Hands and Faces, Fast: Mono-Camera User Detection Robust Enough to Directly Control a UAV in Flight

Sepehr MohaimenianPour and Richard Vaughan*

***Abstract*— We present a robust real-time system for simultaneous detection of hands and faces in RGB and gray-scale images, and a novel dataset used for training. Our goal is to provide a robust sensor front-end suitable for real-time human-robot interaction using face-engagement and gestures. Using hand-labelled videos obtained from real human-UAV interaction experiments, we re-trained the YOLOv2 Deep Convolutional Neural Network to detect only hands and faces. This model was then used to automatically label several much larger third-party datasets. After manual correction of these results, we modified and re-trained the model on all these labelled data. We obtain qualitatively good detection results at 60Hz on a commodity GPU: our simultaneous hand-and-face 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 simple but practical gestural human-UAV interface for entertainment or industrial applications. All software, training and test data are freely available.**

I. Introduction

Several researchers have demonstrated Human-Robot Interfaces that use either face detection, hand detection or both [1]. We present a fast and accurate detector that finds the hands and faces of multiple people in RGB images at frame-rate, which can be used directly as an input to an HRI system. This extends our work on HRI for UAVs with *un-instrumented* users [2].

The contributions of this paper are: (i) we describe and provide software to simultaneously locate all the hands and faces from multiple people in a single 2D camera image, with state of the art accuracy and speed compared to dedictated face detectors and hand detectors. The software uses a scalable CNN model that can be resized for speed/accuracy trade-off based on YOLOv2 [3]; (ii) the first integrated hands and faces detection dataset for HRI, gathered from real HRI experiments and labeled by hand, plus new labels for previous well-known datasets; (iii) 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 un-instrumented human-UAV interaction; (iv) a demonstration of a real-world, end-to-end, close-range interaction system by which an un-instrumented user can take-off, steer, command and land a UAV by gestures alone with a single on-board camera. This is the first demonstration of 'joysticking' a UAV with directional commands by an un-instrumented user with only mono-camera sensing.

*Autonomy Lab, School of Computing Science, Simon Fraser University, Canada {smohaime, vaughan}@sfu.ca

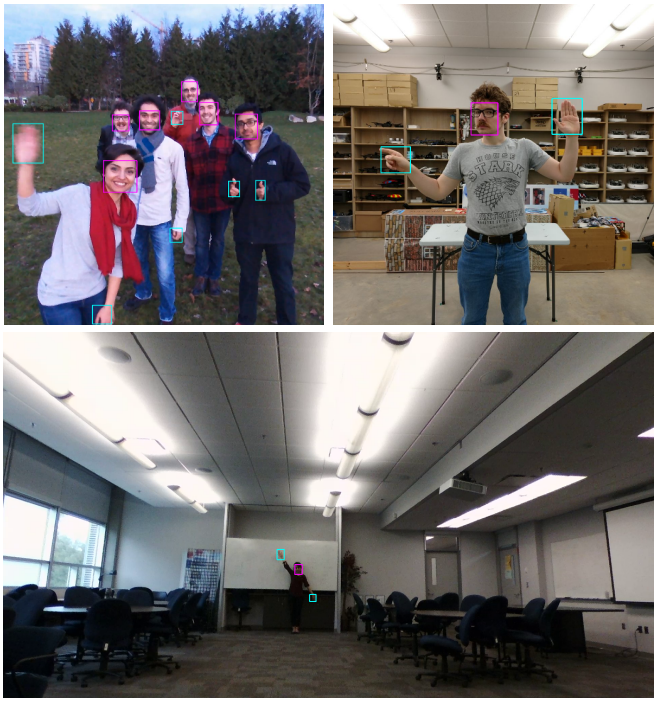

Fig. 1: Typical outputs: hands and faces detected accurately in 15msec per frame in images from a commodity UAV in flight. The user in the lower image is 10m away.

We evaluate our implementation by comparing it with dedicated hand detectors and face detectors in offline benchmarks, and by real-world experiments determining the gesture recognition accuracy versus human-robot distance in various environments under different illuminations and backgrounds.

