a certain mental model and, as a result, a specific behaviour, a user needs to understand the metaphor that guides to the intended mental model.

Based on these assumptions, they divided the vocabularies into three different classes: The *Imitative* class implies that a vehicle is capable of imitating movements performed by an operator. They further divided this class to *hand, head, upper-body* and *full-body* gestures. The only vision-based method among those is the upper-body method introduced in [80]. The *Instrumented* class suggests that an operator controls a vehicle through an imaginary intermediate link that can be an imaginary physical object (*e.g.* a joystick), a link that allows manipulating a vehicle like a marionette or the ability to use super force to move a vehicle without touching it[14]. The *Intelligent* class is that a UAV is treated as an intelligent creature; this explains the fact that, in many cases, this class is deemed to resemble the natural interaction the most. Most of the previously introduced works fall under this class [59, 71, 70, 74]. Figure 2.16 demonstrates the resulting classification.

[14]Similar work video demonstration: `https://youtu.be/hfq2SisPvCU`



(a) The hand gesture vocabulary introduced by Ng and Sharlin.



(b) Communicating commands to a UAV through gestures.

Figure 2.17: Ng and Sharlin [74]'s proposed idea for co-located and direct interaction with UAVs (© 2011 IEEE)

Ng and Sharlin [74] were amongst the first researchers to explore the idea of co-located and un-instrumented interaction with real flying robots. Inspired by interaction schemes used by humans to communicate with birds (*e.g.* falconers and hawks), the authors crafted a set of hand and arm gestures to communicate commands to a UAV. The command set consists of *takeoff, land, ascend, descend, stop, come closer, circle* and *find* commands as shown in Figure 2.17a. The authors used the designed system to perform a series of Wizard of Oz experiments with two participants. They asked the participants to command the UAV using the designed gestures set from close proximity while the UAV was flying. A human operator manually controlled the UAV to imitate the execution of the participant's issued commands. The authors observed that participants were "very engaged" when performing gesture-based interaction with a flying robot and interacted with it as if it was a pet. They conclude that natural interaction with a flying robot is "easy to understand and perform".

Pfeil *et al.* [80] explored 3D gesture metaphors for indirect and non-embedded human-UAV interaction using a Microsoft Kinect depth sensor. The authors discussed the design and implementation of five interaction techniques using a Microsoft Kinect, based on the metaphors inspired by flying UAVs, to support a variety of flying operations a UAV can perform. Techniques include: (i) a first-person interaction metaphor where a user takes a pose like a winged aircraft, (ii) a game controller metaphor, where a user's hands mimic the control movements of console joysticks, (iii) the Throne where the user imitates the role of a king or a queen, (iv) "proxy" manipulation, where the user imagines manipulating the UAV as if it was in their grasp, and (v) a pointing metaphor in which the user assumes the identity of a monarch and commands the UAV as such. These gestures are shown in Figure 2.18.

Taralle *et al.* [107] conducted a research on defining an efficient gesture vocabulary for interaction between infantrymen and their accompanying UAVs in battlefields. The authors asked a group of 39 volunteers, whom separated into three groups, to propose, elect, and evaluate gestures for the following tasks: *takeoff, land, to the next waypoint, to the previous waypoint, stop, to the base, validate* and *cancel* with some constraints such as: being one-handed, intuitive, and avoid large and tiresome movements. The final selected gestures are shown in Figure 2.19.

Cauchard *et al.* [13] conducted another study with the same setup as [74] around "how users interact naturally with flying robots"[15]. Results showed a strong agreement between the participants for many interaction techniques, as when gesturing for the drone to stop. They also mentioned that people interact with drones as with a person or a pet, using interpersonal gestures, such as beckoning the drone closer. Tasks were split into five categories: (i) "within the body frame commands" such as *fly closer*, (ii) "outside the body frame commands" such as *fly further away*, (iii) "general motion" such as *takeoff*, (iv) "relative to the user commands" such as *follow*, and (v) "photo commands" such as *take a selfie*. Based on their individual subjective ratings, the participants found their interaction experience natural, safe and not physically or mentally demanding. Majority of the users expressed that they felt in total control of the drone. They report four task-gesture pairs with highest agreement scores among participant as illustrated in Figure 2.20.

[15]Demonstration video: `https://youtu.be/vrWF3t7a_HU`

Figure 2.18: Pfeil *et al.* [80] explored 3D gesture metaphors for indirect and non-embedded human-UAV interaction (© 2013 ACM)

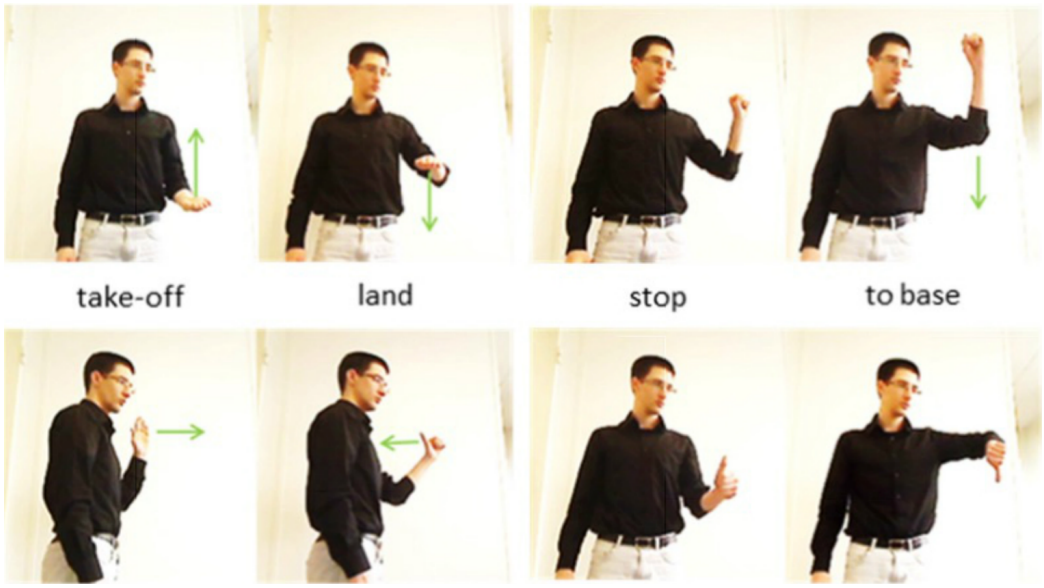


Figure 2.19: The final 8 gestures selected by Taralle *et al.* [107] from initial 160 proposed candidates. The association score calculated based on the matching performed by the third group is 94% (© 2015 ACM)

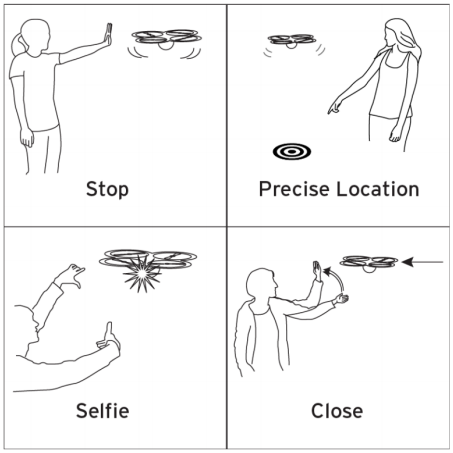


Figure 2.20: The getsure-action pairs with the highest agreement score among participants from the work by Cauchard *et al.* [13] (© 2015 ACM)

Chapter 3

Method

3.1 Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks such as driving with little conscious thought. You Only Look Once (YOLO) [86] is a generic object detection system, in which a single CNN predicts bounding boxes and probabilities for objects directly from the input image pixels, with only one single forward pass on the image. YOLO trains on full images and directly optimizes detection performance. YOLO is extremely fast. It also reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time, so it implicitly encodes contextual information about the classes as well as their appearance and learns generalizable representations of objects. The second version of this work YOLOv2 [87] has been proved to produce the state-of-the-art results in both accuracy and speed by outperforming other near-real-time CNN object detectors such as Faster R-CNN [88] and SSD [61].

3.1.1 YOLO: You only look once

The main idea behind YOLO is that the model only looks once to the image. In order to do that, the authors frame object detection as a regression problem to spatially separate bounding boxes and associate class probabilities. A neural network predicts bounding boxes and class probabilities directly from the full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on the detection performance.

YOLO divides the input image into an $S \times S$ grid. If the centre of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts β bounding boxes and a confidence score for each bounding box. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Equation 3.1 defines the confidence. If no object exists in

that cell, the confidence scores should be zero. Otherwise, we want the confidence score to be equal to the I = IOU between the predicted box and the ground truth.

$$Pr(Object) * I_{pred}^{truth} \tag{3.1}$$

Each bounding box consists of 5 predictions: x, y, w, h, and a confidence score. The (x, y) coordinates represent the centre of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally, the confidence prediction represents the IOU between the predicted box and any ground truth box.

Each grid cell also predicts C conditional class probabilities, $Pr(Class_i|Object)$. These probabilities are conditioned on the grid cell containing an object. One of the disadvantages of YOLO is that it only predicts one set of class probabilities per grid cell, regardless of the number of boxes β. Thus if two objects' centres fall in a similar grid cell, the system can only detect the most dominant (*i.e.* highest confidence) one.

At test time, multiplying the conditional class probabilities and the individual box confidence predictions (Equation 3.2) gives class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

$$Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Class_i) * IOU_{pred}^{truth} \tag{3.2}$$

The network architecture is inspired by the GoogLeNet model [103] for image classification. It has 24 convolutional layers followed by 2 fully-connected layers. Instead of the inception modules by GoogLeNet, simple 1×1 reduction layers followed by 3×3 convolutional layers is used, similar to Lin *et al.* [60]. Table 3.1 shows the network architecture for Pascal VOC 20 classes object detection. The input size for the network is 448×448, it uses $S = 7$ and $\beta = 2$. It divides the image into an $S \times S$ grid and for each grid cell predicts β bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (\beta * 5 + C)$ tensor, so the final prediction is a $7 \times 7 \times 30$ tensor.

In this architecture, like any other CNN-based object detector, the initial convolutional layers of the network extract features from the image while the fully-connected layers predict the output probabilities and coordinates. The first 20 convolutional feature extractor layers were pre-trained on the ImageNet 1000-class competition dataset [90] followed by an average-pooling layer and a fully-connected layer. The trained model then converted to perform detection, based on the method by Ren *et al.* [89] which suggested adding both convolutional and fully-connected layers for object detection task. The final layer predicts both class probabilities and bounding box coordinates. In YOLO the bounding box width and height is normalized by the image width and height so that they fall between 0 and 1. The bounding box x and y coordinates are also parametrized to be offsets of a particular grid cell location, so they are also bounded between 0 and 1.

	Type	Filters	Size	Stride	Input	Output
0	Convolutional	64	7×7	2	$448 \times 448 \times 3$	$224 \times 224 \times 64$
1	MaxPooling		2×2	2	$224 \times 224 \times 64$	$112 \times 112 \times 64$
2	Convolutional	192	3×3	1	$112 \times 112 \times 64$	$112 \times 112 \times 192$
3	MaxPooling		2×2	2	$112 \times 112 \times 192$	$56 \times 56 \times 192$
4	Convolutional	128	1×1	1	$56 \times 56 \times 192$	$56 \times 56 \times 128$
5	Convolutional	256	3×3	1	$56 \times 56 \times 128$	$56 \times 56 \times 256$
6	Convolutional	256	1×1	1	$56 \times 56 \times 256$	$56 \times 56 \times 256$
7	Convolutional	512	3×3	1	$56 \times 56 \times 256$	$56 \times 56 \times 512$
8	MaxPooling		2×2	2	$56 \times 56 \times 512$	$28 \times 28 \times 512$
9	Convolutional	256	1×1	1	$28 \times 28 \times 512$	$28 \times 28 \times 256$
10	Convolutional	512	3×3	1	$28 \times 28 \times 256$	$28 \times 28 \times 512$
11	Convolutional	256	1×1	1	$28 \times 28 \times 512$	$28 \times 28 \times 256$
12	Convolutional	512	3×3	1	$28 \times 28 \times 256$	$28 \times 28 \times 512$
13	Convolutional	256	1×1	1	$28 \times 28 \times 512$	$28 \times 28 \times 256$
14	Convolutional	512	3×3	1	$28 \times 28 \times 256$	$28 \times 28 \times 512$
15	Convolutional	256	1×1	1	$28 \times 28 \times 512$	$28 \times 28 \times 256$
16	Convolutional	512	3×3	1	$28 \times 28 \times 256$	$28 \times 28 \times 512$
17	Convolutional	512	1×1	1	$28 \times 28 \times 512$	$28 \times 28 \times 512$
18	Convolutional	1024	3×3	1	$28 \times 28 \times 512$	$28 \times 28 \times 1024$
19	MaxPooling		2×2	2	$28 \times 28 \times 1024$	$14 \times 14 \times 1024$
20	Convolutional	512	1×1	1	$14 \times 14 \times 1024$	$14 \times 14 \times 512$
21	Convolutional	1024	3×3	1	$14 \times 14 \times 512$	$14 \times 14 \times 1024$
22	Convolutional	512	1×1	1	$14 \times 14 \times 1024$	$14 \times 14 \times 512$
23	Convolutional	1024	3×3	1	$14 \times 14 \times 512$	$14 \times 14 \times 1024$
24	Convolutional	1024	3×3	1	$14 \times 14 \times 1024$	$14 \times 14 \times 1024$
25	Convolutional	1024	3×3	2	$14 \times 14 \times 1024$	$7 \times 7 \times 1024$
26	Convolutional	1024	3×3	1	$7 \times 7 \times 1024$	$7 \times 7 \times 1024$
27	Convolutional	1024	3×3	1	$7 \times 7 \times 1024$	$7 \times 7 \times 1024$
28	FullyConnected				$7 \times 7 \times 1024$	4096
23	FullyConnected				4096	1470 ($7 \times 7 \times 30$)

Table 3.1: The YOLO detection network has 24 convolutional layers followed by 2 fully-connected layers. Alternating 1×1 convolutional layers reduce the features space from the preceding layers. The horizontal line separates the feature extraction part of the network from bounding box regressor/classifier.

The loss function of the end-to-end model must then simultaneously solve the object detection and object classification tasks. This function shown in Equation 3.3 simultaneously penalizes incorrect object detections as well as considers what the best possible classification would be.

$$
\begin{aligned}
\mathcal{L} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{\beta} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{\beta} \mathbb{1}_{ij}^{obj} [(\sqrt{\omega_i} - \sqrt{\hat{\omega}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
+ \sum_{i=0}^{S^2} \sum_{j=0}^{\beta} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \\
+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{\beta} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\
+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{3.3}
$$

From the Equation 3.3, $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$ are two independent hyper-parameters. The former increases the loss from bounding box coordinate predictions, and the latter decreases the loss from confidence predictions for boxes that don't contain objects. Similarly, $\mathbb{1}_{i}^{obj}$ denotes if an object appears in the ith cell and $\mathbb{1}_{ij}^{obj}$ denotes that the jth bounding box predictor in the cell i is responsible for that prediction. The 5 bounding box outputs of the box j of cell i are the coordinates of the center of the bounding box (x_{ij},y_{ij}), height h_{ij}, width w_{ij} and a confidence index C_{ij}. The values with a hat are the real one read from the ground truth and the one without hat are the predicted ones, the real value for the confidence score for each bounding box $\hat{C}_i$ is the IOU of the predicted bounding box with the one from the ground truth. The loss function tries to make the confidence score equal to the IOU between the object and the prediction when there is an object in the grid cell otherwise tries to converge the confidence score close to 0. Note that the loss function only penalizes classification error if an object is present in that particular grid cell (because of the conditional class probability discussed earlier). It also only penalizes bounding box coordinate errors if that predictor is "responsible" for the ground truth box (*i.e.* has the highest IOU among all predictors in that grid cell). The square root in calculating the bounding box dimension loss is present so that errors in small bounding boxes are more penalizing than errors in big bounding boxes.

