Stereo vision based 3D positioning for real-time control applications

Ivan Sládek
University of Žilina
Research centre
Univerzitná 8215/1, 010 26 Žilina, Slovak republic
ivan.sladek@uniza.sk

Gabriel Gašpar
University of Žilina / Faculty of Materials Science and Technology in Trnava
Research centre / Slovak University of Technology in Bratislava
Univerzitná 8215/1, 010 26 Žilina, Slovak republic / Jána Bottu 25, 917 24 Trnava, Slovak republic
gabriel.gaspar@uniza.sk / gabriel.gaspar@stuba.sk

Juraj Repka
Faculty of Materials Science and Technology in Trnava
Slovak University of Technology in Bratislava
Jána Bottu 25, 917 24 Trnava, Slovak republic
xrepkaj@stuba.sk

Roman Budjač
University of Žilina
Research centre
Univerzitná 8215/1, 010 26 Žilina, Slovak republic
roman.budjac@uniza.sk

Abstract. *Humans observe the world predominantly through vision, naturally perceiving depth and distance. This ability is thanks to binocular vision, where our two eyes collaborate to form a single, three-dimensional image. This principle is also applied in computers through a technology known as stereo vision. Stereo vision is a pivotal technology with applications across various fields, including medicine, entertainment, robotics, and industry. This paper uses this innovative technology to enhance gaming experiences and human-machine interactions. By integrating stereo vision with deep learning techniques for object detection, we estimate the 3D positions of detected objects and utilize this information to control specific aspects of computer games.*
Keywords. Stereo Vision, Object Detection, 3D Position Estimation, Game Controller

1 Introduction

Humans observe the world primarily through their eyes, which possess the natural ability to perceive the depth of objects. This depth perception allows us to determine whether something is near or far. This ability, known as binocular vision, relies on our two eyes to create a three-dimensional image. Inspired by this biological phenomenon, computer vision employs a similar approach called stereovision. Stereovision enables machines to interpret 3D space with applications across various fields [1, 2]. In medicine, it aids in performing precise surgical procedures [3]. In entertainment, it allows for the creation of immersive 3D movies; in robotics, it facilitates navigation and manipulation; and in industry, it enhances automated quality control. As technology advances, stereovision continues integrating into more segments of our lives.

Gaming has also become a significant part of daily life for many individuals, providing a means to relax and socialize. While gaming can benefit mental health, it often negatively impacts physical health due to the sedentary nature of most modern games. This paper addresses this issue by exploring using the human body as a game controller. By integrating stereovision with deep learning techniques, we propose a system for real-time human tracking. This system detects objects in images, estimates their 3D positions, and uses this information to control various aspects of computer games, leading to more immersive and interactive ex-

periences. Additionally, we propose a cost-effective stereovision solution, addressing the high expense of traditional systems.

Our motivation for this project began last year when we aimed to create an engaging project to attract the broad public into scientific projects. It led to the developing of a game controller project based on augmented reality. The controller captured images with an ESP32-CAM camera, which were then processed by a server running a program that detected characters in the image and recorded their positions on the X and Y axes. Such coordinates were used in an application environment, such as controlling the keyboard or character movement in a game. We tested this system during our faculty's open days, and it was well-received by students. However, the system had several issues, including problems with motion recognition in images with multiple people, low FPS causing lag, and inaccurate position tracking using only two axes.

The most prevalent issue was inaccurate tracking. This problem arose due to the camera's perception of objects at different depths. For instance, consider two actors of the same height, each on a plane with uniform characteristics in Fig. 1.

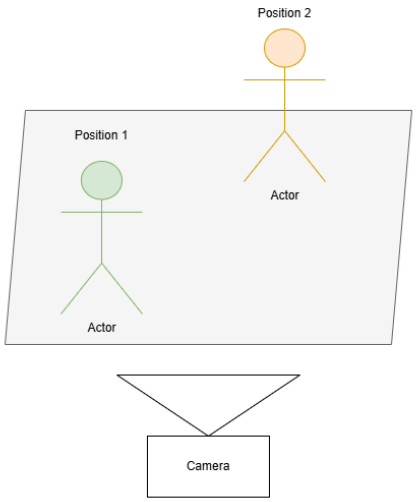

Figure 1: Real world Actors Position

For this example, we use the actors pelvises as reference points in Fig. 2. In the real world, their pelvises are at the same height (Y coordinate).