II. Background

A. Object Detection

Object detection in a 2D image is a crucial and challenging topic in computer vision. Researchers have examined many approaches, generally involving extraction of low-level features from the image, followed by finding the features that match the object of interest.

The earliest widely-used object detection method was *Viola-Jones*[4], using Haar-like features. Later methods use other hand-designed features, notably *Discriminatively trained part based models* (DPM) [5] using *Histogram of Oriented Gradients* (HOG) features, and *Integral Channel Features* (ICF) [6] which combined multiple transformations

Preprint submitted to 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 1, 2018.

(*i.e.* color, gradient, edge, gradient histogram, difference-of-Gaussians, thresholding, and absolute value of Gaussian). Most pre-CNN detectors use some combination of these methods for feature extraction, followed by Support Vector Machines (SVM) or Adaptive Boosting (AdaBoost) for object classification.

AlexNet [7] showed the effectiveness of Convolutional Neural Networks (CNNs) for object detection, winning the ILSVRC 2012 image detection challenge by a large margin. Since then, CNNs have been the main 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. Object bounding box proposals are generated either before passing the image through the feature extractor, or after extracting the features from the image and considering those features in the region proposal generation. Two kinds are seen: direct classification using region-free methods [3], [8], [9], [10], [11] and refined classification approaches in region-based models [12], [13], [14], [15]. In general, region-free methods are faster but less accurate than region-based methods. An examination of speed/accuracy trade-off in recent CNN object detectors can be found in [16].

Region-based methods first regress the region proposal for a refined bounding box and then pool the features of the refined region from the common features volume and classify the object by these features. Listing notable examples: R-CNN [12] uses Selective Search [17] to generate almost 2000 region proposals per image, crops these proposals, and feeds them to AlexNet for classification. Subsequently, Faster R-CNN [13] improved accuracy and speed by replacing selective search with a Region Proposal Network (RPN) as suggested in [11] as well as classification by taking advantage of VGG16 instead of AlexNet. Another variant is R-FCN [14], which uses fully-convolutional RPN with almost all computation shared on the entire image, in contrast to previous region-based detectors that apply a costly per-region sub-network hundreds of times. Recently, Mask R-CNN [15] extended the Faster R-CNN to efficiently detect objects while simultaneously generating a high-quality segmentation mask for each instance, resulting in pixel level object detection.

In the alternate approach, region-free models simultaneously regress region proposals and classify objects directly from the same input region. OverFeat [9] uses AlexNet to extract the features at multiple evenly-spaced square windows in the image over multiple scales. MultiBox [11] was not an object detection method, but introduced CNN-based region extraction using RPNs. You Only Look Once [8] (YOLO) extends MultiBox from a region proposal solution to an object detector by adding a softmax layer, in parallel with the box regressor and box classifier layers, to directly predict the object class. SSD [10] increases the diversity of region proposal resolution by running the Faster R-CNN RPN on multiple convolutional layers at different depth levels. Finally, YOLOv2 [3] is a fully convolutional version of YOLO [8] integrating several techniques such as batch-normalization, anchor boxes, multi-scale training, as well as a new image classifier model called Darknet19 with Network in Network and Residual layers. We chose YOLOv2 as the basis of our system due to its combination of speed and accuracy.

B. Hand and Face Detection

Hand detection and hand pose recovery are important problems because of the importance of hands in communication and many other applications. Hand detectors are less successful than face detectors because of the difficulty of the problem. State-of-the-art hand detectors are mostly multi-modal methods that combine several different features. Skin color is a substantial module in most methods, which makes them vulnerable to illumination changes. Gloves and extreme differences in skin color are also challenging for these approaches. Most of these methods are too slow [18], [19] to be used for real-time purposes. Spruyt *et al.* have designed a near-real-time multi-modal method for hand detection and tracking [20] which only works from very close-range, in a controlled environment at 10-14 FPS. The closest work to ours is [19], but their choice of the R-CNN for hand detection makes it too slow for real-time use, and [21] which focuses on very close range hands in egocentric first-person views.