3.1.2 YOLOv2

YOLOv2 [87] is the result of various improvements on the YOLO object detection method. YOLO compared to Fast R-CNN makes a significant number of localization errors. It also has a relatively low recall compared to region proposal-based methods. Thus in YOLOv2, the authors focused mainly on improving the recall and localization while maintaining the classification accuracy.

The first improvement on YOLO was adding batch normalization which led to significant improvements in convergence while eliminating the need for other forms of regularization. By adding batch normalization on all of the convolutional layers in YOLO, the model obtained 2% improvement in mAP, and the dropout layers were removed from the model without any overfitting issue.

YOLOv2 is a fully convolutional model. The fully-connected layers are removed from YOLO and anchor boxes are adopted from Faster R-CNN [88]. Using only convolutional layers, the RPN in Faster R-CNN predicts offsets and confidences for anchor boxes. Since the prediction layer is convolutional, the RPN predicts these offsets at every location in a feature map. Predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn. It also decouples the class prediction mechanism from the spatial location and instead predicts class and objectness for every anchor box. Following YOLO, the objectness prediction still predicts the IOU of the ground truth and the proposed box and the class predictions predict the conditional probability of that class given that there is an object. The network can learn to adjust the boxes appropriately, but the authors used K-means on the ground-truth data with a heuristic that leads to good IOU scores, which is independent of the size of the box. Selecting better priors for the network to start with makes it easier for the network to learn to predict good detections.

YOLO predicts detections on a 14×14 feature map. While this is sufficient for large objects, it may benefit from finer-grained features for localizing smaller objects. Faster R-CNN [88] and SSD [61] both run their proposal networks at various feature maps in the network to get a range of resolutions. YOLOv2 takes a different approach, simply adding a pass-through layer that brings features from an earlier layer at a higher resolution. The pass-through layer concatenates the higher resolution features with the low resolution features by stacking adjacent features into different channels instead of spatial locations, similar to the identity mappings in ResNet [38].

In YOLOv2 a new fully convolutional classification model called Darknet-19 replaced the GoogLeNet based feature extraction part in YOLO. Similar to the VGG [94] models Darknet-19 mostly uses 3×3 filters and doubles the number of channels after every pooling step. Following the work on Network-in-Network [60] the authors used global average-poolings to make predictions as well as 1×1 filters to compress the feature representation between 3×3 convolutions. Darknet-19 has 19 convolutional layers and 5 max-pooling lay-

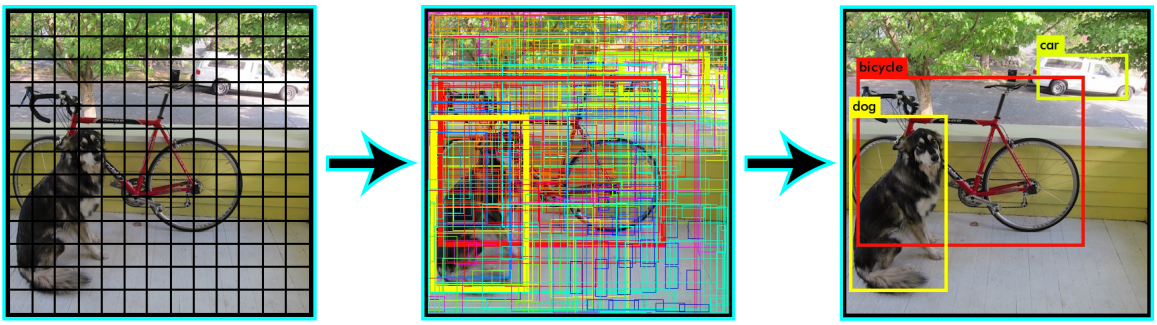


Figure 3.1: YOLOv2 divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities [87] (© 2017 IEEE)

ers. Darknet-19 has 224×224 input size and downsamples the image by the factor of 32, resulting 7×7 feature maps.

YOLOv2, similar to YOLO, divides the image to an $N \times N$ grid and if an object's centre is inside a grid cell, that grid cell is responsible for detecting the object (Figure 3.1). It still suffers from detecting only one object from a group of objects which centres share the same grid cell (and if those objects are in a single category, it will merge their bounding boxes as a single object). The authors suggest selecting an odd number of columns and rows (N) for generating the grid so that there exist a single central cell. Objects, especially large objects, tend to occupy the centre of the image, so it is good to have a single location right at the centre to predict these objects instead of four locations that are all nearby. They selected $N = 13$ and considering the convolutional downsampling of 32 in the model their input size was 416.

They have also come up with a new multi-scale method for training the network. We previously discussed in 2.1.2 that the fully-connected layers have a constant size and can not be resized after training, though, the convolutional layers are constant size filters that could be applied on any input regardless of their size and/or aspect ratios. Since the YOLOv2 model only uses convolutional and pooling layers it can be resized on-the-fly. Instead of fixing the input image size the authors changed the network every few iterations. Every 10 batches the network randomly chooses a new image dimension size. Considering the model downsamples the input by the factor of 32, they pick from the following multiples of 32: $320, 352, ..., 608$. Thus the smallest option is 320×320 and the largest is 608×608.

This regime forces the network to learn to predict well across a variety of input dimensions; which means the same network can predict detections at different resolutions. The network runs faster at the smaller sizes, so YOLOv2 offers an easy trade-off between the speed and accuracy.

3.2 System Overview

Although our system is a general purpose interaction system for un-instrumented humans to interact with any kind of robot (or even device/computer) equipped with a monocular camera, we have chosen UAVs as our target platform to demonstrate this system on.

Our system starts with the UAV on the ground and its motors disabled, waiting for the takeoff command. A laptop computer processes every frame of the video stream received from the UAV over WiFi, searching for hands and faces in the image using our CNN hands-and-face detector.

We adopted the Autonomy_Human[1] [70] pipeline for selecting the interaction partner between all the people in front of the UAV as in [67]. We changed the OpenCV-based face detection part of this package with the results from our CNN face detector. Detections belonging to the interaction partner are fed to our gesture recognizer at frame-rate (30Hz) in the form of Regions of Interest (ROIs) with the class ids and confidences.

The gesture detector considers the proposed ROIs with the confidence score above a tunable threshold and attempts to classify a static gesture from our vocabulary based on the user's *relative* hands and face positions. A behaviour control state-machine module maps the detected gestures to the low-level commands (*e.g.* velocity commands, take a picture, takeoff/land) and transmits them over the same WiFi connection to the UAV for execution.

For the safety purposes and rejecting the rare false-negative detections, the behaviour module only considers the gestures that have been detected in at least three consecutive frames (100ms in 30Hz transmitted video). For being extra safe when flying a UAV close to a person, our gestures are only for controlling the lateral and vertical movements of the UAV: we do not allow the gestures to control the forward motion of the UAV, which might hit the user. Inspired by [67] an independent controller with appearance-based depth estimation tries to keep the UAV in a pre-defined constant safety depth distance of 2.5 to 10m from the user.

The behaviour module also adjusts the tilt angle of the UAV's camera based on the altitude to keep the interaction partner roughly at the centre of the camera's vertical field of view. Assuming an average human height of 1.75m, this keeps the user in the view based on the aforementioned predefined distance from the UAV.

The UAV continually communicates its state and intents to the user with its front-facing coloured-light-based feedback system similar to [68], with a new set of animations. When no user command is detected, the UAV enters an *idle* state, in this state, it will hover stationary in front of the user unless landed.

[1] `https://github.com/AutonomyLab/autonomy_hri/tree/master/autonomy_human`

3.3 Hand and Face Detection

The fact that YOLO reasons globally (*i.e.* uses the features from the entire image to predict each bounding box and also predicts all bounding boxes across all classes for an image simultaneously) makes it ideal for our application. Because it reduces the false-positive detection rate as hands and face locations and mutual appearance are correlated with each other and with arms and torso locations and appearance.

3.3.1 CNN Model

We used the Darknet-19 object classifier model [87] as the feature extractor part of our CNN model. Our goal was repurposing the YOLOv2 model for a robust and real-time hand and face detector that can be used for un-instrumented human-robot interaction. To use YOLOv2 for detecting our two object classes (*i.e.* hands and faces), we needed to make some modification to the original model.

We changed the number of filters in the last convolutional layer of the model before performing the object classification. YOLOv2 predicts $\beta = 5$ boxes on each grid cell. For each box, it calculates $P = 5$ parameters: the x, y coordinates of the centre of the box with respect to the box itself which indicates the centre of the object. Also, w, h dimensions of the box with respect to the whole image since the object dimensions might occupy several cells. As well as a confidence value for that box which shows how confident the RPN is that there is an object in the region. It further predicts a conditional probability value for all the object classes it has been trained on for that region. Here we have $C = 2$ classes for *hand* and *face*. Concretely, the number of filters needs to be set to $(C + P) \times \beta = (2 + 5) \times 5 = 35$ for our two-classes model.

Redmon *et al.* showed since their model uses only convolutional and pooling layers, it can be resized on-the-fly for multi-scale training as well as running at varying sizes. This offers an easy trade-off between speed and accuracy which is crucial in robotics. Following their suggestion, we scaled our model for the second set of training to take 736×736 pixels images as the input, and after down-sampling by the factor of 32 it will divide the image to 23×23 regions to predict the anchor boxes. Table 3.2 shows our final model.

The original YOLOv2 model is designed such that their smallest input option is 320×320 and the largest is 608×608. Our motivation for scaling up the model is that hands and faces are tiny in an image at the distances used for applications like mobile robot HRI. We selected 736×736 because it was the largest size that the model could run at 60fps on our NVIDIA GeForce Titan Xp GPU. However, the model can still be resized to process smaller images at higher frame rates or on smaller GPUs at runtime thanks to multi-scale training. It can be resized to any input size multiple of 32, but following the original paper we suggest selecting an odd number multiplied by 32 in small models, as the objects are

	Type	Filters	Size	Stride	Output
0	Convolutional	32	3×3	1	$736 \times 736 \times 32$
1	MaxPooling		2×2	2	$368 \times 368 \times 32$
2	Convolutional	64	3×3	1	$368 \times 368 \times 64$
3	MaxPooling		2×2	2	$184 \times 184 \times 64$
4	Convolutional	128	3×3	1	$184 \times 184 \times 128$
5	Convolutional	64	1×1	1	$184 \times 184 \times 64$
6	Convolutional	128	3×3	1	$184 \times 184 \times 128$
7	MaxPooling		2×2	2	$92 \times 92 \times 128$
8	Convolutional	256	3×3	1	$92 \times 92 \times 256$
9	Convolutional	128	1×1	1	$92 \times 92 \times 128$
10	Convolutional	256	3×3	1	$92 \times 92 \times 256$
11	MaxPooling		2×2	2	$46 \times 46 \times 256$
12	Convolutional	512	3×3	1	$46 \times 46 \times 512$
13	Convolutional	256	1×1	1	$46 \times 46 \times 256$
14	Convolutional	512	3×3	1	$46 \times 46 \times 512$
15	Convolutional	256	1×1	1	$46 \times 46 \times 256$
16	Convolutional	512	3×3	1	$46 \times 46 \times 512$
17	MaxPooling		2×2	2	$23 \times 23 \times 512$
18	Convolutional	1024	3×3	1	$23 \times 23 \times 1024$
19	Convolutional	512	1×1	1	$23 \times 23 \times 512$
20	Convolutional	1024	3×3	1	$23 \times 23 \times 1024$
21	Convolutional	512	1×1	1	$23 \times 23 \times 512$
22	Convolutional	1024	3×3	1	$23 \times 23 \times 1024$
23	Convolutional	1024	3×3	1	$23 \times 23 \times 1024$
24	Convolutional	1024	3×3	1	$23 \times 23 \times 1024$
25	Copy Layer 16				$46 \times 46 \times 512$
26	Convolutional	64	1×1	1	$46 \times 46 \times 64$
27	Reshape			2	$23 \times 23 \times 256$
28	Concat. 27&24				$23 \times 23 \times 1280$
29	Convolutional	1024	3×3	1	$23 \times 23 \times 1024$
30	Convolutional	35	1×1	1	$23 \times 23 \times 35$
31	Classification				

Table 3.2: The structure of our final model. The two horizontal lines separate the feature extractor part from the region proposal/classification part. Layers 25-27 show how Network-in-Network is implemented in YOLOv2, layer 25 pools the fine-grained features from layer 16, layer 28 concatenates the reshaped fine grained features with features from layer 24.

more likely to be at the centre of the images, and if the centre point is on the border of two cells, the model will have a harder time detecting it.

Note that at both training and testing time, we pad the image dimensions to a square shape (1:1 aspect ratio) prior to resizing them to network's input size. By doing this, we omit any information loss may be caused by cropping and eliminate the futures altering effect caused by wrapping the entire image in a square-shaped image using affine transformation. These phenomena are illustrated in Figure 2.3.

3.3.2 Data Augmentation

The Darknet[2] framework supports a set of pre-defined augmentations. We made some modification in their original augmentation methods and implemented some more to increase the number of training images. Multiple CPU threads are responsible for loading the data from HDD and augment them to form batches for training, this procedure is explained later on in more details.

We randomly crop up to 40% of the image's height and/or width separately, then scale the resulting image again randomly between 25% to 200%. Then we add a random rotational augmentation in the range of $-30°$ to $30°$, and place the resulting image on a gray rectangular canvas with the size of our model input to avoid the hight/width ratio change shown in Figure 2.3. We arbitrarily flip the resulting image horizontally and/or vertically, change the *saturation* and *exposure* of the image up to 50% and the *hue* up to 10% also at random, convert 6% of the images to gray-scale.

3.3.3 Training

For the first round of training on our first set of data, we used the first 23 convolutional and max-pooling layers of the Darknet19_448 pre-trained model on the ImageNet 1000-class dataset as the initial weights for the first 23 layers feature extraction part of our model. This model was trained for 160 epochs using stochastic gradient descent with a starting learning rate of 0.1, polynomial rate decay with a power of 4, weight decay of 0.0005 and momentum of 0.9 using the Darknet neural network framework.

The bounding box regressor and classification layers weights were initialized randomly for the first round of training. We trained the network end-to-end for 80 epochs with a starting learning rate of 10^{-3}, dividing it by 10 at 40th and 60th epochs. We used a weight decay of 0.0005 and momentum of 0.9. We used the same method in YOLOv2 for multi-scale training.

[2]Open source neural networks in C. `http://pjreddie.com/darknet`

In a bootstrap approach, we used the resulting model from the first round of training to annotate the second set of our datasets faster (more on this in Chapter 4), and fine-tuned our new model illustrated in Table 3.2 with a combination of both datasets.

For training the final model, we resized our first model which was based on YOLOv2 to accept 736×736 (23×23 grid) as input instead of 416×416 (13×13 grid). In the same multi-scale training manner we randomly resized the model from 320×320 (10×10 grid: the fastest model) to 960×960 (30×30 grid: the highest accuracy). For the last epoch, we disabled the resizing and kept the input size as 736×736 which was our preferred model that was going to be used in our HRI system. We trained the network for 90 epochs with a starting learning rate of 10^{-3}, dividing it by 10 at 30th, 45th, and 60th epochs. We used a weight decay of 0.0005 and momentum of 0.9.

To increase the training speed and make sure that the GPU does not need to wait for the data augmentation, several CPU threads try to load and augment batches of data from HDD. Whenever the GPU is done with processing a loaded batch, it fetches another loaded batch from the CPU, and the freed CPU thread starts loading and augmenting a new batch of data. We selected 64 images to form a batch, which means the weights will update after the network sees every 64 images to have a smoother training curve and avoid divergence especially at the beginning stages of training. The image data structure in the Darknet framework is an array of float numbers. At the biggest input size, we had 64 images each having 3 channels (RGB) of 960×960 pixels, which is $64 \times 3 \times 960 \times 960 = 176947200$ floating point numbers. We had to divide the loaded data into smaller mini-batches to be able to fit it in the limited GPU memory size, especially when using NVIDIA CudNN[3] to increase the training speed which uses more GPU memory. We divided each batch to 16×4 images mini-batches. Thus the GPU would update the weights after seeing 16 mini-batches and jumped to a new already loaded CPU thread to read the data, leaving the previous thread to load a new batch.

