Fall monitoring and detection for at-risk persons using a UAV

Cristi Iuga, Paul Drăgan, Lucian Buşoniu

Technical University of Cluj-Napoca, Memorandumului 28, 400114 Cluj-Napoca, Romania
(e-mails: iugacristi@gmail.com, paul.andrei.dragan@gmail.com, lucian@busoniu.net)

Abstract: We describe a demonstrator application that uses a UAV to monitor and detect falls of an at-risk person. The position and state (upright or fallen) of the person are determined with deep-learning-based computer vision, where existing network weights are used for position detection, while for fall detection the last layer is fine-tuned in additional training. A simple visual servoing control strategy keeps the person in view of the drone, and maintains the drone at a set distance from the person. In experiments, falls were reliably detected, and the algorithm was able to successfully track the person indoors.

Keywords: unmanned aerial vehicles, deep learning, fall detection.

1. INTRODUCTION

More than a billion people today experience some form of disability, while the world population is rapidly ageing. Between 2000 and 2050, the proportion of people over 60 years will double from about 11% to 22%. We consider here a robotic assistant for elderly and disabled persons that must be constantly monitored due to the risk of falling. Falling frequency is about 28-35% each year for people over 64 years of age and 32%-42% for those over 70 (WHO, 2007).

State-of-the-art efforts in assistive robotics are predominantly placed on fixed manipulators or mobile ground robots, which are limited in the types of environments they can address (Boucher et al., 2013). On the other hand, unmanned aerial vehicles (UAVs) trade off manipulation capability to gain mobility and speed, being unaffected by terrain difficulty (Mathe and Busoniu, 2015). Despite their potential (Baer et al, 2014) UAVs have so far been virtually unexplored in assistive care.

In this paper we present a first application along this direction: a UAV that uses computer vision to monitor a person for falls while autonomously following them in an indoor environment. While UAVs have been widely used in disaster monitoring and search-and-rescue, see e.g. Andriluka et al. (2010), Murphy (2014), to our knowledge they have never been used for person fall detection. Instead, this is done usually with fixed cameras (Cucchiara et al., 2007, Skubic et al., 2016), wearable or other sensors (Bourke et al., 2007, de Lima et al. 2017). Our work is closely related to that of Andriluka et al. (2010), where UAVs are used to find injured persons in a search-and-rescue setting.

We choose the Parrot AR.Drone 2.0 quadrotor, a very popular low-cost UAV that is easy to automate using ROS (Robotic Operating System). Our method achieves person detection by running the deep learning object detector YOLOv2 (You Only Look Once version 2) (Redmon and Farhadi, 2017) on the images received from the Parrot AR.Drone 2.0. The output data from the detector is used to find the position and distance of person in the scene, and thereby to direct the drone to remain at a set distance and orientation from the person, using a simple visual servoing strategy (Thuilot et al., 2002). Fall detection is achieved with the same deep-learning method, YOLOv2, but in this case we fine-tune the last layer using a custom dataset. We chose a vision-based learning solution because it is simple to apply, flexible, and relies on a sensor already available on the UAV (the camera) rather than e.g. wearable sensors. All the detection and control algorithms run off-board, on a computer that wirelessly communicates with the drone. Our experimental results confirm that the method works for tracking the person indoors and detects falls reliably.

Next, Section 2 provides the background required in the vision techniques we use. Section 3 explains the most important component of our method – detection of upright and fallen persons from images. Section 4 outlines the method for tracking the person across multiple images. Section 5 presents the control technique as well as the overall results obtained. Section 6 gives our conclusions and outlines future work.

2. BACKGROUND

2.1 Classification and object detection from images

Image classification and object detection are two of the most important and well-studied computer vision tasks. The aim of classification is to assign a label to an image, where the label is taken from a fixed set of classes, while object detection focuses on localizing in an image all the objects that belong to one or multiple categories. Classification algorithms usually take as input an image and output a single label or class, while object detection algorithms output the enclosing bounding boxes and classes of objects present in a given image. With the advent of large image datasets, e.g. Imagenet (Russakovsky et

al., 2015) and MS-COCO (Lin et al., 2014), Convolutional Neural Networks (CNNs), became the *de facto* method of approaching both classification (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; He et al., 2016) and object detection (Ren et al., 2015; Redmon and Farhadi, 2017).