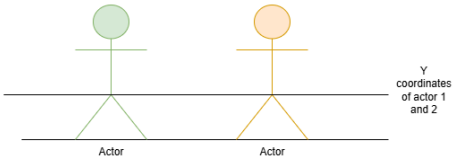

Figure 2: Height of actors pelvises

However, when viewed through the camera, the closer actor is projected at a larger scale compared to the one further away in Fig. 3. This means that in the image, the actor who is closer, but at the same height in the real world, appears higher on the image plane.

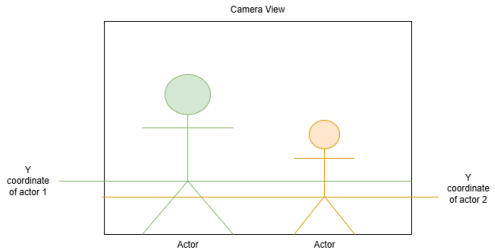

Figure 3: Real height of actors pelvises

This issue is significant because we are trying to detect when an actor jumps. This is usually done using Human Action Recognition techniques [4, 5, 6, 7]. However, these techniques are computationally demanding for our system. Therefore, in the older version of our system, we used a threshold to detect jumps. We detected persons in the image and took the middle of the bounding box of each person. We then checked if this midpoint was above a certain threshold. If it was, it meant that the person had jumped. This threshold was adaptive because not every person and environment are the same. While this approach worked, it introduced many false jump detections into our system, which was useful but not as enjoyable in the game play.

Due to these problems, we decided to develop a new version, focusing primarily on addressing the issue of incorrect tracking with only two axes.

Additionally, one of the motivations for this article was the high price of stereo cameras on the market. Stereo cameras are usually costly for regular use. However, our system is built using inexpensive webcams, improving accessibility to various applications. While these cheaper cameras are not as high-quality as the commercially available stereo systems, this limitation does not affect our use case and is suitable for character tracking.

2 Hardware and Software

In our initial attempt to build a stereo vision system, we used the ESP32-CAM module primarily for its affordability. The ESP32-CAM is a microcontroller with an integrated camera, offering both wireless and wired connectivity, which provides flexibility in our implementation. However, during testing, we encountered significant limitations. The image quality was poor, adversely affecting the performance of our deep learning model, which struggled to detect persons accurately. The frame rate was too low for real-time applications, rendering it unsuitable for our needs.

We replaced the ESP32-CAM with the Powerton Universal Webcams in Fig. 4 to address these shortcomings. This webcam offers a good balance between cost and performance, providing higher-resolution images and a better frame rate, which significantly improves the quality of our stereo vision system. The primary drawback of this webcam is that it can only be used with a wired connection.

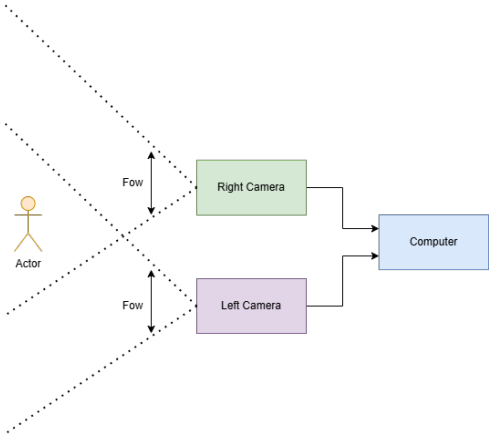

Figure 4: Stereo vision application hardware design

For the computational tasks required for image processing and deep learning algorithms, we used a notebook computer with the following specifications:

- **Processor**: Intel Core i5 - 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
- **Graphics Card**: NVIDIA GeForce MX450
- **RAM**: 16.0 GB

The processor handles data input from the cameras, while the graphics card accelerates the computations of deep learning algorithms necessary for real-time object detection.