In order to be able to train the model using the Darknet framework, we needed to adapt the bounding box annotations to this framework. For Darknet each bounding box centre position and dimensions need to be normalized to the image size so that all the values will fall in the range of $[0.0, 1.0]$. Thus the final annotation should be such that each image has a single annotation text file, which includes one line per bounding box in the image. Each bounding box has a class ID based on the number of classes we want to detect, in our case, $\in \{0, 1\}$ specifying {Hand, Face}, followed by four numbers in the range of $[0.0, 1.0]$. These numbers indicate the normalized value of the centre coordinate of the bounding box and its height and width based on the image's dimensions respectively.

As mentioned before, YOLO uses the IOU of the ground truth and the proposed box to calculate the confidence of each detection. One way to calculate the *Precision* and *Recall*

[3]NVIDIA cuDNN: GPU Accelerated Deep Learning `https://developer.nvidia.com/cudnn`

for an object detection method is using the same definition used in classification problems: the ratio between true-positive detections and all the detections or all the ground truths indicate the precision and recall respectively. This method which is shown in Equation 3.4 was introduced by Zisserman *et al.* in [27] is the base for almost all the benchmarking tools available in the community, and we used it later on in our evaluation. However results from this method is very dependent on the IOU value for accepting a bounding box as a true detection, which is normally set to 0.5.

$$P = \frac{\#TP}{\#TP + \#FP} \qquad R = \frac{\#TP}{\#TP + \#FN} \tag{3.4}$$

To omit the effect of IOU thresholding for calculating the precision and recall, another definition of these terms could be used: In the field of information retrieval, precision is the fraction of retrieved documents that are relevant to the query and recall is the fraction of the relevant documents that are successfully retrieved. This is shown more formally in Equation 3.5.

$$\begin{aligned} \text{Precision} &= \frac{\text{Relevant Documents} \cap \text{Retrieved Documents}}{\text{Retrieved Documents}} \\ \text{Recall} &= \frac{\text{Relevant Documents} \cap \text{Retrieved Documents}}{\text{Relevant Documents}} \end{aligned} \tag{3.5}$$

To adapt this definition to the object detection problem, we use the detected bounding boxes as our "Retrieved Information" and the ground-truth as the "Relevant Information" as illustrated in 3.2. Using this method we can obtain the precision and recall regardless of the chosen threshold for IOU for true-positive detection.

Ideally while training a CNN model, the error on training data keeps decreasing, but the error on new unseen data might start increasing after some point, especially on small datasets. This phenomenon is called *over-fitting* and is illustrated in Figure 3.3. We used 5% of each dataset as our validation data. In order to avoid the model to over-fit on the training data, we saved the trained weights regularly in the course of training. Then by calculating the recall on the 5% validation data using the second method, we chose the weights resulting in the highest recall as our best result.

We observed over-fitting on our first stage of training, which might be caused by the limited amount of data available for training. On the other hand, the second stage did not over-fit, although the loss value trend was flat for the last six epochs of training (we observed fluctuation after each resizing of the model, but flat overall trend). We can unofficially claim that the diversity of the last set of data, combined with the vast amount of random augmentation, kept the model from over-fitting, and the flat loss value trend shows our model has been trained enough. Also, the fluctuations by each resizing show that the model will perform the best on the size that it has been trained on.

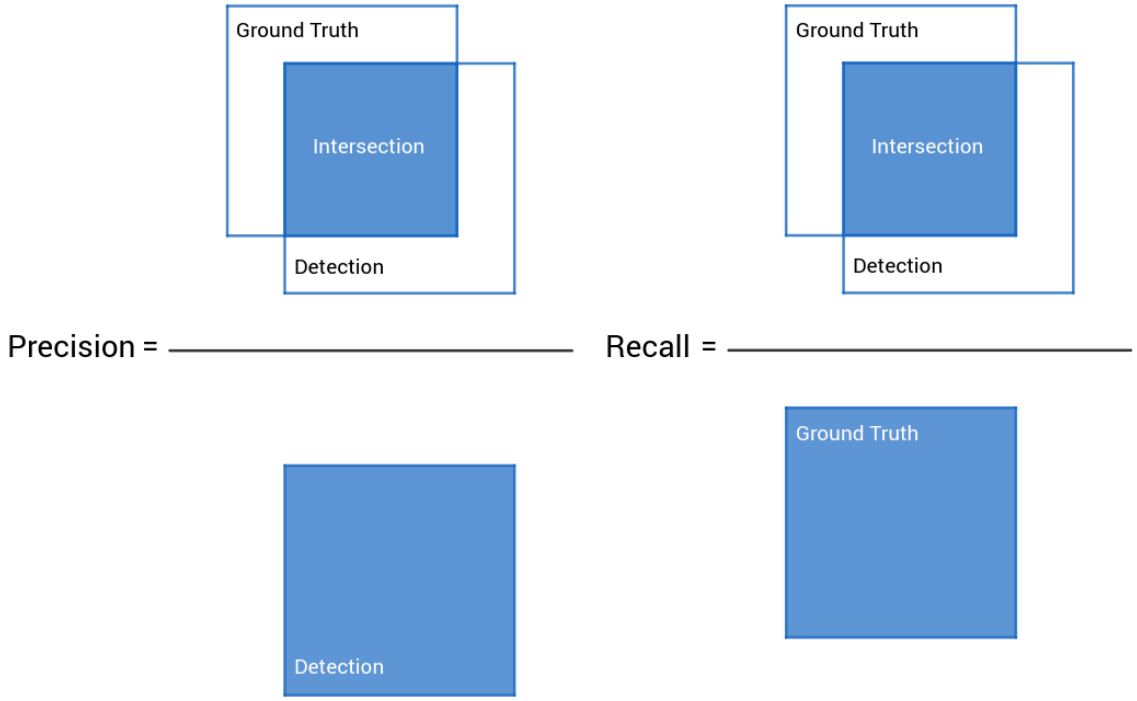


Figure 3.2: In information retrieval, precision is the fraction of retrieved documents that are relevant to the query and recall is the fraction of the relevant documents that are successfully retrieved.

3.4 Gesture Detection

Researchers have been working on defining efficient gestural vocabularies for natural interaction with UAVs for the last decade. In [74, 107, 13] the authors have defined different sets of gestural vocabularies with user studies in the Wizard of Oz manners, yet, none have been implemented successfully on an autonomous flying robot.

The literature on practical and autonomous systems that allow direct, situated, and un-instrumented communication of intents and commands to the UAVs using monocular cameras is very limited. [16] is one of the first successful attempts, however, they need a data gathering and training phase for each particular user that is going to work with the system which is impractical for social robots.

Using the waving-based gestures introduced in [70] and [67] one can detect repetitive motion-based gestures over time in a particular RoI. This method uses the Fast Fourier Transform (FFT) to detect periodic hand motions for gesture detection based on Optical-Flow. These gestures are somewhat slow to pick up. They are vulnerable to losing some frames in the video stream which will cause the Optical-Flow and consecutively the FFT to calculate an incorrect value. Basically, video encoding artifacts or loosing only a couple of frames every second in the video will make the gesture undetectable for the system. This method is also limited to only three gestures: left hand, right hand, and dual hand gestures.

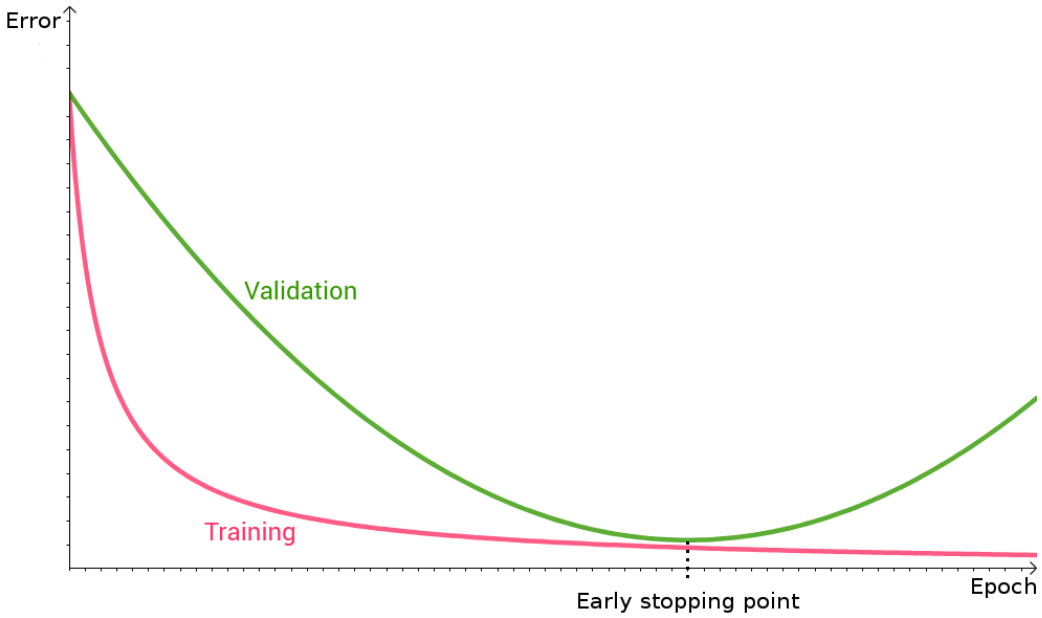


Figure 3.3: Over-fitting causes the model to provide very accurate results on the training data, but increases the error on real-world data unseen by the model.

Relying on the skin detection enabled detection of a user's arms in [101]. The authors used this method and the relative position of the user's arms and torso to generate richer commands for the drone, but skin detection lacks robustness especially under different illuminations and considering that human skin tone varies a lot in the world, which makes using this method not always feasible.

Rautaray *et al.* [85] proposed that gesture recognition techniques consist of three phases: *detection*, *tracking*, and *recognition*. However, our detector is so robust that **we do not need the tracking phase**. We use our per-frame detection to perform static gestures recognition directly from the hands-and-face position in the image frame.

We have come up with a new set of static hand gestures for controlling a drone. Our static gestures vocabulary consists of eight static hand gestures (*i.e.* take-off, land, move up, move down, move left, move right, flip, and take a picture) as illustrated in Figure 3.4.

Our system reliably detects the user's hands and face in each frame of the live video stream from the done. In order to recognize these gestures, we consider two rectangular regions on both sides of the body at the torso level relative to the face bounding box's position and size. These two regions work as home-zones/dead-zones for positional gesture detection as represented in Figure 3.5. We detect a gesture based on each hand's position relative to their corresponding dead-zones and the face bounding box.

Although for directional gestures using only one hand is enough, we hold the other hand as a safety switch inside its dead-zone region for these commands while giving the move direction using the free hand. Using both hands protects the system against any

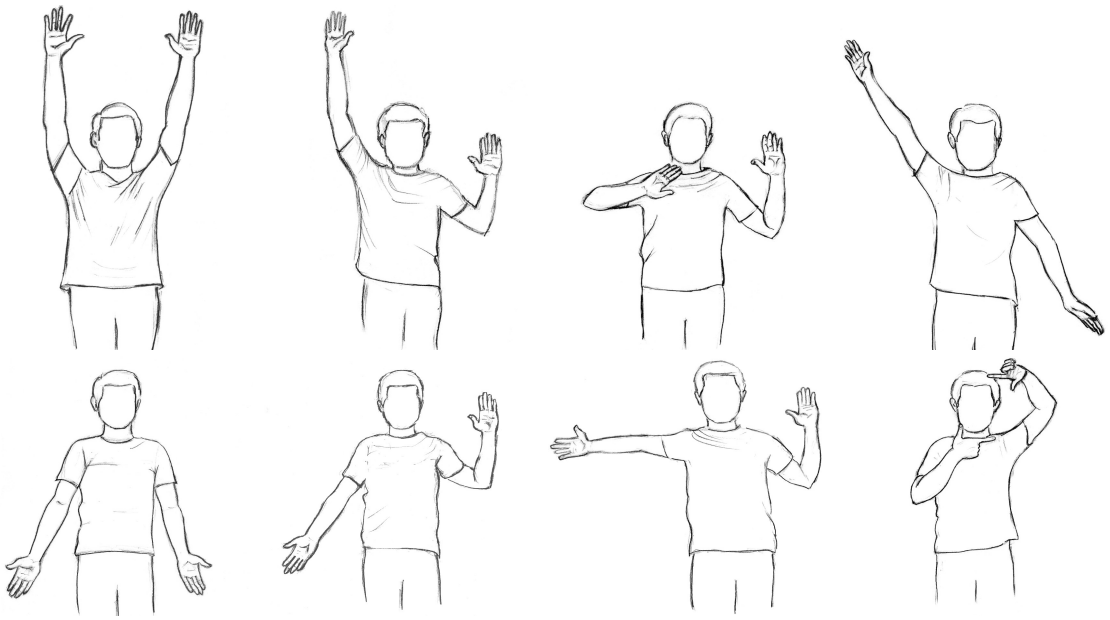


Figure 3.4: Our hand gesture vocabulary for Human-UAV interaction. From left to right: ^ Take-off, ↑ Move up, → Move left, ↻ Flip, v Land, ↓ Move down, ← Move right, and 📷 Take a picture.

false-positive commands detection while the user is in the field of view of the UAV but is not necessarily interacting with it. This is an important safety feature when deliberately maneuvering the UAV very close to a human.

3.5 Hardware

In this work, we designed and implemented a general purpose interaction system for uninstrumented humans and robots. Nevertheless, we only have demonstrated our system working on one platform, a small form factor multi-rotor UAV. However this system is not bound to this particular platform or even only flying robots, it is easily expandable to any camera-equipped robot which has enough computation power for running our CNN model or has the ability to transmit the data in real-time to a more powerful computer to do the computation.

There are many factors to be considered for selecting a UAV suitable for close-range situated interaction with a human. Our main criteria for choosing the platform is safety around the interaction partner. Monajjemi [68] defines a UAV platform that enters the *social space* of a human *safe* if it at least is equipped with physical propeller guards and provides automatic shutdown systems in case of contact between any of its propellers and an object.

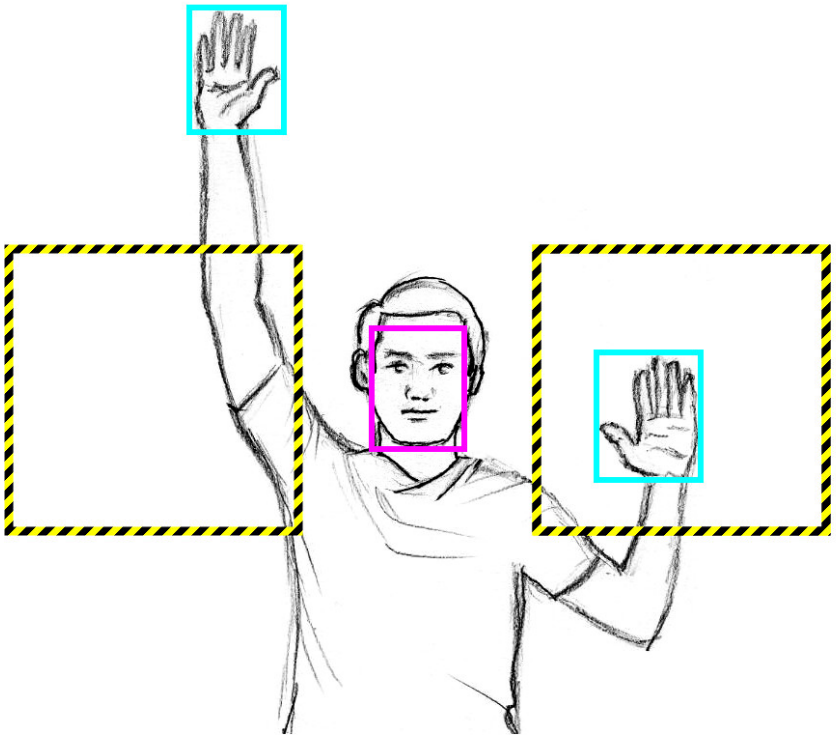


Figure 3.5: Our static gesture detection works based on the position of the interaction partner's hands relative to her face. This shows the dead-zone rectangles which are anchored relative to the detected face position and size.