Face detection in RGB images is very well explored in the literature using both now-classical and CNN methods. The structure of faces make them relatively easy to see, and several methods detect faces well. However only a few methods achieve the combination of high-accuracy and real-time performance required for HRI. The OpenCV face detector[1] is fast and widely used in robotics, however it suffers from many false positives that make it hard to use for robust HRI. *dlib*[2] is another open source face detector which uses Deep Metric Learning [22] method and HOG features for face detection. dlib is more accurate but slower than the OpenCV detector. Most of these methods can not be repurposed for hand detection.

C. Situated Human-UAV Interaction

Using gestural interfaces for communicating humans' intent to a UAV has become popular. In [23] and [24] authors use a Microsoft Kinect on-board a flying UAV to perform gesture detection and human following. These sensors are improving, but RGB-D sensors have frustrating problems, *e.g.* poor results in direct sunlight, large data size, low resolution and limited range. In [25] the authors apply transfer learning to develop a person-specific gestural interface to command a UAV. Monajjemi *et al.* [2], [26] use hand waving (*i.e.* motion-based) detection to create a small vocabulary of gestures (*right*, *left*, and *dual* hand waving) to give commands to a UAV. In [27] the authors argue that in order to increase the visibility of the hand and reduce the effect of environment, they use arm gestures instead of hand gestures, using skin detection. Other authors simplify the task by having the user wear brightly-colored gloves [28].

[1] `https://sourceforge.net/projects/opencvlibrary/`
[2] dlib, Author: King, DE, Access on: `http://dlib.net`

III. METHOD

As in our previous work [26] our simple hardware setup is a consumer Parrot Bebop2 UAV with a commodity gaming laptop with GPU for offboard computation[3]. The UAV has a rolling shutter camera, so fast camera and subject movement distorts the shape of objects. The UAV streams compressed video and telemetry to the laptop by WiFi, and control commands are sent back. Our detector must therefore be robust to rolling shutter and video compression distortions. The lightweight plastic UAV is safe to use close to people. At any time a safety pilot can take control using a dedicated hardware controller.

Our system starts with the UAV on the ground and its motors disabled, waiting for the take-off command. The laptop computer processes every frame of the streamed video to locate hands and faces using our CNN hands-and-face detector. Given these detections, which may come from several people, we use Autonomy_Human[4] [2] to choose a single interaction partner in front of the robot as in [26]. Our new detector was a drop-in replacement for the OpenCV Viola-Jones face detector we previously used. The detected hands and face of the interaction partner are fed to our gesture recognizer at frame-rate (30Hz) in the form of regions of interest (ROIs) with class ids and confidence. The gesture detector considers proposed ROIs above some confidence threshold and attempts to classify a static gesture from our vocabulary (Figure 2) based on the user's *relative* hands and face positions. A behaviour state-machine module maps detected gestures to low-level commands (*e.g.* velocity commands, take a picture, take-off/land) and transmits them to the UAV for execution. Thus we detect gestures in a single frame. While incorrect gesture detections are rare, we were very cautious for safety reasons since gestures were directly flying the robot. Gestures were only accepted as commands after identical detection in three consectutive frames, giving a response time of $\approx$ 100ms in 30Hz transmitted video. Also for safety our gestures are only for lateral and vertical movements: we don't allow gestures to control the forward motion of the UAV, which might collide with the user. An independent controller with depth estimation based on the apparent size of the user tries to keep the UAV within configurable constant safety distance bounds from the user [26]. We used 2.5m to 10m.

The behaviour module also adjusts the tilt angle of the UAV's camera based on altitude to keep the interaction partner roughly at the center of the camera's vertical field of view.

The UAV constantly communicates its state and intents to the user with its front facing colored-light-based feedback system [26], with a new set of animations. When no user command is detected, the UAV enters the *idle* state and hovers stationary in front of the user (unless landed).