Detection of fallen persons (called simply fall detection in the sequel) can be formulated both as a classification task, where the algorithm should output whether an image contains a fallen person or not, or as an object detection problem, requiring the localization of all the fallen people in an image. Works such as Nunez-Marcos et al. (2017) take the first approach, while in this paper we will treat fall detection as an object detection task and we will make use of the YOLO (You Only Look Once) (Redmon and Farhadi, 2017) architecture to detect both upright and fallen people, the former being necessary to track the person with the drone.

2.2 YOLOv2 object detection network

YOLOv2 (Redmon and Farhadi, 2017) is a CNN architecture for object detection. The method obtained good results on the VOC 2012 detection dataset (Everingham et al., 2012) performing on par with state-of-the-art detectors at that time such as Faster R-CNN (Ren et al., 2015). Due to its lightweight structure, YOLO can run at around 40 FPS on a GeForce GTX Titan X and at between 20 and 25 FPS on a GeForce GTX 970, the graphics card in our hardware setup, rendering it attractive for applications that require soft real-time constraints.

YOLOv2 outputs, for each object O_j present in an image I, a probability distribution over the classes $c_i \in C$, where C is the set of all classes; and in addition a bounding box $B_j = [x, y, w, h]$ enclosing object O_j, where (x, y) are the coordinates of the top-left corner of the bounding box and (w, h) are its width and height, respectively. The network is trained in a supervised manner, meaning that both true labels and true bounding box coordinates must be fed for each object to the training algorithm.

3. PERSON AND FALL DETECTION

Due to its speed and accuracy, we select the YOLOv2 object detection framework, and in this section we explain how we adapt and use it for our application.

To detect the person in the upright position, we use the CNN with the standard set of weights pre-trained on the MS-COCO dataset. Although this method is sensitive to arm positioning, since it tends to enlarge the bounding box in order to include the possibly raised arms of a subject, it still achieves 75% accuracy. This result is good enough to infer distance information from the bounding boxes, keeping in mind that these bounding boxes are further filtered by using a Kalman filter, as explained in the upcoming Section 4. Thus, detection of a standing person works nearly off-the-shelf, and we dedicate the rest of this section to the more interesting fall detection task. Note that we aim to detect falls *a posteriori*, which may then be used to alarm a caretaker, rather than in real-time, which could conceivably help the monitored person recover from the fall; the latter would be unlikely to work given the limitations of our platform.

3.1 Fall detection methodology

As already hinted in Section 2.1, we formulate the problem of fall detection as an object detection task and use the YOLOv2 CNN architecture to identify the fallen person, if any. We also considered treating the problem as a classification task, but this would mean that the network would be difficult to train for unconstrained environments, body posture and placement, and for variations of clothing and body appearance. This would impair generalization in practice.

Furthermore, solving the problem with an object detection algorithm comes with certain advantages that will be useful in future extensions of our method. Firstly, the algorithm can detect multiple fallen people in the same scene and enclose them in the image within different bounding boxes. Secondly, having bounding boxes means that we know estimates of the position of the detected fallen persons in the scene, which can be useful, for example, for targeted medication delivery.

To work for fall detection, the YOLOv2 object detector has to be retuned. We start from the standard set of weights pre-trained on MS-COCO, which we used above for upright person detection. However, here we replace the last layer with a simpler one, capable of only single-class detection, and then fine-tune the network on a custom dataset. This custom training dataset consists of 500 manually labeled images, taken from the frames of two videos. These images contain a single person as subject, wearing the same set of clothes across all frames. The videos are recorded indoors, in our laboratory, using the AR.Drone 2.0 camera, from two different perspectives, while the UAV is inflight.

For training we use the stochastic gradient descent algorithm with momentum, with a learning rate of 0.001 and a momentum of 0.99. We use mini-batches that each consist of 4 randomly selected images. We train the network for 2000 iterations or 2000 mini-batches, meaning that each image is seen by the algorithm around 16 times.

While in principle the same network (or at least the common parts up to the last layer) can be used for both upright and fallen people detection, for ease of implementation and testing we decided to run two separate networks, one for upright person detection and one for fall detection, in parallel on separate threads.