We chose the Parrot Bebop Drone 2[4] as our UAV platform. Although this UAV provides the required minimum safety measures, its on-board [flight controller] computer is not powerful enough to execute our GPU intensive CNN model. Thus we opted to control the UAV using an off-board ground control station over WiFi, a commodity gaming laptop equipped with GPU[5]. Bebop 2 is a small form factor consumer UAV with an on-board high definition camera and a fisheye lens with the field of view of 180 degrees. The video stream of this camera is digitally stabilized and rectified on-board prior to being transmitted over WiFi with the reduced resolution of $856 \times 480px$ at 30 frames per second. The rectification using their patented 3-axis digital stabilization system simulates a virtual pan/tilt camera with a stabilized gimbal provides remarkable stability by reducing vibration and aliasing. The desired pan and tilt of this camera is also controllable over WiFi. Bebop transmits its telemetry data (i.e. altitude and attitude) over WiFi to the off-board computer at the rate of 5 Hz.

[4] `https://www.parrot.com/ca/drones/parrot-bebop-2`

[5] Dell Alienware 15 with NVIDIA GeForce GTX 1060/6GB

The GPU equipped ground control station processes every single frame of the video using our CNN model and software stack, generates control commands for the UAV, and sends them back for the execution at the same rate as our video input (30 Hz). At any time a safety pilot can take control and override the whole system using a dedicated hardware controller.

Chapter 4

Dataset Engineering

Answering *"How much data do we need to train our CNN model ?"* is not easy, but something that we all agree on is "The more, the better". Statistical learning algorithms can not learn well from a few examples because of the fundamental principles of their design. CNNs are constructed to model the underlying distribution of the dataset they are being trained on. The assumption is that there is a minimum on the error surface and the search for that optimal position proceeds by small gradual steps towards it, the direction of search is usually determined by the first order gradients of the error surface in question. Because of the complexity of approximation function (CNN model) and the error surface, most large scale learning models need tremendous amounts of iterations to successfully traverse the surface and reach the minimum. One might argue that showing the same data over and over to the system, or increasing the learning rate (the gradient descent, descending step) will cause the model to converge to minima faster with a smaller amount of data; In the latter case, having a huge step size for learning might cause the training to diverge and send the training to further away from the desired minima, especially in the early stages of training when the system makes more false-positive predictions and it is more susceptible to noise. Also even if the gradient steps are toward the correct position, having huge step size will cause the result to bounce around the minima when it is close enough. For the former, the dataset itself no matter how big it is is only a fraction of all possible cases. The error surface in question can have many minima, using small amount of data could leave the model in a local minima, or worse, could cause the model to overfit to that small amount of data, which means memorizing the result for those exact inputs but poor performance on a new unseen data.

Our goal is to obtain the underlying pattern, but all we can realistically do with a computer is to sample a tiny fraction of data and try to extract the pattern. As we show more and more data to our model, it learns to ignore the differences and detect the similarities in the data. The word "regression" itself was coined as "regression towards the mean", and *the mean*, in this case, is that underlying trend that we are hoping to find. If the sample size is sufficiently big, the mean will not fluctuate a lot on testing, and the model will be

reliable. Deep learning provides a very flexible tool for discovering those trends, at the cost of using millions of trainable parameters. Simply put, it is a very, very huge regression model. The depth of the network is an assumption about the complexity. Lots of additional data points balance the outliers. In statistical learning, the signal-to-noise ratio is very low, and small datasets can be either very misleading or non-representative concerning the underlying trend.

Another critical factor in designing a dataset is the "variety" of data. Although it is practically impossible to show every single possible representation of our data in the nature to the model, it is essential to cover as much variety as possible, especially covering the corner cases, in our case showing as many different hands and faces to the network, as possible. Each of these differences (including but not limited to: people different skin colours, illumination, background, scale, translation, rotation, postures and gestures, *etc.*) will add another dimension to the dataset complexity. For instance, if we only show the open hand palms to the network, we can not expect it to detect the fist posture. Similarly, if the hands we use for training are always on light backgrounds, the network will learn it as a feature and will have difficulty in detecting the ones on a dark background, or showing only the skin-toned colour hands will make the hands wearing gloves undetectable. Thus we need to cover as much data as we possibly can and leave the rest to the network to learn the connection and interpolate between all the possibilities.

One of the contributions of this work is providing the biggest and only publicly available dataset for hand and face detection. Although there are many available datasets for face detection and recognition, most of them are labelled in an object classification fashion: they mostly contain mugshots without any bounding box around the faces like Faces in the Wild dataset [9]. The Face Detection Data Set and Benchmark (FDDB) [47] is a small portion of the dataset mentioned above with bounding box labelling suitable for face detection with localization in the image frame, which contains the bounding box annotations for 5171 faces in a set of 2845 images taken from the Faces in the Wild dataset. WIDER FACE [119] is a face detection benchmark dataset that offers a lot more bounding box annotated faces with huge variety in terms of scale, pose, occlusion, expression, makeup, and illumination. There are also some publicly available hand detection datasets such as Mittal's hand detection dataset [4], VIVA challenge hand detection dataset [19], and EgoHands [7]. However, to the best of our knowledge, there is no integrated hand and face detection dataset publicly available. The closest to ours is a small portion of Pascal VOC [27] dataset called "Person Layout" that is annotated for body parts and contains around 600 images which are too few for deep learning, also they annotated heads instead of faces thus people might appear from behind.

We started off creating a hand-labelled dataset specific for human-UAV interaction. Then adapting and adding some third party datasets to our needs, we trained our first model for hand and face detection. For increasing the accuracy, and amount of data for

Dataset	Frames	Hands	Faces
Autonomy Hands and Faces	16,883	34,588	18,838
Mittal [4]	5,628	13,050	11,045
Sensors [95]	1,251	2,502	1,251
Pascal VOC [27]	582	1,532	1,055
Helen [126]	2,330	700	2,909
VIVA [19]	5,500	13,229	296
EgoHands [7]	4,800	15,053	1,033
Faces in the Wild [9]	13,391	14,200	21,783
	50,365	**94,854**	**58,210**

Table 4.1: Datasets we have annotated and used for training our models. The first four datasets were used in our first iteration of training, for which we tried to adapt their labeling and add the missing. While the latter four are used for fine-tuning our second model; for these datasets we only used the pictures from the original dataset and labelled them in semi-supervised manner.

training, we used our first model to label another set of data in a semi-supervised manner. Table 4.1 summarizes our contributed datasets.

We set a standard rule for labelling our data: a hand or face that is at least 50% visible (faces up to full profile) is suitable for HRI, is called *detectable* and will be labelled. Thus based on this rule, we define our standard annotation format based on the Pascal VOC [27] *detection* challenge and the ImageNet [90] *localization* and *detection* challenges. Our standard annotation format is a tight axis-aligned bounding box around the visible part of each object of interest in the image, for which at least 50% of the object is visible.

We adopted our first trained model on our first set of training data (*i.e.* Autonomy Hands and Faces, Mittal, Sensors, and Pascal VOC datasets) to annotate more images and create a much larger labelled dataset. For annotating these data, we predicted the hands and faces in the images by bootstrapping our trained model, with the human annotator only adding the missing labels as well as fixing the infrequent errors in the labels generated by the model. We call this method our semi-supervised labelling method which made the labelling procedure an order of magnitude faster.

We divided each of these datasets to $\approx 80\%$ training and $\approx 20\%$ validation data, and used the combination of all these data to train our final model. Except a small part of Autonomy hands and faces dataset from group interaction with a UAV which we used for empirical testing of our trained models, we used all the data explained in this chapter as our training/validation data. The testing is done later on offline benchmark tools for hand detection and face detection separately, and an independent testing dataset with significant illumination, background, and scale changes is gathered as our integrated hands and faces detection test set for HRI which is explained in Section 5.3.

4.1 Autonomy Hands and Faces Dataset

For creating the Autonomy hands and faces detection dataset we selected 69 different duration videos from Human-UAV interactions from our archive. These videos were mostly from close range interaction part of our previous work [67] recorded on-board the UAV, and some from the previous attempt in our lab for creating a Human-UAV interaction dataset called Autonomy-Human.

In these scenarios, a flying UAV hovers in front of a human as its interaction partner. The human shows different motion-based hand gestures such as single or dual-hands waving, swiping horizontally or vertically, and drawing different geometric shapes. They include some static hand gestures as well: pointing to different directions and raising hands are among these static gestures. Other by-standing or moving people also could appear in the field of view of the UAV whom would act as distractors.

47 out of 69 selected videos were from the close range interaction part of [67]. These videos are very brief and only contain hand waving gestures. In these videos eight different users used the system in our experiments, in seven different outdoor environments, in different times of days, in different sunny and cloudy days. We kept two videos from this set as our qualitative test data to empirically evaluate our trained models on, these videos are not from the official experiments for the aforementioned work in which a group of six people are trying to interact with the UAV and take a group selfie using the system[1].

The Autonomy-Human dataset is the outcome of a previous attempt in our lab for generating a human-UAV interaction and gesture detection onboard a UAV. No similar dataset was available. In these data three different users (non of which where presented in our previously mentioned data) show more complicated hand gestures to a flying UAV. These gestures consist of moving gestures such as waving, swiping, and drawing geometric shapes, as well as static gestures such as pointing, raising hands. These data were collected in two different indoor environments and an outdoor environment. In these data the UAV is not always hovering stationary in front of the user: UAV motions occur that cause changes in the size of the user in the images. We selected the data from one of the users in these portion of dataset ($\approx 20\%$) as our validation set, in which the user is the only user wearing sunglasses, the illumination is extreme due to sunlight, and the background is also unseen by the model. Samples from our validation data are shown in Figure 4.1.

Another small part of this dataset is recorded in a tabletop setup using a webcam inside our lab. We hand-labelled all the hands and faces appearing in all the frames considering our criteria for acceptable objects for detection. The resulting dataset consists of $16,883$ selected frames. We hand-labelled $34,588$ *hand*s and $18,838$ *face*s in the selected images.

[1]Result of our final model `https://youtu.be/zGMZHEV429k`

Figure 4.1: The portion of Autonomy H&F dataset used as validation data in our training. In this data user is wearing sunglasses and the illumination is challenging. It contains both static and dynamic gestures and scale changes.

We also have prepared a temporal version of this dataset in which the data are arranged in sequences of images for each video clip. In this version of the dataset, the objects that go from *detectable* state to *undetectable* (based on the standard as mentioned earlier) are being labelled for up to nine more frames even if they are completely occluded by another object. This dataset is useful for benchmarking the tracking algorithms as well as for training a network that can track and predict the position of objects that are not in the field of view anymore.

4.2 Well Known Datasets

To increase the quantity and variety of data, we also made use of some third-party datasets. Not all of the annotations included in those datasets were useful for us; some had only labelled hands or faces that we could modify and adapt for our purpose, others were to be labelled from scratch. In the rest of this chapter, we will introduce these third-party datasets and will address how we managed to use them for the first phase of training or second finetuning phase.

4.2.1 Mittal hands dataset

Mittal *et. al.* provided one of the best publicly available hand detection datasets [4] along with their state-of-the-art complex hand detection pipeline. This dataset is a comprehensive collection of hand images from various public image dataset sources. A total of 13050 hand instances are annotated in this dataset.

In their pipeline hand instances larger than a fixed area for bounding box (1500 sq. pixels) are considered *"big"* enough for detections and are used for evaluation. However in their data collection, no restriction was imposed on the pose or visibility of people, nor was any constraint imposed on the environment. In each image, all the hands that can be perceived clearly by humans are annotated.

The provided annotation files are in standard *".mat"* Matlab format which contain annotations for the four end-points of each appearance of hand bounding boxes. Their annotation bounding boxes are not axis-aligned, they are oriented with respect to the wrist. They also are not bound to the image frame and could be partially outside the image frame.

Figure 4.2: Sample images from our hand-labelled data with bounding box annotations overlaid for initial training phase

Thus for converting these bounding boxes to our standard format, we selected new axis-aligned bounding boxes enclosing the original annotation for each instance and bounded them to the image frame. Figure 4.3 illustrates this difference.

It is evident that the axis-aligned generated bounding boxes are not tightly embracing the objects and a considerable margin including background is being selected as hand bounding box as well. This might cause some of the object detection methods to select a huge amount of background around an object. Moreover using the Pascal VOC standard for object detection criteria ($IOU \geq 0.5$) to validate a detector which selects tight bounding boxes around the objects of interest will result poorly on such a dataset. Our experiments show that by training learning algorithms in which IOU is used in their loss function on such a dataset, although will get an excellent result in test time, the loss value will not decrees as much as expected in training time.

For this dataset, we needed to label the faces as well. We tried the fast and easy to use face detector algorithms to help to label the faces. Among those OpenCV face detector[2]

[2] `https://sourceforge.net/projects/opencvlibrary/`

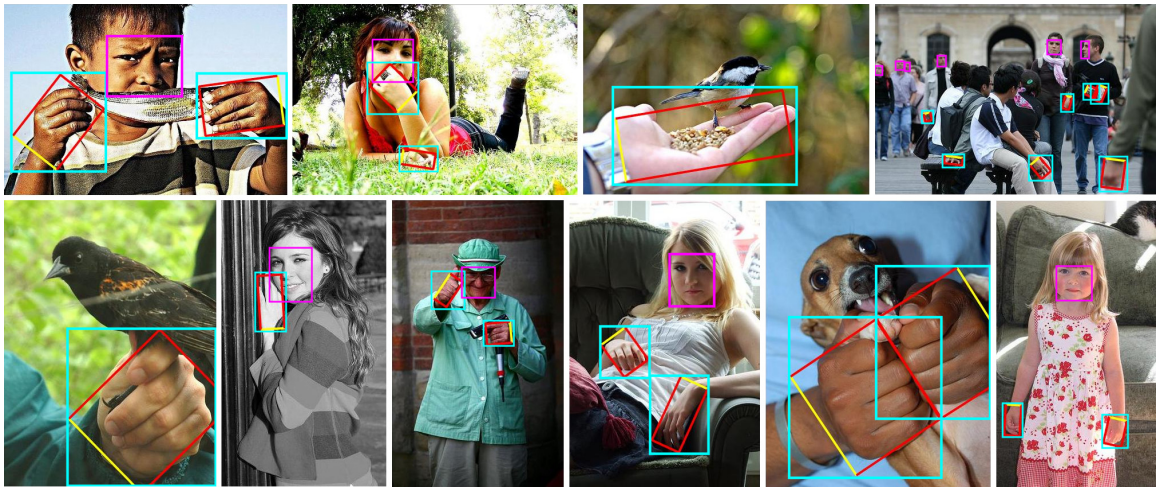


Figure 4.3: Sample images from the Mittal *et al.* hands dataset [4] with bounding box annotations overlaid. Red annotation rectangles are the original dataset annotations, in which rectangle sides are ordered so that the wrist is along the first side marked with a yellow line. Cyan bounding boxes are our converted annotations so that they enclose the original annotation, and magenta bounding boxes are hand annotated faces

and dlib[3] were faster than hand-labelling approach and easy to bootstrap in the labelling application we had written. OpenCV face detector is faster but makes a lot of false-positive detections mainly in the background. dlib tends to make less false-positive detections in the price of detecting only prominent and highly visible faces with very sharp features; it is also slower than OpenCV. Using these methods, we could roughly label 30% of the faces in this dataset and had to refine/label the rest manually.

4.2.2 Sensors dataset

Spruyt *et. al.* also provided a very small hand tracking dataset [95], consisting of 9 sequences of 45 seconds 25fps videos[4]. These 1251 frames of data do not have much variety; all sequences are recorded in simple environments. They all have simple backgrounds, the distance from the subject to the camera is constant, and the hands are big and easy to detect. The subject in all these sequences is a single person and the same person for all the data. The hands are mostly open hands showing the palms. The only challenging part of this dataset is the illumination.