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

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

A. CNN Model

YOLOv2 [3] is a generic object detection system, in which a CNN predicts bounding boxes and label probabilities for objects. This model produces state-of-the-art results in terms of both accuracy and speed by outperforming other near-real-time CNN object detectors such as Faster R-CNN [13] and SSD [10]. The main idea behind YOLO is that the model can predict object bounds by a single forward pass on the image while exploiting global context from the whole image. This behavior is ideal for our application, since it reduces the false-positive rate as hand and face locations and mutual appearance are correlated with each other and with those of other body parts.

We used the Darknet19 object classifier model as our feature extractor. In order to use YOLOv2 for detecting just our two object classes - hands and faces - we changed the number of filters in the last convolutional layer of the model before performing the object classification in the region layer.

Redmon *et al.* showed since their model uses only convolution 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 [3]. Following their suggestion, we scaled our model for the second set of training to take 736×736 pixel images as input, and after down-sampling by a factor of 32 it will divide the image to 23×23 regions to predict the anchor boxes. The original YOLOv2 model is designed such that their smallest 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 60 FPS 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.

B. Training

For the first round of training on the first set of data, we used the Darknet19_448 model pretrained on the ImageNet 1000-class dataset as the initial weights for the first 23-layer feature extraction part of the model. The bounding box regressor and classification layers weights were initialized randomly. Using our novel human-UAV HRI dataset described below, we trained the modified model for 80 epochs with a starting learning rate of 10^{-3}, dividing it by 10 at the 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.

In a bootstrap approach, we take the trained model and use it to annotate larger third-party generic hand datasets and face datasets, correct any errors by hand, then fine-tune the model with a combination of all datasets. Dataset engineering is discussed further in Section IV. For training the final model, as discussed above we resized it to accept 736×736 (23×23 grid) pixel images as input instead of 416×416 (13×13 grid). To retain the useful multi-scale feature of YOLOv2 we occasionally randomly resized the model during training

from 320×320 (10×10 grid: the fastest model) to 960×960 (30×30 grid: highest accuracy). For the last epoch we disabled the resizing and kept the input size as 736×736 which was our preferred model. We trained the network for 90 epochs with an initial learning rate of 10^{-3}, dividing it by 10 at the 30th, 45th, and 60th epochs. We used a weight decay of 0.0005 and momentum of 0.9.

The Darknet[5] framework supports a set of pre-defined dataset augmentations. We added some more augmentation methods to increase the number of training images and cover a larger part of the appearance space. We randomly crop up to 40% of the image's height and width separately, then scale the resulting image randomly between 25% to 200%, arbitrarily flip the 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, and add a random rotational augmentation in the range of $-30°$ to $30°$.

C. Gesture Detection

Researchers have been working on defining efficient gestural vocabularies for natural interaction with UAVs for the last decade. In [29], [30], [31] authors have defined different sets of gestural vocabularies with a user study in the *Wizard of Oz* manner. However none of these methods 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. [25] is one of the first successful attempts, however they need a data gathering and training phase for each user. The waving-based gestures in [2] and [26] are motion-based, using the FFT to detect periodic hand motion for gesture detection. These gestures are rather slow to pick up, vulnerable to losing some frames in the video stream and only three gestures have been shown. Relying on skin detection enabled detection of a user's arms and to generate richer commands for the drone [27], but skin detection lacks robustness and is not always feasible.

Rautaray *et al.* [1] propose 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 detect the static gestures directly from the detection results (with an optional 3-frame agreement policy for extreme confidence).

We consider a set of eight static hand gestures (*i.e.* take-off, land, move up, move down, move left, move right, flip, and take a picture) for controlling a drone as illustrated in Figure 2. 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 for positional gesture detection as illustrated in Figure 3. 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 only one hand is needed, we hold the other hand as a safety dead-man switch inside its home-zone region for these commands while giving the move direction using the free hand. Actively using the second hand as a dead-man switch is logically optional, but we recommend it as a safety feature when deliberately maneuvering the UAV close to a user.