3.2 Fall detection results

To evaluate the effectiveness of detection, we test the retuned network on a different dataset, which was obtained in a similar manner as the training dataset. In this case, however, we take the video from a single perspective, with two subjects, both wearing different clothes than in the training images. In total, we have gathered 619 images with our subjects in upright or fallen-like postures.

Table 1 shows the test results. We can see that, in spite of the small number of examples in the training dataset, the network obtains high accuracy and generalizes well for different clothing styles and different subjects. An example of the output from the specialized network can be observed in Fig. 1.

Table 1. Fall detection test results

	Total	**Positives**	**False positives**	**False negatives**
Number of images	619	535	37	84
Percentage	100%	86.24%	5.97%	13.57%

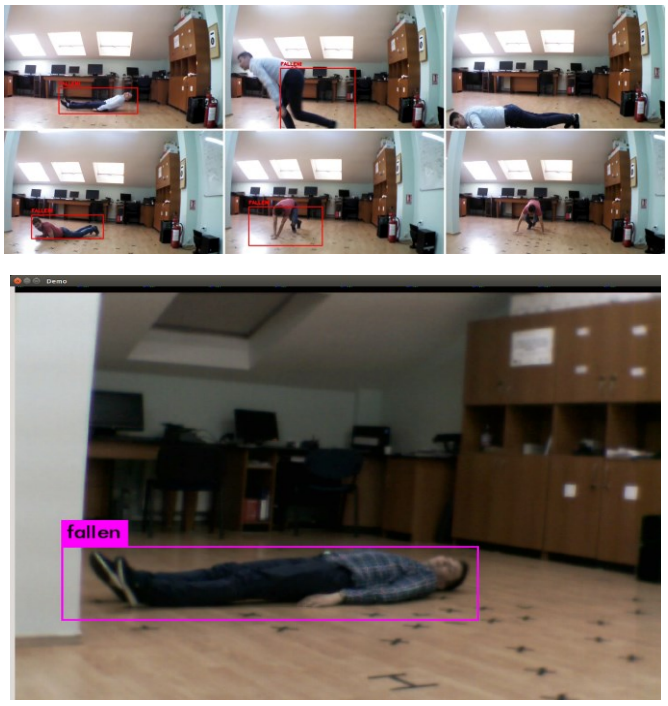

Fig. 1. Fall detection with the modified YOLOv2 CNN

4. PERSON TRACKING

In order to follow the person, the controller we will discuss in Section 5 takes as input a bounding box enclosing the subject in the image. The bounding box detected by YOLOv2 is sometimes too noisy and therefore the vertices coordinates change abruptly, causing performance issues with the control of the drone. We address this problem using a Kalman filter that smooths the bounding box outputs from the detection network. This will make the coordinates of the vertices of the bounding boxes change steadily as the target moves laterally, distances themselves from, or approaches the drone.

The model used in the Kalman filter equations is detailed next. We use the state vector:

$$x_k = [x_{1,k}\ \ y_{1,k}\ \ x_{2,k}\ \ y_{2,k}\ \ v_{x1,k}\ \ v_{y1,k}\ \ v_{x2,k}\ \ v_{y2,k}]^T$$

where $(x_{1,k}\ \ y_{1,k})$ and $(x_{2,k}\ \ y_{2,k})$ are the coordinates of the top-left and bottom-right corners of the bounding box, and $(v_{x1,k}\ \ v_{y1,k})$ and $(v_{x2,k}\ \ v_{y2,k})$ are the velocities of these corners. The discrete time sample is denoted by k. The measurement (output of the vision algorithm) only provides the positions of the corners, denoted together by $z_k = [x_{1,k}\ \ y_{1,k}\ \ x_{2,k}\ \ y_{2,k}]^T$. Denote also the vector of velocities $v_k = [v_{x1,k}\ \ v_{y1,k}\ \ v_{x2,k}\ \ v_{y2,k}]^T$.

The dynamics describing state transitions consists of a noisy first-order Euler integration of the velocities to obtain the positions, and random-walk velocities:

$$z_{k+1} = z_k + v_k \cdot T_S + w_{zk}\,,\ v_{k+1} = v_k + w_{vk}$$

Such a model is called a constant-velocity model in computer vision, where it is often used to track objects with unknown motion. These dynamics are noisy linear:

$$x_{k+1} = Ax_k + w_k$$

with $A = \begin{bmatrix} I_4 & T_S I_4 \\ 0_4 & I_4 \end{bmatrix}$ and the overall noise $w_k = [w_{zk}^T, w_{vk}^T]^T$ is zero-mean Gaussian with covariance matrix $Q = 10^{-4} I_8$.