The faces in this dataset were relatively easier to annotate than the previously mentioned datasets. The face position did not change much between the frames, thus annotating the

[3]dlib, Author: King, DE, Access on: `http://dlib.net`

[4]Dataset availeble to download from `https://telin.ugent.be/~vspruyt/sensors/`

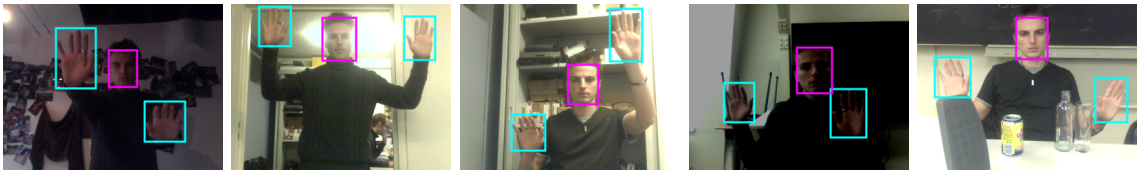


Figure 4.4: Sensors dataset [95] by Spruyt *et. al.* is a very small hand tracking dataset containing easy to detect hands with sequential frames

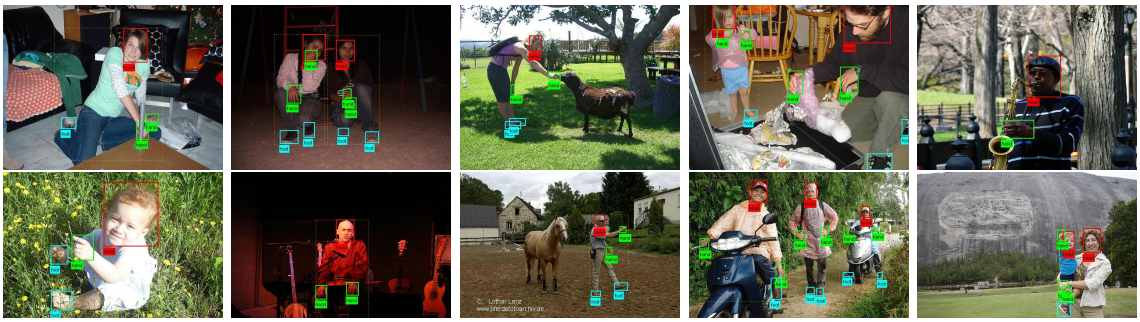


Figure 4.5: A small portion of the Pascal VOC [27] dataset is annotated for "person layout" which we extracted, pruned, and adapted for our work

face's bounding box every other dozen frames and interpolating between those frames did the job. Figure 4.4 shows some examples of this dataset.

4.2.3 Pascal VOC Person Layout

A small portion of the well known Pascal VOC dataset [27], called "Person Layout", is annotated for human body parts detection. In the Pascal VOC dataset, each image has an annotation file giving a bounding box and an object class label for each object in one of the twenty classes present in the image. The Person Layout subsets have been additionally annotated with the parts of the people. These body parts include *Hand*, *Head* and *Foot*. The annotations are tree-based *".xml"* files for which the *person* class for this subset has additional entities for body parts.

We extracted the *hands* and *heads* bounding boxes from the annotation files and converted them to our format. The *hand* bounding boxes were used out of the box. However for the *heads*, we first removed all the images for which the label was not including a face (*e.g.* images of people from behind and the ones without 50% face visibility), then tried to shrink the provided bounding boxes to the face part of the image, not the entire head.

By doing so, we obtained 582 images including $1,532$ hands and $1,055$ faces. Figure 4.5 shows some samples with their annotations. This dataset increased the variety of our data.

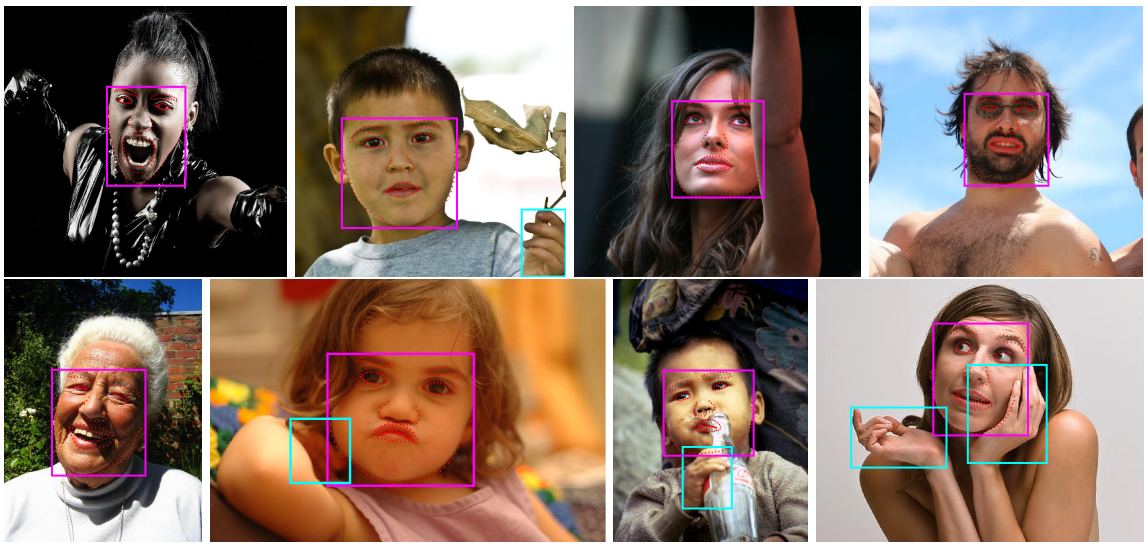


Figure 4.6: The Helen dataset [126] is a facial landmark detection dataset. These images illustrate their landmark detection annotations, as well as our annotation overlaid

4.2.4 Helen dataset

The Helen dataset [126] is a face landmark detection dataset, which consists of 2,330 high resolution very close-range images of faces. It is designed for facial feature localization under a broad range of appearance variation, including pose, lighting, expression, occlusion, and individual differences from Flickr images. The images were hand-annotated using Amazon Mechanical Turk to precisely locate the eyes, nose, mouth, eyebrows, and jawline. If there is more than one face in the image, they have annotated only the dominant face.

We wrapped the landmarks in each image's label with an axis-aligned rectangle to convert their annotations to our standard bounding boxes as shown in Figure 4.6. We then used our semi-supervised labelling method to annotate all the hands and the rest of the missing faces in the images. The final result was 2,330 images including 2,909 faces and 700 hands.

4.2.5 VIVA hand detection benchmark

The VIVA hand detection challenge's dataset [19] consists of 2D bounding boxes around the drivers' and passengers' hands from 54 videos collected in naturalistic driving settings. These data are recorded under large illumination variations, large hand movements, and common occlusions with 7 possible viewpoints.

The 13,229 original bounding box annotations were easy to convert to our standard format. There also were not much face instances in the 5,500 frames of data in this dataset, we used our semi-supervised approach for annotating 296 faces in these images as shown in Figure 4.7.

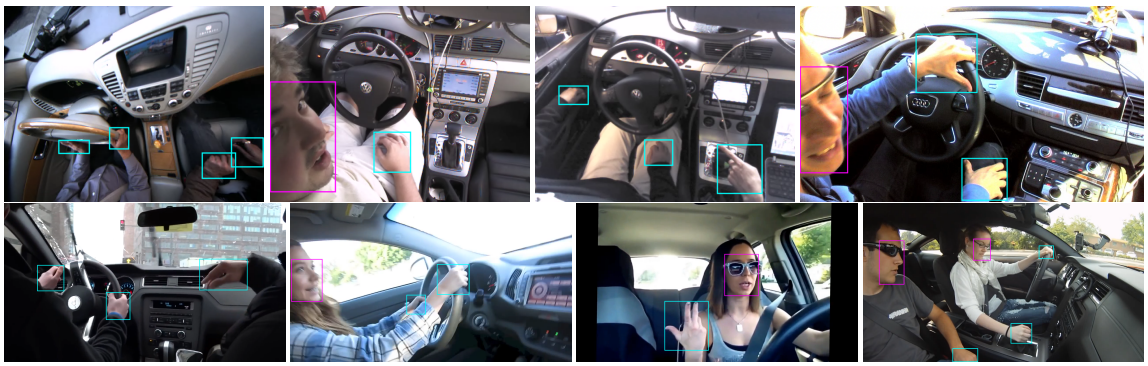


Figure 4.7: The VIVA [19] hands detection dataset is recorded in naturalistic driving settings for detecting drivers' and passengers' hands

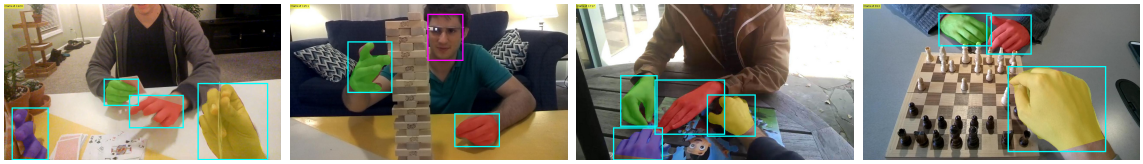


Figure 4.8: The Egohands [7] dataset has pixel level segmentation for each hand. These samples show their annotations and our wrapping bounding boxes, the faces were labeled using our semi-supervised method

4.2.6 EgoHands

The EgoHands dataset [7] contains 48 Google Glass videos of complex, first-person interactions between two people. The primary intention of this dataset is to enable better data-driven approaches for understanding hands in first-person computer vision. These data were collected from different pairs of four participants who sat facing each other while engaged in different activities (*i.e.* playing cards, chess, Jenga, and solving jigsaw puzzle) in three different locations (*i.e.* a table in a conference room, a patio table in an outdoor courtyard, and a coffee table in a home).

The annotations for this dataset is pixel level segmentation of the hands in Matlab, which allows the user to semantically distinguish between the observer's hands and someone else's hands, as well as left and right hands. We used this dataset by converting their annotations to our standard bounding box annotation format and also labelling the occasional appearances of faces in these data as illustrated in Figure 4.8.

Labelling the hands was as easy as wrapping each hand's segmentation in an enclosing axis-aligned rectangular bounding box. We labelled the faces from the subject's occasional glance to her partner in the images using our semi-supervised technique. The result was $4,800$ egocentric frames including $15,053$ hands and $1,033$ faces.

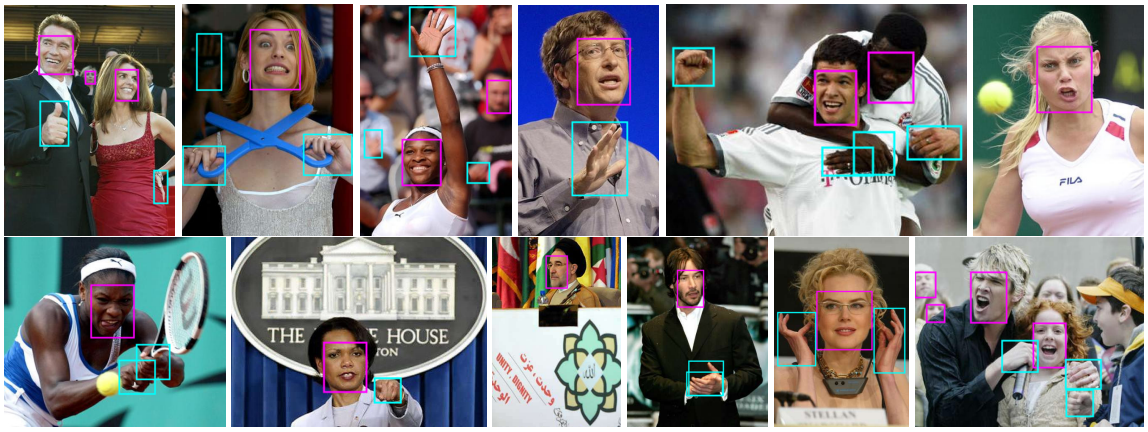


Figure 4.9: Faces in the Wild [9] is a huge collection of images from celebrities and news that was originally labelled to detect who is in the picture. We used or semi-supervised approach to label all the faces and hands in this dataset

4.2.7 Faces in the Wild

The Faces in the Wild dataset [9] consists of $30,281$ faces collected from the News photographs and celebrities. The original annotations of these pictures only indicate who is the most dominant face in the picture, without any information about the position of the faces in the image, or even the number of faces visible in the image.

We used our semi-supervised approach to label all the hands and faces in a fraction of this dataset. Figure 4.9 shows a preview of this dataset with our labels.

4.2.8 WIDER dataset

The WIDER FACE dataset [119] is a face detection benchmark dataset, of which images are selected from the publicly available WIDER dataset. It consists a total of $32,203$ images and labels for $393,703$ faces with a high degree of variability in scale, pose and occlusion. The WIDER FACE dataset is organized based on 61 event classes, some of which were not suitable for our work. For instance, images in the "traffic" class which contain very tiny faces behind the windshield captured by a traffic cam, or faces with very atypical pose or heavy occlusion that causes them to disqualify by our labelling rule. Figure 4.10 illustrates such images and their annotations from the WIDER FACE dataset.

After pruning the unsuitable images from the dataset, we ended up with $12,185$ images. Surprisingly enough we noticed that the annotation quality was not as good as mentioned in the original paper. Filtering their labels by provided information in their annotations (*i.e.* blur, occlusion, and pose information) did not obtain the desired results. Thus we had to cherry pick the suitable ones from all the available bounding boxes. We also had to annotate all the unlabeled faces. Again, we labelled the hands using our semi-supervised method.

Figure 4.10: WIDER FACE [119] is a huge face detection dataset with lots of variety, however many instances from this dataset are not acceptable by our criteria for HRI, thus are not suitable for our work

Due to time constraint annotations for this dataset was not ready for our training our model, but the final selection with annotations is available for future use and is released as a part of our training data.

Chapter 5

System Evaluation

Before this work, there was no benchmarking tool and dataset to evaluate hand and face detection at the same time. We first evaluate our model for hand and face detection separately on available benchmark tools and datasets in the community, then we demonstrate our system's application for close range human-UAV interaction.

5.1 Hand Detection

Mittal *et al.* [4] employed the Pascal VOC evaluation kit to measure the performance of their complex multi-module pipeline for hand detection. This benchmark method is old, and not many works have used it for measuring their performance. The only available data point from this evaluation method is the original work [4].

We used a more recent benchmarking tool provided by the VIVA hand detection challenge [19]. In Section 4.2.5 we discussed that we used the VIVA hand detection training data as a part of our training dataset. Here we use their provided evaluation kit to measure our performance on their test dataset.

The provided kit computes the area under the precision-recall curve (AP) and average recall (AR) rate for evaluation. AR is calculated over 9 evenly sampled points in log space between 10^{-2} and 10^{0} false-positives per image. The original detection evaluation for the challenge was done using the PASCAL overlap requirement of 50%.

The hand detection challenge is evaluated on two levels:

- Simple: Hand instances with the minimum height of 70 pixels, only over the shoulder (back) camera view.
- Hard: Hand instances with the minimum height of 25 pixels, all camera views.

The provided evaluation kit is in Matlab. For this evaluation, we used our system to predict the hands in the provided test data, converted our predictions to the compatible

Rank	Method	Simple AP	Simple AR	Hard AP	Hard AR	fps
1	**Autonomy Hands and Faces** 736×736	**96.1**	**94.9**	**93.7**	**87.8**	60
2	MS-RFCN [55]	95.1	94.5	86.0	83.4	4.65
3	SCUT Augmented FRCNN [44]	93.9	91.5	85.2	77.8	6.32
4	MS-RFCN [55]	94.2	91.1	86.9	77.3	4.65
5	YOLOv2	93.4	91.0	84.0	74.4	123
6	Multi-scale fast RCNN	92.8	82.8	84.7	66.5	0.3
7	MS-FRCNN [55]	90.8	84.1	77.6	65.1	
8	SCUT Augmented FRCNN [44]	89.5	86.0	73.4	64.4	4.9
9	FRCNN [127]	90.7	55.9	86.5	53.3	
10	Modified Faster-RCNN	81.7	59.2	72.8	47.2	
11	ACF_Depth4 [19]	70.1	53.8	60.1	40.4	
12	YOLO	76.4	46.0	69.5	39.1	35
13	**Autonomy Hands and Faces** 160×160	74.6	53.6	62.1	38.6	**192**
14	CNN with Spatial Region Sampling [7]	66.8	48.1	57.8	36.6	0.783
15	ACF [75]	62.4	36.9	52.3	27.5	11.6

Table 5.1: The VIVA hand detection challenge results AP/AR for the Simple and Hard evaluation settings. For both of the evaluation metrics, higher is better. The results are sorted by AR on the Hard setting

format and fed it to their evaluation code. We compared our method's results with the provided challenge's results from different methods for hand detection[1] in Table 5.1.