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

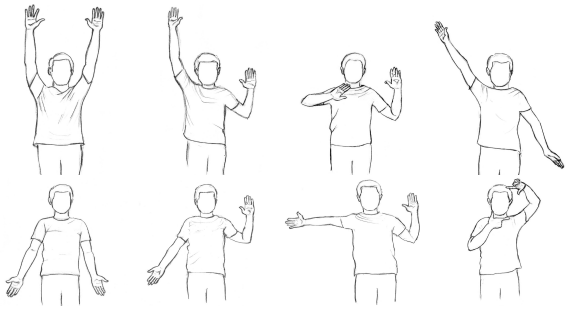

Fig. 2: Our hand gesture vocabulary for Human-UAV interaction. From left to right: ^ Take-off, ↑ Move up, → Move left, ↻ Flip, v Land, ↓ Move down, ← Move right, and 📷 Take a picture.

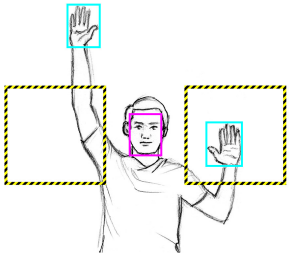

Fig. 3: Our static gesture detection works based on the position of the interaction partner's hands relative to her face. This shows the dead-zone rectangles which are anchored relative to the detected face position and size.

IV. Dataset Engineering

This paper is accompanied by the unique data used for training our model.

For the first round of training, we created a dataset from 69 drone camera videos we recorded in previous Human-UAV interaction work [26]. In the videos 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 also include some static hand gestures like pointing to different directions. The resulting dataset consists of $16,883$ selected frames. We hand labeled $34,588$ *hand*s and $18,838$ *face*s in the selected images.

We define our standard annotation format inspired by the Pascal VOC [32] *detection* challenge. Our annotation format is a tight axis-aligned bounding box around each object of interest in the image, for which at least 50% of the object is visible. To train the model using the Darknet framework, we needed to change the annotation so that each

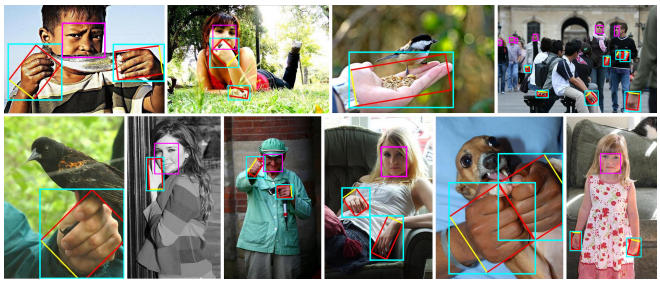

Fig. 4: Sample images from Mittal *et al.*'s hands dataset with bounding boxes overlaid. Red annotation rectangles are the original dataset annotation, in which rectangle sides are ordered so that the wrist is along the first side marked with a yellow line. Cyan bounding boxes are our converted annotations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

image has a single annotation text file, which includes one line per bounding box in the image. Each bounding box has a class ID $\in \{0, 1\}$ specifying {Hand, Face}, followed by four numbers in the range of $[0.0, 1.0]$. These numbers indicate the normalized value of the center position of the bounding box and its height and width based on the image's dimensions respectively.

To increase the amount and variety of data for training, we also made use of some third party datasets, most of which needed some modification to their annotation so that they could be used for our purpose. Mittal *et al.* provide one of the best publicly available hand detection datasets [18]. The provided annotation files are in standard Matlab format which contain annotations for the four end-points of the hand bounding boxes. The original annotation bounding boxes are not axis-aligned, thus we selected new axis-aligned bounding boxes enclosing the original annotations, Figure 4 illustrates this difference. Spruyt *et al.* also provide a small hand tracking dataset [20] under challenging illuminations. We mixed these two datasets and hand annotated the faces in them. A small portion of the Pascal VOC dataset, called "Person Layout", is annotated for the body parts (head, hand, and foot) detection. They have provided bounding boxes for these body parts in a portion of their dataset; we extracted the head and hand annotations from the '.xml' files, and removed the images in which annotated heads were not frontal faces. A combination of these four datasets was used for the first round of training.

We used our first trained model to annotate more images for creating a much larger labeled dataset than was previously available, in order to use them for training smaller models from scratch in the future. For annotating the rest of the data, we predicted the hands and faces in the images using 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. The remaining datasets were annotated using this method, which made the labeling procedure an order of magnitude faster.