We utilized the OpenCV library for image processing, which is widely used for computer vision applications. Next, we employed MediaPipe, a Python library designed to implement deep learning algorithms in projects for object detection within images.

3 Implementation

The implementation section details the practical steps and methodologies employed to develop the stereovision system for real-time human tracking and interaction.

3.1 Camera Mounting

We secured both Powerton Webcams to the fixed base using a strong adhesive to ensure a stable and reliable stereovision system prototype in Fig. 5. Fixing the cameras in place is essential for maintaining consistent calibration, as any movement could disrupt the alignment and accuracy of the stereovision setup. The cameras were positioned at a predetermined distance from each other and oriented parallel to ensure appropriate fields of view overlap.

Figure 5: Stereo vision application

3.2 Image Collection for Calibration

For accurate calibration, capturing images from both cameras simultaneously is crucial. This synchronization ensures that corresponding points in each image pair are captured under identical conditions. We collected a series of images from each camera, capturing various objects and scenes at different distances and angles. In these images, we used a chessboard pattern that provides well-defined corners essential for calibration. These images served as the dataset for calibrating the stereovision system.

3.3 Camera Calibration

We used a series of collected images in Fig. 6, containing a chessboard pattern to perform camera calibration. The goal was to detect the chessboard corners in these images using the OpenCV library. Identifying these corners in both the left and right camera images is essential to ensure accurate calibration. The detected corner points from both images were then used as reference points for calculating the camera's intrinsic pa-

rameters. These parameters are crucial for accurately aligning the stereo-vision system and ensuring precise depth perception.

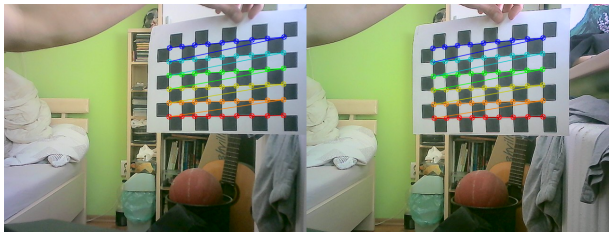

Figure 6: Left and Right image corner detection

Intrinsic calibration of a camera determines internal parameters that affect how images are captured. These parameters include the focal length, optical center, and lens distortion coefficients. We calculated these parameters using OpenCV functions and optimized them to reduce distortion and enhance image quality. This optimization process helps minimize the effect of lens distortion and improve the accuracy of the camera parameters.

Stereo calibration ensures that the spatial relationships between cameras are accurately determined and their images are aligned to enable depth perception [8, 9, 10]. This process involves determining the relative positions and orientations of two cameras, computing the rotation and translation vectors, and the essential and fundamental matrices, as shown in Fig. 7. These matrices describe the geometric relationships between corresponding points in the two images, which are necessary for accurate 3D reconstruction.

Using the parameters obtained from stereo calibration, we perform stereo rectification. Stereo rectification aligns the image planes of the two cameras, making subsequent depth estimation more straightforward and more accurate. This process involves computing rectification maps for both cameras, which are then stored for future use.