Our method performed significantly better than others in case of both accuracy and recall despite the fact that the model is trained to detect both faces and hands in comparison to all the other methods which are only hand detectors. We carefully engineered the network architecture to run exactly 60 frames per second with the highest possible accuracy because most of the widely used cameras in robotics fields are between 27 to 60 fps. We also trained the Original YOLOv2 model out of the box for comparison and added the results to the Table 5.1 which occupies the 5th rank. The evaluation kit provides the detection results for seven of the methods mentioned in the Table (*i.e.* methods number: 3, 7, 8, 9, 11, 12, 14). Using the provided detection data in the evaluation kit and the results from our model and trained YOLOv2 model we drew the precision-over-recall chart, as well as the true-positive detection rate for both Simple and Hard evaluation set shown in Figure 5.1.

Although the Pascal VOC evaluation standard for object detection defines 50% overlap between the ground-truth and the detected bounding box indicates a *correct* detection, to further push the boundaries and for better illustration of the differences in the methods, we made the evaluation criteria even more challenging and evaluated these methods in a new setting with 75% overlap as the *correct* detection. The results are shown in Table 5.2.

[1] `http://cvrr.ucsd.edu/vivachallenge/index.php/hands/hand-detection/`

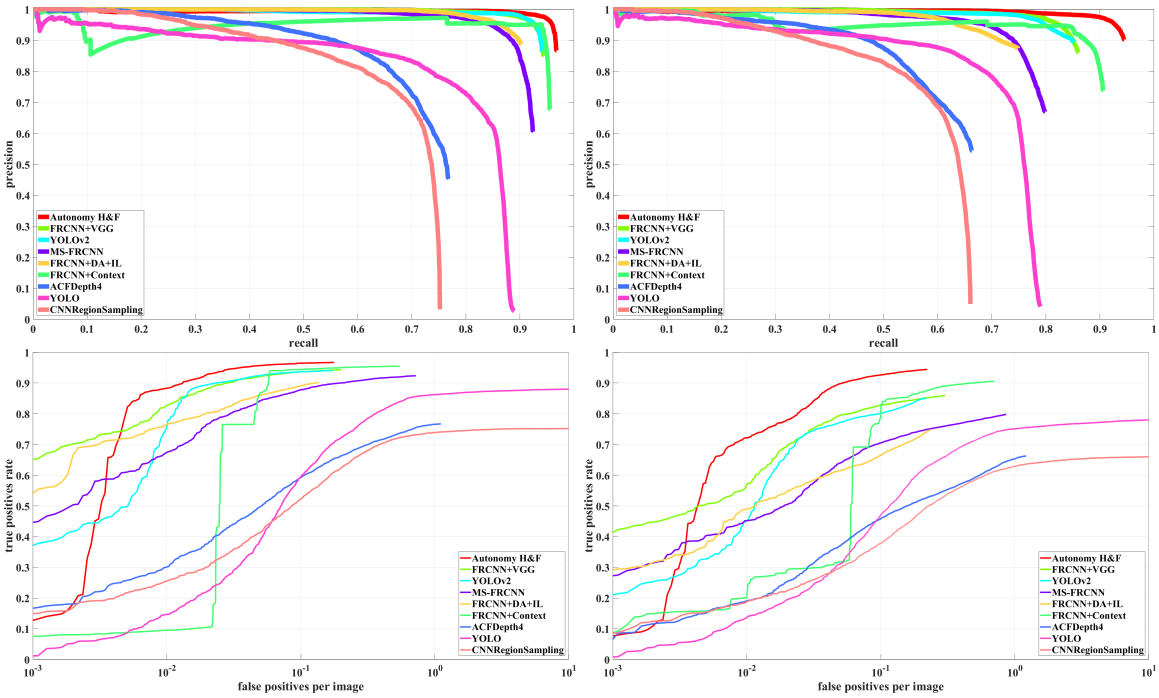


Figure 5.1: Hand detection evaluation results for 9 methods in Table 5.1

		Simple		Hard		
Rank	Method	AP	AR	AP	AR	fps
1	Autonomy Hands and Faces 736 × 736	53.4	26.7	45.1	16.4	60
2	YOLOv2	44.5	26.6	33.6	14.2	123
3	SCUT Augmented FRCNN [44]	42.7	23.0	34.0	14.1	6.32
4	MS-FRCNN [55]	34.7	19.9	26.5	11.9	
5	FRCNN [127]	33.1	1.0	34.2	10.5	
6	SCUT Augmented FRCNN [44]	34.0	10.4	24.8	6.2	4.9
7	YOLO	10.2	6.2	9.9	4.3	35
8	CNN with Spatial Region Sampling [7]	4.4	3.9	4.2	2.9	0.783
9	ACF_Depth4 [19]	5.2	3.3	4.5	1.9	

Table 5.2: Hands detection benchmark in a more challenging setting, we changed the correct detection criteria from 50% overlap to 75% overlap threshold

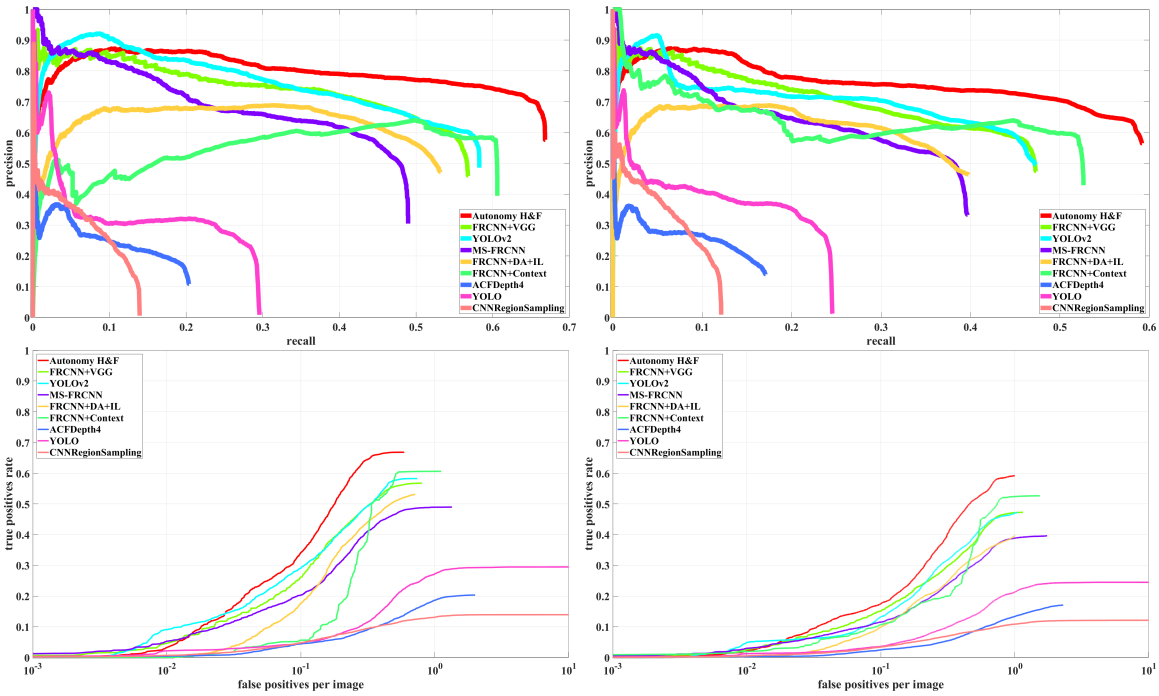


Figure 5.2: Hand detection evaluation results for 9 methods in Table 5.1 with 75% overlap as a correct detection threshold

The precision-over-recall chart for 75% overlap threshold in Figure 5.2 confirms our argument in Chapter 2 that region-based methods do a better job in localizing the objects in the image and select a better bounding box around the ROI. Also as all the other methods were only trained to detect the hands in the designated dataset for the challenge, they can more precisely localize the hands if they are successfully detected. However, our hand detection which is trained on more general hand postures and gestures in order not to be bound to form, size, or point of view, still does a better job overall.

5.2 Face Detection

For evaluating our face detection, we used the WIDER [119] face detection benchmark tool. WIDER too adopts the same evaluation metric employed in the Pascal VOC dataset: If the ratio of the intersection of a detected region with an annotated face region is greater than 0.5, a score of 1 is assigned to the detected region, and 0 otherwise.

This benchmark tool provides a huge training dataset as well; therefore the authors retrained some baseline models on their dataset to show the effectiveness of their training data. The provided results in the paper [119] consists of two parts: (i) evaluating the available baseline methods on the validation and test datasets (ii) retraining Faceness [118] model which showed to be the state-of-the-art in face detection as well as some other face detection methods to evaluate the effect of their training data.

Our model is not trained on the WIDER training data, which makes the first approach of evaluation a fair comparison for this work. We compared our model to state-of-the-art methods without fine-tuning on the WIDER training data as well as the new methods trained on the WIDER training data. Figure 5.3 shows the precision-over-recall plots for this benchmark. The left column shows the evaluation among the state-of-the-art methods without fine-tuning on the WIDER training data, and the right column shows our method among those trained on the WIDER training data. As WIDER does not provide the test data ground-truth we had to evaluate our model on the validation set (10% of the dataset). The evaluation data is divided to three different categories, *easy, medium,* and *hard* for which the results are shown from top to bottom in Figure 5.3. Table 5.3 shows the quantitative average precisions for these methods.

As we mentioned in Section 4.2.8, not all of the WIDER face data is suitable for our work (*i.e.* direct and close-range human-robot interaction); We believe that for close-range human-robot interaction the robot should be within the *social space* ($\approx 3-4m$) of the user [35]. Detecting Very small faces behind the cars' windshield or a tiny faces in the crowd are not useful examples for HRI, in fact, detecting such faces might even increase the vulnerability of the system. As mentioned in Chapter 4 we carefully designed our training dataset to be as close as possible to HRI applications while keeping the diversity in the data.

Considering these facts, plus our model detects hands as well, and also is the fastest method available between all these methods, we still beat all the baseline methods which are not trained on the WIDER dataset by a considerable margin (left column in Figure 5.3). Our method also competes with carefully designed face detectors for WIDER in the right column. We believe that adding a set of carefully selected data from the WIDER training data to our training dataset can increase the face detection accuracy. Though we do not want to compete with the methods trained on the WIDER dataset because we do not want our model to be able to detect the faces that are not directly interacting with the robot.

5.3 End-to-end Close Range Human-UAV Interaction

For all the experiments we used the platform described in Section 3.5. Except for the LED animation generator which runs on an Intel Edison embedded computer on-board the UAV, all the software stack runs on a commodity gaming laptop equipped with an NVidia GeForce GTX 1060/6GB. For the data-intensive communication with the UAV over WiFi, we used a long-range IEEE 802.11ac external network card with a high gain antenna. We extensively used Robot Operating System (ROS) [84] to integrate different software and hardware components of this system.

All the source code for various components of this system (including the ROS wrapper for Darkent, our trained hands-and-face detection model, ROS implementation of static

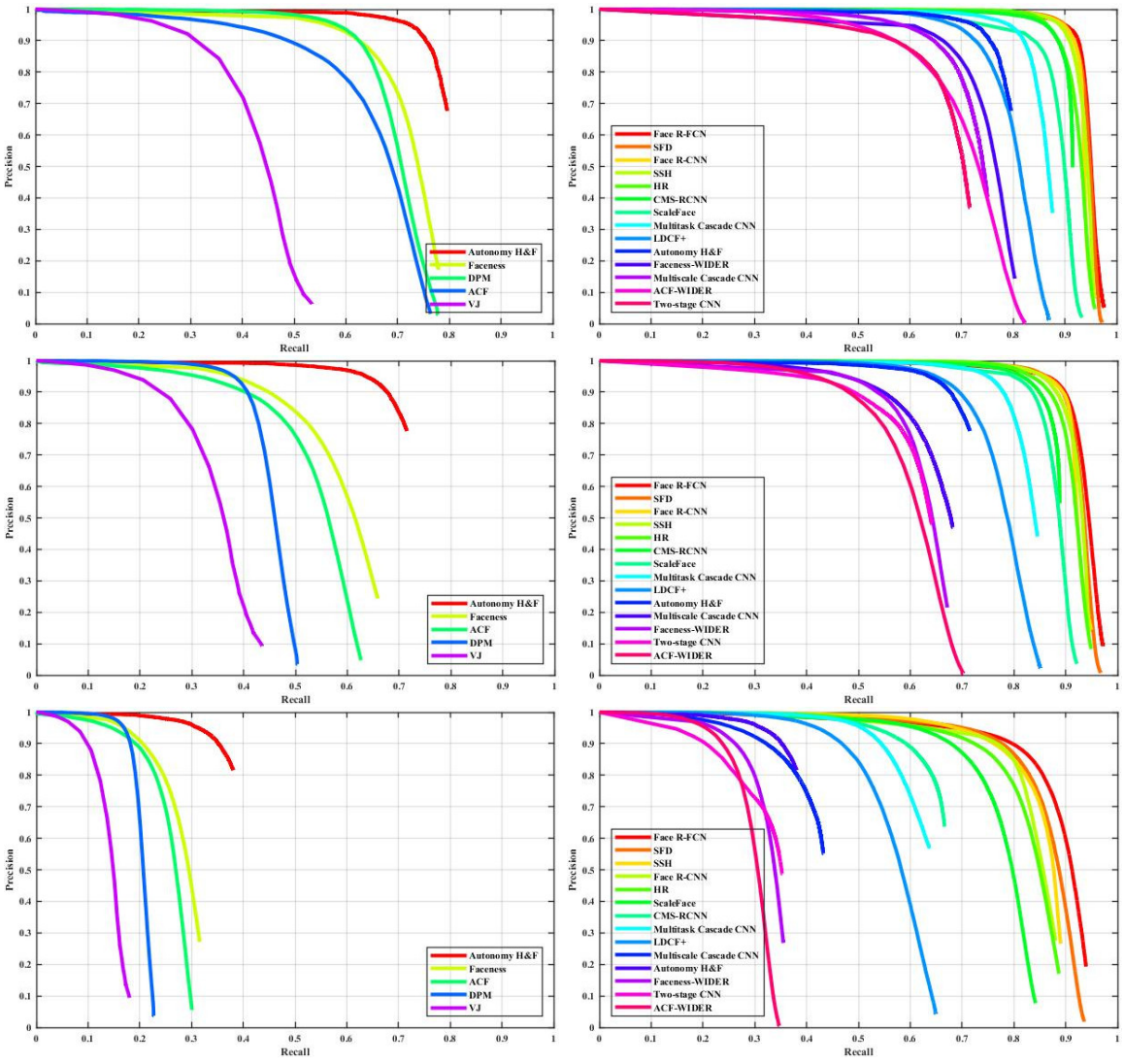


Figure 5.3: Face detection evaluation on the WIDER benchmark tool. The left column shows the comparison of our method among the baseline methods without training on the WIDER training data in three settings (*i.e.* easy, medium, and hard). The right column shows our model is competitive among the models that are designed and trained on the WIDER dataset.