The Helen dataset [33] is a face landmark detection dataset, which consists of $2,330$ very close-range images of faces. If there is more than one face in the image, the authors have annotated only the dominant face. We wrapped the landmarks in the image labels with an axis-aligned rectangle to convert the original annotations to our standard bounding boxes, then used our model to label the hands and the rest of the missing faces in the images.

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

TABLE I: Datasets we have annotated and used for training our models. The first four datasets were used in our first iteration of training, while the latter four are added for fine-tuning our second model.

The VIVA hand detection challenge's training dataset [34] consists of 2D bounding boxes around 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 occlusion with 7 possible viewpoints. In a similar setup, the EgoHands dataset [21] contains 48 Google Glass videos of complex, first-person interactions between two people with pixel level segmentation of the hands. We used these two datasets by converting their hands annotations to our standard bounding box annotation format and also labeling the occasional appearances of faces in the data.

Finally, the Faces in the Wild dataset [35], is a huge collection of faces from news photographs. The original annotation of these pictures only indicates who appears in the picture, without any information about the position of the face in the image. We picked half of the images in the dataset and labeled all the occurrence of hands and faces in those images. Table I shows the number of frames and annotated hands and faces in our datasets.

All our original labelled video and annotations for third-party datasets are freely available from `http://autonomylab.org/hands_and_faces`.

V. Experiments

To the best of our knowledge there is no other work in the literature for dedicated simultaneous hand and face detection with which to compare. Thus, to empirically evaluate performance, we compare hand and face detection separately with their respective benchmarks and datasets. Later we demonstrate the detector's suitability for practical HRI by using it to directly control a UAV in flight.

A. Hand Detection

To evaluate our hand detector, we used the benchmarking tool provided by the VIVA hand detection challenge [34].

Method	Simple AP	Simple AR	Hard AP	Hard AR	FPS
Auto. H&F 736×736	**96.1**	**94.9**	**93.7**	**87.8**	60
MS-RFCN	95.1	94.5	86.0	83.4	4.65
SCUT Aug. FRCNN	93.9	91.5	85.2	77.8	6.32
MS-RFCN	94.2	91.1	86.9	77.3	4.65
YOLOv2	93.4	91.0	84.0	74.4	123
Multi-scale fast RCNN	92.8	82.8	84.7	66.5	0.3
MS-FRCNN	90.8	84.1	77.6	65.1	
SCUT Aug. FRCNN	89.5	86.0	73.4	64.4	4.9
FRCNN	90.7	55.9	86.5	53.3	
Modified Faster-RCNN	81.7	59.2	72.8	47.2	
ACF_Depth4	70.1	53.8	60.1	40.4	
YOLO	76.4	46.0	69.5	39.1	35
Auto. H&F 160×160	74.6	53.6	62.1	38.6	**200**
SRS CNN	66.8	48.1	57.8	36.6	0.783
ACF	62.4	36.9	52.3	27.5	11.6

TABLE II: 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 Hard setting.

This benchmark 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. This kit considers the PASCAL overlap requirement of 50% as true detections.

The hand detection challenge is evaluated on two levels: (i) Simple: Hand instances with minimum height of 70 pixels, only over the shoulder camera view (ii) Hard: Hand instances with minimum height of 25 pixels, all camera views.

The results in Table II show our method achieves state-of-the-art scores[6] despite being also able to detect faces. The other methods detect hands only. We carefully engineered the network architecture to run at 60 frames per second with good accuracy because typical frame rates in robot applications are between 27 and 60 FPS. We also trained the YOLOv2 method out of the box for comparison which is ranked 5th in the Table II. Another state-of-the-art method for hand detection is the work by Mittal *et el.* [18] which takes approximately two minutes for an image as small as 640×360 pixels and is therefore not suited for real-time HRI applications.

B. Face Detection

To evaluate the face detector, we used the WIDER [36] face detection benchmark tool. WIDER also provides a large and diverse training dataset, organized based on 61 event classes, some of which were not suitable for HRI purposes.