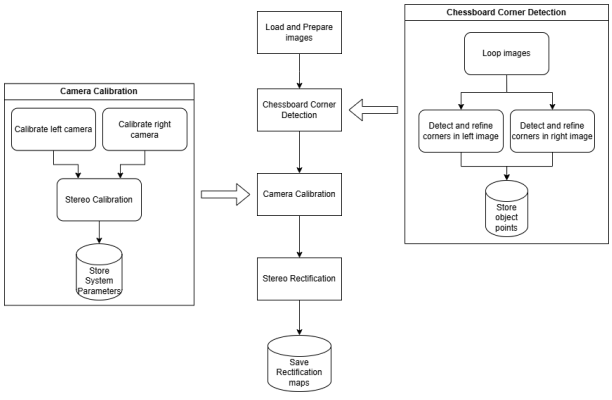

Figure 7: Camera Calibration Diagram

3.4 3D Tracking

For 3D tracking, we utilize a live stream of images from our stereo vision system, simultaneously capturing left and right images, as displayed in Fig. 8. We cannot use the raw images directly for depth estimation due to inherent lens distortions and misalignments. These distortions can significantly affect the accuracy of the depth calculation. Therefore, we apply stereo rectification and undistortion techniques using precomputed stereo maps. These stereo maps, calculated as described in Section 3.3, are essential for correcting lens distortions and aligning the images from both cameras. The rectification process involves remapping the pixels of the images to a common plane, ensuring that corresponding points in the left and right images are horizontally aligned. This alignment is crucial for accurate depth measurement. By applying the stereo maps to every pair of images in the live stream, we ensure that each frame is corrected for any distortions, providing a reliable basis for subsequent depth calculations. We use the deep learning library MediaPipe to detect and track person landmarks. MediaPipe is a versatile framework designed for building machine learning pipelines. It provides a robust solution for detecting and tracking person landmarks in real time. Once the hand landmarks are detected in both images, we can calculate the 3D position of these landmarks using triangulation. Triangulation involves measuring the disparity between the positions of corresponding landmarks in the left and right images [11]. The disparity is the horizontal shift between the same point in the two images, which is directly related to the depth of the point. In addition to depth, we can also compute the X and Y coordinates of the landmark in the 3D space relative to the camera's coordinate system.

3.5 Deep Learning implementation

For person detection, we utilize the MediaPipe deep learning framework, which includes multiple pipelines such as person detection, face detection, hand tracking, and pose estimation. MediaPipe enables rapid implementation of these pipelines, but other algorithms, such as Convolutional Neural Networks (CNNs), can also be used for object detection.

MediaPipe's person detection pipeline leverages advanced deep learning techniques to accurately detect and localize humans in images or video frames. The pipeline begins by preprocessing the data to ensure it meets the model's requirements. This typically involves resizing the image to a standard dimension, normalizing pixel values, and converting the image format if necessary.

The framework then employs Convolutional Neural Networks (CNNs) [12] for feature extraction. The lower layers of the CNN capture simple features such as edges and textures, while the deeper layers identify more complex patterns like shapes and object parts.

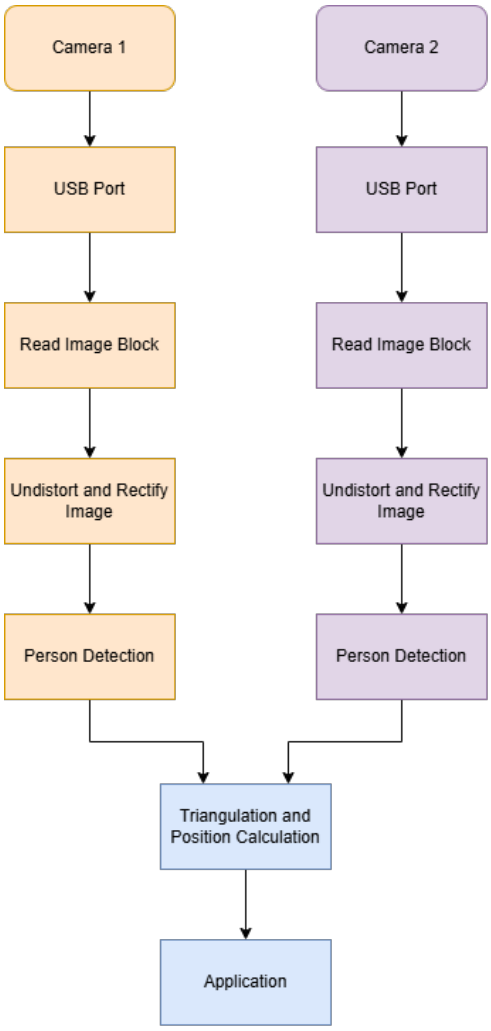

Figure 8: Tracking Diagram

These extracted features are used for the detection and localization of persons within the image.