Method	Easy AP	Medium AP	Hard AP
Autonomy H&F	**0.779**	**0.700**	**0.370**
Faceness [118]	0.704	0.573	0.273
DPM [28]	0.690	0.448	0.201
ACF [22]	0.642	0.526	0.252
Viola Jones [110]	0.412	0.333	0.137
Face R-FCN [114]	0.943	0.931	0.876
SFD [125]	0.935	0.921	0.858
Face R-CNN [112]	0.932	0.916	0.827
SSH [72]	0.927	0.915	0.844
HR [42]	0.923	0.910	0.819
CMS-RCNN [128]	0.902	0.874	0.643
ScaleFace [120]	0.876	0.866	0.764
Multitask Cascade CNN [124]	0.851	0.820	0.607
LDCF+ [76]	0.797	0.772	0.564
Faceness-WIDER [118]	0.716	0.604	0.315
Multiscale Cascade CNN [119]	0.711	0.636	0.400
ACF-WIDER [117]	0.695	0.588	0.290
Two-stage CNN [119]	0.657	0.598	0.304

Table 5.3: Average precision of different methods on the WIDER benchmark. The horizontal line separates state-of-the-art methods which are not trained on the WIDER training data and were tested on this dataset, and the methods which are designed and trained on the WIDER training data

gesture detection module, and the LEDs animation generator) are available online[2] and as open-source contributions to the community.

In order to test the effective range of our hands-and-face detector as well as our gesture detection method with a flying UAV, we performed a set of experiments in which the distance between the UAV and the user increased linearly from 2.5 to 10m. We measured the hand and face detection rates as well as the gesture detection accuracy. These experiments were performed in four different locations (two indoors and two outdoors) with different lighting and against different backgrounds. Figure 5.4 illustrates the scale, illumination, and background changes in our evaluations.

5.3.1 Hands and Face Detection

For evaluating the hand and face detection accuracy, we passed every single frame of the recorded videos through our CNN model and counted the number of correctly detected hands and faces. Table 5.4 summarizes the detection accuracy over varying distances.

Results show a noticeable decrease in detection accuracy as the distance between the user and the camera increases. However, the system works with $\simeq 95\%$ accuracy in distances

[2] `http://autonomy.cs.sfu.ca/hands_and_faces/`

Figure 5.4: Scale, illumination, and background changes in our evaluation

		Hands		Faces	
Distance	Frames	#	%	#	%
$2.5m$	16,836	33,431	99.28	16,761	99.55
$5.0m$	16,592	31,367	94.52	14,827	89.36
$7.5m$	16,615	20,318	61.14	9,281	55.86
$10.0m$	16,562	8,586	25.92	6,388	38.57

Table 5.4: Results of evaluating the accuracy of individual hand and face detection at different ranges. The cumulative number of frames and the number and percentage of correctly detected hands and faces are shown

$\leq 5m$. We believe that if a human intends to have a *close-range* interaction with a robot, the robot should be in her social space. Hall [35] defines $\approx 3-4m$ as the social space of a person. We also tested our system at the 20m range with hands and faces detected correctly in less than 1% of the frames; thus we were not able to detect any gestures at that distance.

Our system is very robust to false-positives. The false-positive detection rate over all experiments was less than 0.06%, mostly falsely detecting motion blurred very close flowers or grass as hands or faces during takeoff/landing.

Although we could increase the network input size for more accurate detection, which would divide the image into smaller cells grid and certainly would increase the detection rate furthermore, we did not want to miss the frame rate that the UAV's camera provided for us, and our system ought to work in real-time.

5.3.2 Gesture Detection

To evaluate our gesture detection system, we conducted a series of experiments. All the experiments were repeated with two subjects, one male and one female. Each user performed all the 8 gestures 3 times per location per distance, resulting in 96 trials for each gesture. We considered a trial successful if the system could detect the gesture in less than 1 second or 30 frames. The result of the evaluation is provided in Table 5.5.

Note that we had 3 frames threshold for safety purposes. A gesture would trigger only if the hands and faces were detected for 3 consecutive frames in the same relative regions.

Distance	^	v	↑	↓	←	→	📷	🔄	Total
$2.5m$	100	100	100	100	100	100	100	100	**100**
$5.0m$	88	100	96	100	100	100	75	92	**94**
$7.5m$	54	92	71	75	88	42	25	71	**65**
$10.0m$	38	63	25	33	46	4	0	42	**31**

Table 5.5: The fraction of gestures correctly identified at different ranges reported as a percentage over 96 repeats for each. The eight gestures are: Take-off, Land, Move up, Move down, Move right, Move left, Take a picture, and Flip

Our results show that the 'take a picture' gesture was the hardest to detect for our system (*i.e.* it was the first to degrade with distance) because the hands could cover some portion of the face. Also as we discussed before YOLO method has a flaw for objects very close to each other; thus hands and faces near each other will appear in the same grid cell in the image, therefore, are indistinguishable at long range.

The 'landing' gesture was the easiest to recognize. We suspect that this is because this gesture is close to normal standing posture, which was very common in the training data, and we argued that the YOLO method looks at the whole image and reasons globally. Thus it might be vulnerable to false-positives in practice.

Based on this data and our observations, and consistent with our previous experience, the best distance for situated interaction between an un-instrumented human and a small form factor UAV is between 2.5 to 5m. At these ranges, the UAV can see the interaction partner well enough to detect her hands and face, and also the human is close enough to see the UAV's feedback (in our case the colour-based LED feedback). At distances further than 5m, the humans' ability to detect the UAV's movements and intents properly are degraded, and control becomes difficult. Also flying a UAV closer than 2.5m to a human will violate the safety criteria.

The data from our experiments form another contribution to the community as the first integrated hand and face detection benchmark.

Chapter 6

Conclusion and Future Work

6.1 Hand and Face detection

In this thesis, we provided surveys on state-of-the-art methods for monocular image CNN based object detection, and hand- and face-detectors. We proposed and demonstrated a CNN based vision system that can accurately detect hands and faces in images at 60 frames-per-second on high-end GPUs and 30 frames-per-second on commodity GPUs. Our fully-convolutional model is end-to-end resizable for speed/accuracy trade-off.

We trained our model on a novel dataset we gathered for human hand and face detection and evaluated this model using previously published third-party benchmarking tools. The combination of our model and dataset hand detector obtained state-of-the-art results on the VIVA [19] hand detection challenge benchmark dataset both in accuracy and speed. Although our face detector obtained similar results on the WIDER [119] face detection benchmark to those methods that were not trained directly on the WIDER dataset, it still provides competitive result between other methods that are designed specifically for the WIDER benchmark and have been trained on it which obtain average higher precision in benchmark but none of them are nearly as fast as our method. It worth to mention that not all the face instances in the WIDER dataset is useful for HRI systems and training on those data will increase the false positive probability concerning interaction partner selection for HRI purposes.

6.2 Human-Robot Interaction

We also surveyed related work on HRI systems. We provided a broad survey on un-instrumented human-robot interaction systems, and the modalities used for this purpose. Hand gesture communication has initially been suggested by Karam [50] as the most common communication gestures amongst humans, and hence the most natural gestures for Human-Computer interaction, Monajjemi [68] proposed hands-and-gaze communication for un-instrumented human-UAV interaction. Thus informally, we found hand-and-face ges-

ture communication intuitive, easy to use and learn by users, and easily expandable, both in terms of variety of commands and being universally used as standard robots gestural language.

Our survey suggested that the need for these kinds of systems is increasing and researchers are turning their attention towards hand gestures for HRI. We described work on gestural vocabularies for human-robot interaction especially on human-UAV interactions as we chose UAVs as our target platform to demonstrate the usability of the system. Many researchers have worked on defining human-UAV interaction hand gesture vocabularies. In fact, designing such a vocabulary is more explored in the literature than actually using them in practical systems. Our method can be used to test previously designed vocabularies practically.

We designed a novel, very simple but effective method for static hand gesture detection based on the relative positions of the interaction partner's hands and face. We used this method to design a small vocabulary of static hand gestures for an un-instrumented human, to be able to interact with a UAV equipped only with a monocular camera with only process off-board the UAV.

6.3 End-to-End Human Robot Interaction

We used our hands-and-face detector and the gesture detector to implement an end-to-end human-robot interaction system. In our system, a UAV transfers its camera feed over WiFi to an off-board ground control station equipped with a commodity GPU. The off-board computer uses the CNN model for finding the location of all hands and faces in each frame of the received video. Then the gesture detection module translates the hands and face relative position to a command from our vocabulary, sends the command to a behavior control module which interprets the command to low-level UAV control commands while a separate controller keeps the UAV in a fixed distance from the user. The generated commands are sent to the UAV via same WiFi connection for execution.

The system is sufficiently robust to scale, illumination and background changes to be practical for real application. We showed that an un-instrumented user could control a co-located UAV to take-off, translate, flip, take pictures and land successfully without any need for physical interaction with the robot or other equipment such as a hardware controller.

We have shown our system to be competitive in offline benchmarks and demonstrated it in a real robot system in different environments under challenging illuminations and backgrounds. Both our hands-and-face detector and gesture detector give qualitatively and quantitatively good results in videos at 30 FPS on a commodity GPU. Per-frame detection is so reliable that we do not need tracking in our application. Our system is easily expandable to detect other objects as well as human hands and faces by adding the extra data to the dataset. The whole model is also scalable for accuracy/speed trade-off.

6.4 Open-Source Contributions

In addition to the HRI contribution, we presented the largest integrated hands-and-faces detection dataset with very precisely drawn bounding box ground-truth. The data are gathered from different sources and have a vast variety in distance, illumination, background, pose, resolution, *etc.*. To the best of our knowledge this dataset is the only available work that combines these features and is thus a contribution to help solve challenging HRI tasks.

We also released the data from system evaluation experiments in this work as a test dataset for benchmarking HRI and tracking systems. All our thus labeled data is freely available from the project webpage:
`http://autonomylab.org/hands_and_faces`

We also released all the project source code, documentation, and data including: the source code for ROS wrappers for YOLOv2 generic object detector, gesture detection module, UAV behaviour controller module, new version of *Autonomy_Human* package for interaction partner selection using CNN model as face detector, as well as trained models for hands-and-face detection as a contribution to the HRI community.

A video using our system in the loop for controlling a flying UAV is available on the project's webpage.

6.5 Future Work

For all our human-UAV interaction experiments, a proper user-study with naive participants would be required to evaluate our practical system.

One potentially valuable expansion to this work is changing the CNN model so that it can associate hands with corresponding faces in the images that contain multiple people. Also to expand the model to detect more body parts and even fit the whole skeleton to a person.

Another potential expansion is using recurrent models to track hands and faces in video frames over time.

Most CNN generic object detectors consist of very deep models in order to increase the number of extracted features so that they can detect more categories of objects. However in our line of work, having a very fast and robust model that can only detect a few number of objects accurately but uses less processing power is more desirable. Thus optimizing the network size using CNN pruning methods so that it can run real-time on smaller GPUs such as cellphones or embedded GPUs[1] that could be carried on-board small form-factor UAVs and robots, or more desirably can run near real-time on CPU is also interesting.

[1]NVIDIA Tegra processors and on Jetson development boards:
`https://developer.nvidia.com/embedded-computing`
`http://www.nvidia.ca/object/tegra-superchip.html`

Also designing a standard and universal hand gestures vocabulary for interaction with robots will have a considerable impact on the community, and it is needed sooner or later.

Overall, there are many opportunities for research and commercial development of HRI systems. Robots are coming into our daily lives, and we need to interact with them. We hope the work described in this thesis is a useful contribution in the area.

Bibliography

[1] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face Recognition with Local Binary Patterns. In *European Conference on Computer Vision (ECCV)*, pages 469–481, Berlin, Heidelberg, 2004. Springer.

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the Objectness of Image Windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, Nov 2012.

[3] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale Combinatorial Grouping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 328–335. IEEE, June 2014.

[4] Andrew Zisserman Arpit Mittal and Philip Torr. Hand detection using multiple proposals. In *British Machine Vision Conference (BMVC)*, pages 75.1–75.11. BMVA Press, 2011.

[5] Reza Azad, Babak Azad, Nabil Belhaj Khalifa, and Shahram Jamali. Real-Time Human-Computer Interaction Based on Face and Hand Gesture Recognition. *arXiv preprint (CoRR)*, abs/1408.1549, 2014.

[6] Haris Baltzakis, Maria Pateraki, and Panos Trahanias. Visual tracking of hands, faces and facial features of multiple persons. *Machine Vision and Applications*, 23(6):1141–1157, Nov 2012.

[7] S. Bambach, S. Lee, D. J. Crandall, and C. Yu. Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1949–1957. IEEE, Dec 2015.

[8] P. T. Bao, N. T. Binh, and T. D. Khoa. A New Approach to Hand Tracking and Gesture Recognition by a New Feature Type and HMM. In *International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 4, pages 3–6. IEEE, Aug 2009.

[9] Tamara L. Berg, Alexander C. Berg, Jaety Edwards, and D. A. Forsyth. Who's in the Picture? In *International Conference on Neural Information Processing Systems (NIPS)*, NIPS'04, pages 137–144, Cambridge, MA, USA, 2004. MIT Press.

[10] Ludovic Brethes, Paulo Menezes, Frédéric Lerasle, and J Hayet. Face tracking and hand gesture recognition for human-robot interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1901–1906 Vol.2. IEEE, April 2004.

[11] Jake Bruce. Finding Tiny People: Long-Range Outdoor Sensing for Establishing Joint Attention in Human-Robot Interaction. Master's thesis, Simon Fraser University, December 2015.

[12] J. Carreira and C. Sminchisescu. CPMC: Automatic Object Segmentation Using Constrained Parametric Min-Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328, July 2012.

[13] Jessica R. Cauchard, Jane L. E, Kevin Y. Zhai, and James A. Landay. Drone & Me: An Exploration into Natural Human-drone Interaction. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, UbiComp '15, pages 361–365, New York, NY, USA, 2015. ACM.

[14] Dan C Cireşan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 411–418, Berlin, Heidelberg, 2013. Springer.

[15] Tiago Ogioni Costalonga, Lucas Mendes Ávila, Luís Muniz, and Alexandre Santos Brandão. Gesture-Based Controllers to Guide a Quadrotor Using Kinect Sensor. In *Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol (SBR LARS Robocontrol)*, pages 109–112. IEEE, Oct 2014.

[16] G. Costante, E. Bellocchio, P. Valigi, and E. Ricci. Personalizing vision-based gestural interfaces for HRI with UAVs: a transfer learning approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3319–3326. IEEE, Sept 2014.

[17] Alex Couture-Beil, Richard Vaughan, and Greg Mori. Selecting and Commanding Individual Robots in a Vision-Based Multi-Robot System. In *2010 Canadian Conference on Computer and Robot Vision (CRV)*, May 2010.

[18] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 379–387. Curran Associates, Inc., 2016.

[19] N. Das, E. Ohn-Bar, and M. M. Trivedi. On Performance Evaluation of Driver Hand Detection Algorithms: Challenges, Dataset, and Metrics. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2953–2958. IEEE, Sept 2015.

[20] Tomás Mantecón del Valle, Carlos Roberto del Blanco Adán, Fernando Jaureguizar Núñez, and Narciso García Santos. New generation of human machine interfaces for controlling UAV through depth based gesture recognition. In *SPIE Defense, Security and Sensing Conference*, volume 9084. International Society for Optics and Photonics, May 2014.

[21] Piotr Dollár, Serge J Belongie, and Pietro Perona. The Fastest Pedestrian Detector in the West. In *British Machine Vision Conference (BMVC)*, pages 68.1–68.11. BMVA Press, 2010. doi:10.5244/C.24.68.

[22] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral Channel Features. In *British Machine Vision Conference (BMVC)*, pages 91.1–91.11. BMVA Press, 2009. doi:10.5244/C.23.91.

[23] Michael Donoser and Horst Bischof. Real time appearance based hand tracking. In *IEEE International Conference on Pattern Recognition (ICPR)*, pages 1–4. IEEE, Dec 2008.

[24] George ElKoura and Karan Singh. Handrix: Animating the Human Hand. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, SCA '03, pages 110–119, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[25] Ian Endres and Derek Hoiem. Category Independent Object Proposals. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *European Conference on Computer Vision (ECCV)*, pages 575–588, Berlin, Heidelberg, 2010. Springer.

[26] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable Object Detection Using Deep Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2155–2162. IEEE, June 2014.

[27] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.

[28] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, Sept 2010.