We compare with previous methods of two kinds: (i) evaluating the available baseline methods on test and validation datasets (ii) evaluation on state-of-the-art face detectors trained on WIDER training data. We evaluate on the WIDER evaluation data (10% of the dataset). The evaluation data is divided into three different categories, *easy, medium,* and *hard*. WIDER also 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.

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

TABLE III: Average precision of methods on the WIDER benchmark. The horizontal line separates methods which were not (above) and were (below) trained on the WIDER training data.

As shown in Table III, our model outperforms all other state of the art methods which were not specifically designed and trained on the WIDER dataset[7]. Several methods trained on WIDER score better than ours, though ours is the fastest. We expect that training our model on WIDER would improve its performance.

C. End-to-end Close Range Human-UAV Interaction

Next we demonstrate that our hand-and-face detector is sufficiently fast and robust to use as the input for real-time control of a UAV in flight by an un-instrumented user.

We evaluate the system using the hardware setup described in Section III, with ROS [37] for integration. More details about our system, as well as the source code for various components (including hands-and-face detection models, our datasets, the ROS wrapper for the Darknet framework, and our gesture detection module) is available from the URL above.

In order to test the effective range of our detector with a flying UAV, we performed a set of experiments varying the distance between the UAV and the user from 2.5 to 10m. We measured hand and face detection rates and gesture detection accuracy. These experiments were performed in two indoor and two outdoor locations with different lighting and against different backgrounds, shown in Figure 5.

All the experiments were repeated with one male and one female user. Each user performed all 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 (30 frames). The results are

[6]More information on the methods: `http://cvrr.ucsd.edu/vivachallenge/index.php/hands/hand-detection/`

[7]More information on the methods: `http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/WiderFace_Results.html`

Fig. 5: Scale, illumination, and background changes in our evaluation.

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 IV: 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.

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

shown in Table IV. We also counted the number of correctly detected hands and faces during each scenario to evaluate our detector's accuracy as shown in Table V.

The results show that the 'take a picture' gesture was the hardest to detect for our system. It was the first to degrade with distance, probably because hands could partially overlap the face and YOLO reports only one object per grid cell. The 'landing' gesture was the easiest to recognize, probably because this gesture is similar to a normal standing posture which was very common in the training data. Thus it might be vulnerable to false positives in practice. We also tested our system at 20m range with hands and faces detected correctly in less than 1% of cases, thus we were not able to detect any gesture at this distance. The false positive detection rate over all experiments was less than 0.06%.

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 color-based LED feedback). At distances further than 5m, the human's ability to detect the UAV's movements and intents properly is degraded, and control becomes difficult.

All the UAV video and labelled gesture data from these experiments is released as a test dataset for UAV HRI.

VI. Conclusion and Future Work

In this paper we proposed, demonstrated and provide for use a CNN-based vision system that can accurately detect hands and faces for human-robot interaction. A video using our system in the loop for controlling a flying UAV accompanies this paper and is available online. Our system is robust to scale, illumination and background changes sufficient for practical use indoors and outdoors at ranges of at least 10m on a cheap commodity UAV. We showed that an un-instrumented user can control a co-located UAV using only a monocular camera on-board the UAV with off-board processing at a ground control station via WiFi. We examined a set of eight gestures for controlling a situated UAV, allowing a user to command a drone 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 controller. We defined a novel method for introducing static gestures based on the relative position of a user's face and hands, with important safety features.

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 video at 30 FPS on a commodity GPU. 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. All training and test data, the trained CNN model, and robot source code are freely available from our web site.

Our next goals are (i) to associate hands with corresponding faces in images with multiple people; (ii) to track hands and faces in video directly using a recurrent network; (iii) optimize the network size so it can run real-time on CPUs.

Preprint submitted to 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 1, 2018.

REFERENCES

[1] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, Jan 2015.

[2] 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 *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2013, pp. 617–623.

[3] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6517–6525.

[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2001, pp. I–511–I–518 vol.1.

[5] 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*, vol. 32, no. 9, pp. 1627–1645, Sept 2010.