For the measurement equation, the positions are mapped directly from the measurement to the state, while the velocities are not directly measured and thus they are not included in the mapping. This is represented as:

$$z_k = C \cdot x_k + u_k$$

with $C = [I_4 \quad 0_4]$. The measurement noise u_k is also zero-mean Gaussian, with covariance $R = 10 I_4$. The Kalman filter is run with the initial error covariance matrix $P_0 = 0.1 I_8$. For the equations of the Kalman filter see e.g. Ristic et al. (2004).

Fig. 2 illustrates the tracking results, with the filtered bounding box in cyan smoothly approaching the measurement from the classifier, shown in solid green.

Fig. 2. Measured and filtered box in two subsequent frames

5. DRONE CONTROL

In this section we present the final part of our application, the control strategy, as well as the overall experimental results.

5.1 Vision-based control

We apply a visual servoing strategy where the goal is to maintain the drone at a reference distance d_{ref} of 4 meters from the target, and to maintain the target in the center of the image. The commands correct the orientation θ of the drone on the z axis (the yaw) and the position on the x axis (represented as a distance d_x from the person), by altering the angular and linear velocity setpoints ω, V along these two axes. These setpoints are then sent to the AR.Drone 2 firmware, which applies a low-level control in order to track them. Fig. 3 visually illustrates the control strategy.

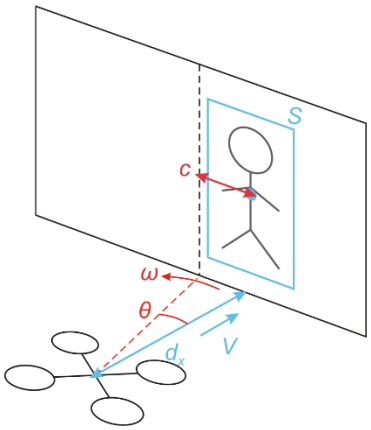

Fig. 3. Illustration of the vision-based control strategy

The feedback to the yaw and distance controllers is computed based on measurements derived from the filtered bounding box around the person. Specifically, the yaw controller uses the deviation of the box center from the center of the image in order to keep the target in the center of the frame, while the distance controller uses the computed distance from the target in order to keep the drone at the desired reference distance from the person. Experiments have shown that in our case, the distance at which the target is from the drone is most reliably computed from the area of the bounding box:

$$d_x = \alpha S + \beta$$

where S is the area, α is a scaling factor and β a scaling bias, both determined experimentally. The area is simply the width times the height of the box in pixels, see again Fig. 3.

Preliminary control experiments with the drone have shown that computing continuous commands and sending them at each frame to the drone does not result in a good flight behavior: the drone either did not respond promptly or well to the commands, or it over-responded and lost the target before further corrections could be applied. A practical solution to this issue was to use for both the yaw and distance controllers tripositional control laws (bipositional above some magnitude threshold, plus a zero level in-between). Therefore, the linear and angular control setpoints to be applied are computed as follows for the rotation and distance correction:

$$\omega = \begin{cases} -0.1, \text{if } c < -\bar{c} \\ 0, \text{if } -\bar{c} \le c \le \bar{c} \\ 0.1, \text{if } c > \bar{c} \end{cases}$$

$$V = \begin{cases} -0.1, \text{if } d_x < d_{ref} - \bar{d} \\ 0, \text{if } d_{ref} - \bar{d} \le d_x \le d_{ref} + \bar{d} \\ 0.1, \text{if } d_x > d_{ref} + \bar{d} \end{cases}$$

where c is the position of the box center relative to the center of the image, normalized to $[-0.5, 0.5]$ over the image width, and the thresholds are $\bar{c} = 0.15, \bar{d} = 0.6\text{m}$. The setpoint values are given in normalized units, as required by the AR.Drone 2.0 firmware.