[29] Sanja Fidler, Roozbeh Mottaghi, Alan Yuille, and Raquel Urtasun. Bottom-Up Segmentation for Top-Down Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3294–3301. IEEE, June 2013.

[30] Juergen Gall and Victor Lempitsky. Class-Specific Hough Forests for Object Detection. In *Decision Forests for Computer Vision and Medical Image Analysis*, pages 143–157. Springer, London, 2013.

[31] Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448. IEEE, Dec 2015.

[32] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. IEEE, June 2014.

[33] D González-Ortega, FJ Díaz-Pernas, Mario Martínez-Zarzuela, Miriam Antón-Rodríguez, JF Díez-Higuera, and Daniel Boto-Giralda. Real-time hands, face and facial features detection and tracking: Application to cognitive rehabilitation tests monitoring. *Journal of Network and Computer Applications*, 33(4):447–466, 2010.

[34] Waqas Haider, Hadia Bashir, Abida Sharif, Irfan Sharif, and Abdul Wahab. A survey on face detection and recognition approaches. *Research Journal of Recent Sciences*, 3:56–62, 01 2014.

[35] Edward Twitchell Hall. *The hidden dimension*. Doubleday & Co, 1966.

[36] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE, Oct 2017.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, Sept 2015.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, June 2016.

[39] Erik Hjelmås and Boon Kee Low. Face Detection: A Survey. *Computer Vision and Image Understanding*, 83(3):236 – 274, 2001.

[40] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint (CoRR)*, abs/1704.04861, 2017.

[41] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K Jain. Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):696–706, May 2002.

[42] Peiyun Hu and Deva Ramanan. Finding Tiny Faces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1522–1530. IEEE, July 2017.

[43] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3296–3297. IEEE, July 2017.

[44] Yichao Huang, Xiaorui Liu, Lianwen Jin, and Xin Zhang. DeepFinger: A Cascade Convolutional Neuron Network Approach to Finger Key Point Detection in Egocentric Vision with Mobile Camera. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 2944–2949. IEEE, Oct 2015.

[45] Nikolaos Kyriazis Iason Oikonomidis and Antonis Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *British Machine Vision Conference (BMVC)*, pages 101.1–101.11. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.101.

[46] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[47] Vidit Jain and Erik Learned-Miller. FDDB: A Benchmark for Face Detection in Unconstrained Settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[48] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *ACM International Conference on Multimedia (ACMMM)*, MM '14, pages 675–678, New York, NY, USA, 2014. ACM.

[49] Agnes Just, Yann Rodriguez, and Sebastien Marcel. Hand Posture Classification and Recognition using the Modified Census Transform. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 351–356. IEEE, April 2006.

[50] Maria Karam. *PhD Thesis: A framework for research and design of gesture-based human-computer interactions.* PhD thesis, University of Southampton, 2006.

[51] Mathias Kölsch and Matthew Turk. Robust Hand Detection. In *IEEE International Conference on Automatic Face Gesture*, pages 614–619. IEEE, May 2004.

[52] David Kortenkamp, Eric Huber, and R. Peter Bonasso. Recognizing and interpreting gestures on a mobile robot. In *National Conference on Artificial Intelligence*, volume 2 of *AAAI'96*, pages 915–921. AAAI Press, 1996.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. Curran Associates, Inc., 2012.

[54] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2169–2178. IEEE, 2006.

[55] T Hoang Ngan Le, Chenchen Zhu, Yutong Zheng, Khoa Luu, and Marios Savvides. Robust hand detection in Vehicles. In *IEEE International Conference on Pattern Recognition (ICPR)*, pages 573–578. IEEE, Dec 2016.

[56] Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann.lecun.com/exdb/lenet*, 1998.

[57] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust Object Detection with Interleaved Categorization and Segmentation. *International Journal of Computer Vision*, 77(1):259–289, May 2008.

[58] Hui Liang, Junsong Yuan, and Daniel Thalmann. Hand Pose Estimation by Combining Fingertip Tracking and Articulated ICP. In *ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry (VRCAI)*, VRCAI '12, pages 87–90, New York, NY, USA, 2012. ACM.

[59] Michael Lichtenstern, Martin Frassl, Bernhard Perun, and Michael Angermann. A Prototyping Environment for Interaction Between a Human and a Robotic Multiagent System. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, HRI '12, pages 185–186, New York, NY, USA, 2012. ACM.

[60] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint (CoRR)*, abs/1312.4400, 2013.

[61] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

[62] Matthew M. Loper, Nathan P. Koenig, Sonia H. Chernova, Chris V. Jones, and Odest C. Jenkins. Mobile Human-robot Teaming with Environmental Tolerance. In *ACM/IEEE International Conference on Human Robot Interaction (HRI)*, HRI '09, pages 157–164, New York, NY, USA, 2009. ACM.

[63] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics Based 3D Skeletal Hand Tracking. In *Graphics Interface*, GI '13, pages 63–70, Toronto, Ont., Canada, Canada, 2013. Canadian Information Processing Society.

[64] Brian Milligan, Greg Mori, and Richard Vaughan. Selecting and Commanding Groups of Robots in a Vision Based Multi-Robot System (video - winner of Best Video prize) . In *Video Proceedings of ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Lausanne, Switzerland, March 2011. ACM. https://youtu.be/Fq2qTlSuGqs.

[65] Sushmita Mitra and Tinku Acharya. Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(3):311–324, May 2007.

[66] Kensho Miyoshi, Ryo Konomura, and Koichi Hori. Entertainment Multi-rotor Robot That Realises Direct and Multimodal Interaction. In *International BCS Human Computer Interaction Conference on HCI 2014 - Sand, Sea and Sky - Holiday HCI (BCS-HCI)*, BCS-HCI '14, pages 218–221, UK, 2014. BCS.

[67] M. Monajjemi, S. Mohaimenianpour, and R. Vaughan. UAV, come to me: End-to-end, multi-scale situated HRI with an uninstrumented human and a distant UAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4410–4417. IEEE, Oct 2016.

[68] Mani Monajjemi. *End-To-End and Direct Human-Flying Robot Interaction*. PhD thesis, Simon Fraser University, August 2016.

[69] Mani Monajjemi, Jake Bruce, Seyed Abbas Sadat, Jens Wawerla, and Richard Vaughan. UAV, do you see me? Establishing mutual attention between an uninstrumented human and an outdoor UAV in flight. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3614–3620. IEEE, Sept 2015.

[70] V. M. Monajjemi, J. Wawerla, R. Vaughan, and G. Mori. HRI in the sky: Creating and commanding teams of UAVs with a vision-mediated gestural interface. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 617–623. IEEE, Nov 2013.

[71] Jawad Nagi, Alessandro Giusti, Luca M Gambardella, and Gianni A Di Caro. Human-swarm interaction using spatial gestures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3834–3841. IEEE, Sept 2014.

[72] Mahyar Najibi, Pouya Samangouei, Rama Chellappa, and Larry Davis. SSH: Single Stage Headless Face Detector. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4885–4894. IEEE, Oct 2017.

[73] Tayyab Naseer, Jürgen Sturm, and Daniel Cremers. FollowMe: Person following and gesture recognition with a quadrocopter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 624–630. IEEE, Nov 2013.

[74] Wai Shan Ng and Ehud Sharlin. Collocated interaction with flying robots. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 143–149. IEEE, July 2011.

[75] Eshed Ohn-Bar and Mohan M Trivedi. Beyond just keeping hands on the wheel: Towards visual interpretation of driver hand motion patterns. In *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1245–1250. IEEE, Oct 2014.

[76] Eshed Ohn-Bar and Mohan M Trivedi. To boost or not to boost? On the limits of boosted trees for object detection. In *IEEE International Conference on Pattern Recognition (ICPR)*, pages 3350–3355. IEEE, Dec 2016.

[77] Eng-Jon Ong and Richard Bowden. A boosted classifier tree for hand shape detection. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 889–894. IEEE, May 2004.

[78] Dennis Perzanowski, Alan C Schultz, William Adams, Elaine Marsh, and Magda Bugajska. Building a multimodal human-robot interface. *IEEE Intelligent Systems*, 16(1):16–21, Jan 2001.

[79] Ekaterina Peshkova, Martin Hitz, and Bonifaz Kaufmann. Natural Interaction Techniques for an Unmanned Aerial Vehicle System. *IEEE Pervasive Computing*, 16(1):34–42, Jan 2017.

[80] Kevin Pfeil, Seng Lee Koh, and Joseph LaViola. Exploring 3D Gesture Metaphors for Interaction with Unmanned Aerial Vehicles. In *International Conference on Intelligent User Interfaces (IUI)*, IUI '13, pages 257–266, New York, NY, USA, 2013. ACM.

[81] Shokoofeh Pourmehr. *Multimodal Interfaces for Human-Robot Interaction.* PhD thesis, Simon Fraser University, December 2016.

[82] Shokoofeh Pourmehr, Mani Monajjemi, Jens Wawerla, Richard Vaughan, and Greg Mori. A Robust Integrated System for Selecting and Commanding Multiple Mobile Robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2874–2879. IEEE, May 2013.

[83] Xianlei Qiu and Shuying Zhang. Hand detection for grab-and-go groceries. *Stanford University Course Project Reports-CS231n Convolutional Neural Network for Visual Recognition.*, 2017. Available online: http://cs231n.stanford.edu/reports.html.

[84] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, volume 3, page 5. Kobe, 2009.

[85] Siddharth S. Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54, Jan 2015.

[86] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. IEEE, June 2016.

[87] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525. IEEE, July 2017.

[88] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99. Curran Associates, Inc., 2015.

[89] Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang, and Jian Sun. Object Detection Networks on Convolutional Feature Maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1476–1481, July 2017.

[90] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec 2015.

[91] Mohammad Amin Sadeghi and David Forsyth. 30Hz Object Detection with DPM V5. In *European Conference on Computer Vision (ECCV)*, pages 65–79, Cham, 2014. Springer.

[92] Andrea Sanna, Fabrizio Lamberti, Gianluca Paravati, and Federico Manuri. A Kinect-based natural interface for quadrotor control. *Entertainment Computing*, 4(3):179–186, 2013.

[93] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv preprint (CoRR)*, abs/1312.6229, 2013.

[94] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint (CoRR)*, abs/1409.1556, 2014.

[95] Vincent Spruyt, Alessandro Ledda, and Wilfried Philips. Robust Arm and Hand Tracking by Unsupervised Context Learning. *Sensors*, 14(7):12023–12058, 2014.

[96] Vincent Spruyt, Alessandro Ledda, and Wilfried Philips. *Robust and real-time hand detection and tracking in monocular video*. PhD thesis, Ghent University, 2015.

[97] Nikolay Stefanov, Aphrodite Galata, and Roger Hubbold. Real-time Hand Tracking With Variable-Length Markov Models of Behaviour. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 73–73. IEEE, June 2005.

[98] Björn Stenger. Template-Based Hand Pose Recognition Using Multiple Cues. *Asian Conference on Computer Vision (ACCV)*, pages 551–560, 2006.

[99] Björn Stenger, Arasanathan Thayananthan, Philip HS Torr, and Roberto Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1372–1384, Sept 2006.

[100] Rainer Stiefelhagen, C Fugen, R Gieselmann, Hartwig Holzapfel, Kai Nickel, and Alex Waibel. Natural human-robot interaction using speech, head pose and gestures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2422–2427 vol.3. IEEE, Sept 2004.

[101] Ting Sun, Shengyi Nie, Dit-Yan Yeung, and Shaojie Shen. Gesture-based piloting of an aerial robot using monocular vision. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5913–5920. IEEE, May 2017.

[102] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv preprint (CoRR)*, abs/1602.07261, 2016.

[103] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9. IEEE, June 2015.

[104] Christian Szegedy, Scott E. Reed, Dumitru Erhan, and Dragomir Anguelov. Scalable, High-Quality Object Detection. *arXiv preprint (CoRR)*, abs/1412.1441, 2014.

[105] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2553–2561. Curran Associates, Inc., 2013.

[106] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. *arXiv preprint (CoRR)*, abs/1512.00567, 2015.

[107] Florent Taralle, Alexis Paljic, Sotiris Manitsaris, Jordane Grenier, and Christophe Guettier. A Consensual and Non-ambiguous Set of Gestures to Interact with UAV in Infantrymen. In *ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 797–803, New York, NY, USA, 2015. ACM.

[108] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks. *ACM Transactions on Graphics*, 33(5):169:1–169:10, sep 2014.

[109] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[110] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[111] Stefan Waldherr, Roseli Romero, and Sebastian Thrun. A Gesture Based Interface for Human-Robot Interaction. *Autonomous Robots*, 9(2):151–173, Sep 2000.

[112] Hao Wang, Zhifeng Li, Xing Ji, and Yitong Wang. Face R-CNN. *arXiv preprint (CoRR)*, abs/1706.01061, 2017.

[113] Robert Y. Wang and Jovan Popović. Real-time Hand-tracking with a Color Glove. *ACM Transactions on Graphics*, 28(3):63:1–63:8, July 2009.

[114] Yitong Wang, Xing Ji, Zheng Zhou, Hao Wang, and Zhifeng Li. Detecting Faces Using Region-based Fully Convolutional Networks. *arXiv preprint (CoRR)*, abs/1709.05256, 2017.

[115] Haibin Yan, Marcelo H. Ang, and Aun Neow Poo. A Survey on Perception Methods for Human–Robot Interaction in Social Robots. *International Journal of Social Robotics*, 6(1):85–119, Jan 2014.

[116] Shiyang Yan, Yizhang Xia, Jeremy S Smith, Wenjin Lu, and Bailing Zhang. Multiscale Convolutional Neural Networks for Hand Detection. *Applied Computational Intelligence and Soft Computing*, 2017:13, 2017.

[117] Bin Yang, J. Yan, Z. Lei, and S. Z. Li. Aggregate channel features for multi-view face detection. In *IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–8. IEEE, Sept 2014.

[118] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. From Facial Parts Responses to Face Detection: A Deep Learning Approach. In *IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 3676–3684, Washington, DC, USA, 2015. IEEE.

[119] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. WIDER FACE: A Face Detection Benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5525–5533. IEEE, June 2016.

[120] Shuo Yang, Yuanjun Xiong, Chen Change Loy, and Xiaoou Tang. Face Detection through Scale-Friendly Deep Convolutional Networks. *arXiv preprint (CoRR)*, abs/1706.02863, 2017.

[121] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Deep Metric Learning for Person Re-identification. In *IEEE International Conference on Pattern Recognition (ICPR)*, pages 34–39. IEEE, Aug 2014.

[122] Stefanos Zafeiriou, Cha Zhang, and Zhengyou Zhang. A survey on face detection in the wild: Past, present and future. *Computer Vision and Image Understanding*, 138:1 – 24, 2015.

[123] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *European Conference on Computer Vision (ECCV)*, pages 818–833, Cham, 2014. Springer.

[124] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016.

[125] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z. Li. S^3FD: Single Shot Scale-invariant Face Detector. *arXiv preprint (CoRR)*, abs/1708.05237, 2017.

[126] Feng Zhou, Jonathan Brandt, and Zhe Lin. Exemplar-Based Graph Matching for Robust Facial Landmark Localization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1025–1032. IEEE, Dec 2013.

[127] Tian Zhou, Preeti J Pillai, and Veera Ganesh Yalla. Hierarchical context-aware hand detection algorithm for naturalistic driving. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1291–1297. IEEE, Nov 2016.

[128] Chenchen Zhu, Yutong Zheng, Khoa Luu, and Marios Savvides. *CMS-RCNN: Contextual Multi-Scale Region-Based CNN for Unconstrained Face Detection*, pages 57–79. Springer, Cham, 2017.

Appendix A

Media

- Supplementary material filename: `demo.mp4`
- Online video URI: `https://youtu.be/7vbmOWVxGPU`
- Description: A demonstration of our system for un-instrumented human-UAV interaction. In this video, an un-instrumented user controls a UAV by commanding it to take-off, he freely manoeuvres the UAV around and asks to take a picture and perform acrobatic flips, and finally lands the UAV using hand gestures only without any external equipment.