[6] P. Dollar, Z. Tu, P. Perona, and S. Belongie, "Integral Channel Features," in *2009 British Machine Vision Conference (BMVC)*. BMVA Press, 2009, pp. 91.1–91.11, doi:10.5244/C.23.91.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems (NIPS) 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–788.

[9] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," *CoRR*, vol. abs/1312.6229, 2013.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 21–37.

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

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014, pp. 580–587.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems (NIPS) 28*. Curran Associates, Inc., 2015, pp. 91–99.

[14] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," in *Advances in Neural Information Processing Systems (NIPS) 29*. Curran Associates, Inc., 2016, pp. 379–387.

[15] K. He, G. Gkioxari, P. Dollr, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2980–2988.

[16] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 3296–3297.

[17] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[18] A. Z. Arpit Mittal and P. Torr, "Hand detection using multiple proposals," in *2011 British Machine Vision Conference (BMVC)*. BMVA Press, 2011, pp. 75.1–75.11.

[19] S. Yan, Y. Xia, J. S. Smith, W. Lu, and B. Zhang, "Multiscale Convolutional Neural Networks for Hand Detection," *Applied Computational Intelligence and Soft Computing*, vol. 2017, p. 13, 2017.

[20] V. Spruyt, A. Ledda, and W. Philips, "Robust Arm and Hand Tracking by Unsupervised Context Learning," *Sensors*, vol. 14, no. 7, pp. 12 023–12 058, 2014.

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

[22] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep Metric Learning for Person Re-identification," in *2014 22nd International Conference on Pattern Recognition (ICPR)*, Aug 2014, pp. 34–39.

[23] M. Lichtenstern, M. Frassl, B. Perun, and M. Angermann, "A Prototyping Environment for Interaction Between a Human and a Robotic Multi-agent System," in *7th Annual ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, ser. HRI '12. New York, NY, USA: ACM, 2012, pp. 185–186.

[24] T. Naseer, J. Sturm, and D. Cremers, "FollowMe: Person following and gesture recognition with a quadrocopter," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2013, pp. 624–630.

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

[26] 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 *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4410–4417.

[27] T. Sun, S. Nie, D. Y. Yeung, and S. Shen, "Gesture-based piloting of an aerial robot using monocular vision," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5913–5920.

[28] J. Nagi, H. Ngo, L. M. Gambardella, and G. A. D. Caro, "Wisdom of the swarm for cooperative decision-making in human-swarm interaction," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1802–1808.

[29] W. S. Ng and E. Sharlin, "Collocated interaction with flying robots," in *20th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, July 2011, pp. 143–149.

[30] F. Taralle, A. Paljic, S. Manitsaris, J. Grenier, and C. Guettier, "A Consensual and Non-ambiguous Set of Gestures to Interact with UAV in Infantrymen," in *33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '15. New York, NY, USA: ACM, 2015, pp. 797–803.

[31] J. R. Cauchard, J. L. E, K. Y. Zhai, and J. A. Landay, "Drone & Me: An Exploration into Natural Human-drone Interaction," in *2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '15. New York, NY, USA: ACM, 2015, pp. 361–365.

[32] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun 2010.

[33] F. Zhou, J. Brandt, and Z. Lin, "Exemplar-Based Graph Matching for Robust Facial Landmark Localization," in *2013 IEEE International Conference on Computer Vision (ICCV)*, Dec 2013, pp. 1025–1032.

[34] N. Das, E. Ohn-Bar, and M. M. Trivedi, "On Performance Evaluation of Driver Hand Detection Algorithms: Challenges, Dataset, and Metrics," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, Sept 2015, pp. 2953–2958.

[35] T. L. Berg, A. C. Berg, J. Edwards, and D. A. Forsyth, "Who's in the Picture?" in *17th International Conference on Neural Information Processing Systems (NIPS)*, ser. NIPS'04. Cambridge, MA, USA: MIT Press, 2004, pp. 137–144.

[36] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A Face Detection Benchmark," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 5525–5533.

[37] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.