5.2 High-level strategy

The high-level behavior of the drone is implemented in the form of a state machine. The states and the algorithm for switching between them are presented in Fig. 4. Assuming that the initialization and takeoff activities of the drone have been carried out successfully, the state machine enters the *Follow target* loop. In this state, the UAV is mostly in a hovering mode, however the yaw and distance controllers do make position adjustments based on bounding box estimates received from the person detector, by following the control strategy presented above. When no bounding box is received, e.g. due to transient network losses, the position error is considered to be 0 and the drone does not exit the *Hover* state.

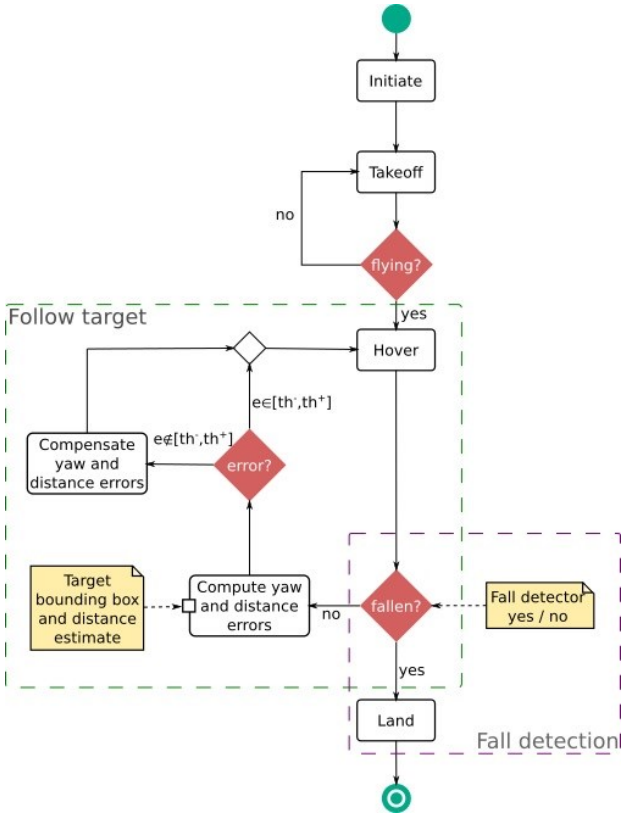

Fig. 4. The state-machine algorithm used to fly the drone

At each loop iteration, the controller also parses the input stream coming from the *Fall detector*. When a *fallen* message is received, the drone proceeds with a landing procedure. Our demonstrator implements no special action here, but this condition could be used to e.g. alert a caretaker or deliver first-aid medication.

5.3 Results

In our practical experiments, the drone generally behaved appropriately, responding correctly to the given commands and flying without losing its target, with some limited resilience to occlusions (see the demo video below).

Fig. 5 illustrates how the distance controller works to maintain the reference distance of 4 meters. In the experiment, the target was first at 6 meters from the drone, before the controller started to send correction commands. After the drone correctly approached the target and reached a relatively stable hover, the person approached the drone. This resulted in the drone moving backwards in order to maintain the reference distance.

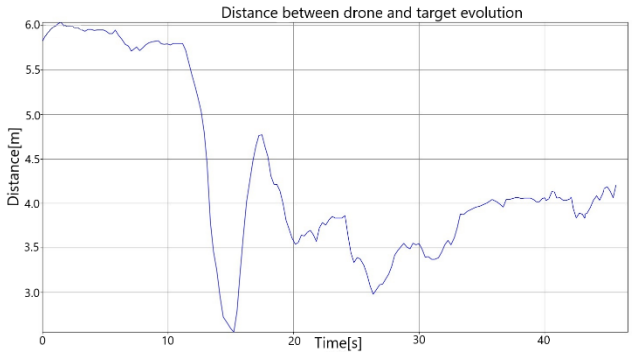

Fig. 5. Evolution of distance between drone and target during a control experiment

Fig. 6 shows, for the same experiment, the evolution of the target center coordinate, in pixels along the horizontal dimension of the image. The center of the image corresponds to 300 pixels. The drone uses yaw rotation to maintain target center at the center of the image. Note that during the experiment the person was actually slightly moving sideways with respect to the drone.

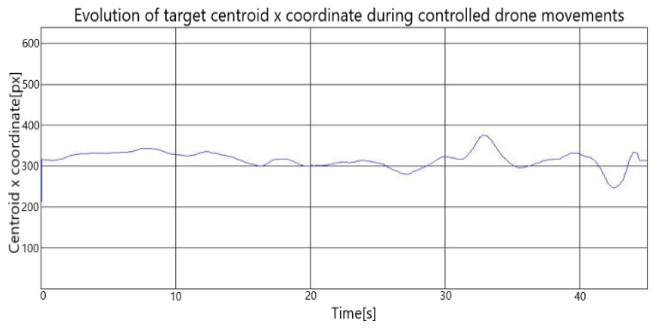

Fig. 6. Evolution of target center coordinate

A video of a practical demonstration is available at http://rocon.utcluj.ro/files/aufdemo.mp4. Fig. 7 shows a few representative video stills, each including on the left the perspective of the drone, where the detected bounding box around the person is shown as a rectangle and its centre is shown as a disk. To the right of each still, the perspective of the monitored person and a third-person view are shown. The three stills illustrate, in order: the normal situation where the person is at the reference distance; the controller in action when the person is moving; and a successful fall detection event (in which case the bounding box becomes red).

Fig. 7. Practical demonstration

6. CONCLUSIONS

In this paper we presented a first application for monitoring and detecting falls of at-risk (e.g. elderly) persons using a UAV. The position and state (upright or fallen) of the person are determined with deep-learning-based vision methods, and a simple control strategy keeps the person in view of the drone and maintains a set distance between the two. In experiments, falls were reliably detected, and the algorithm was able to correct the position of the drone so as to follow the person.

This application is a proof of concept and many elements can be improved. On the vision side, open issues include e.g. obstacle detection, explicit handling of occlusions, and robustness to multiple persons in the scene. At least as important is the control strategy, where better controllers should provide increased performance and a smoother behavior of the drone; here it will be important to address effects due to the wireless communication network, using techniques from networked control systems. On the implementation side, merging into a single network the two networks currently responsible for detecting respectively the position and the state of the person would lead to computational savings for the GPU or CPU. The final application objective is to have the entire sensing and control pipeline run on board of the drone, for which a different drone with stronger on-board processors is needed.

ACKNOWLEDGEMENT

This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-III-P1-1.1-TE-2016-0670.

REFERENCES

Andriluka, M. et al. (2010). "Vision based victim detection from unmanned aerial vehicles", *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Baer, M., Tilliette, M-A., Jeleff, A., Ozguler, A., and Loeb, T. (2014). Assisting older people: From robots to drones, *Gerontechnology* 13(1):57-58.

Boucher, P. et al. (2013). Design and validation of an intelligent wheelchair towards a clinically-functional outcome, *Journal of NeuroEngineering and Rehabilitation* 10(58):1-16.

Bourke, A.L., O'Brien, J.V., Lyons, and G.M. (2007), Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm, *Gait & Posture* 26(2): 194-199.

Cucchiara, R., Prati, A. and Vezzani, R. (2007) A multi-camera vision system for fall detection and alarm generation, *Expert Systems* 24(5): 334-345.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., and Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge (VOC2012) Results.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*.

Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing systems*, 1097-1105.

de Lima, A.L.S. et al. (2017), Freezing of gait and fall detection in Parkinson's disease using wearable sensors: a systematic review, *Journal of Neurology* 264(8): 1642-1654.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., and Zitnick, C.L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*.

Mathe, K., Busoniu, L. (2015) Vision and Control for UAVs: A Survey of General Methods and of Inexpensive Platforms for Infrastructure Inspection, *Sensors* 15(7): 14887-14916.

Murphy, R. (2014). *Disaster Robotics*, MIT Press, 2014.

Nunez-Marcos, A., Azkune, G., and Arganda-Carreras, I. (2017). Vision-based fall detection with convolutional neural networks. In *Wireless Communications and Mobile Computing*.

Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Computer Vision and Pattern Recognition*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 91-99.

Ristic, B., Arulampalam, S., Gordon, N. (2004) *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3): 211-252.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Skubic, M. et al. (2016). Testing non-wearable fall detection methods in the homes of older adults, *IEEE Annual International Conference on Engineering in Medicine and Biology Society*.

WHO (2007). *Global report on fall prevention in older age*, World Health Organization.