MediaPipe utilizes networks such as Single-Shot MultiBox Detector (SSD) [13] or Region-Based Convolutional Neural Network (R-CNN) [14] for this process. These networks detect objects and draw bounding boxes around them, allowing for precise localization within the image. The person detection pipeline also includes post-processing steps, which refine the raw detection output to enhance accuracy and readability.

3.6 System Aplication

To demonstrate the application of our system, we selected the Chrome Dinosaur game. This choice stems from a previous project in which we developed a computer vision gaming controller for this game. However, the earlier controller had a limitation: it could only capture movement in the 2D plane, detecting changes along the X and Y axes. It posed a problem because our system could not accurately interpret its positional change when a detected point was closer to the camera.

Our new system overcomes this limitation by detecting objects in 3D space. This advancement ensures that the distance from the camera does not affect the object's perceived position on the X and Y axes. To simplify the process, we adjusted the system to detect the index finger in images and calculate its 3D position shown in Fig. 9. With this capability, we can accurately track the finger's position.

We implemented a threshold on the Y (height) axis to facilitate interaction as displayed in Fig. 10. The system can detect when the finger crosses the defined Y axis threshold by live tracking the finger's movement. When the finger crosses this threshold, the system simulates a press of the spacebar on the keyboard. After pressing the spacebar, the system waits for the finger to move back across the threshold before triggering another press. This mechanism enables the user to control the jumping action in the Chrome Dinosaur game. The player can effectively control the dinosaur's jumps by moving the finger up and down.

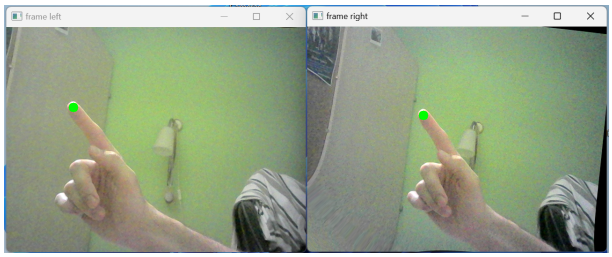

Figure 9: Detected finger Left and right image

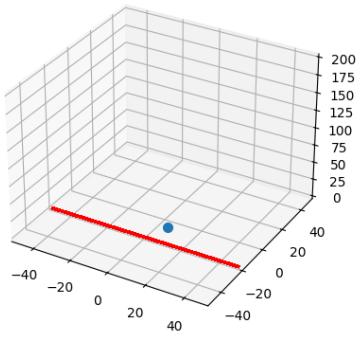

Figure 10: 3D Space with Y threshold

Compared to the first system, the new system does not require dynamic thresholding, enhancing its precision and reducing false jump click occurrence.

Additionally, our new camera system employs higher-resolution cameras, enabling the deep learning algorithm to recognize objects in images better.

The system operates at an average of 20 FPS when using a GPU. When using a CPU, the FPS drops to about 10. While both setups are playable, utilizing the

CPU results in noticeable latency, causing a slight delay between the finger crossing the threshold and the jump action. Although this latency does not render the game unplayable, it negatively impacts the gaming experience.

3.7 Performance Results

To evaluate the precision of our system, we conducted an experiment. We set detection to person detection and performed 100 jumps for each system and recorded instances where the system falsely registered a jump when no jump occurred. These 100 jumps were performed consecutively, and data was recorded in intervals of 10 jumps. The counter did not reset after each interval, but continued counting from the previous total.

We used an external button to manually count the jumps. While the system automatically counted the jumps, we manually pressed the button after each actual jump, incrementing a separate counter for real jumps. After every 10 real jumps, we calculated the difference between the system's jump count and the real jump count to determine the number of false jumps.

Jumps	**False Sys 1**	**False Sys 2**
10	4	0
20	7	0
30	8	2
40	13	3
50	14	3
60	17	5
70	22	5
80	23	5
90	27	5
100	29	6
Percentage:	**29%**	**6%**

Table 1: Comparison of False Jumps in Old System (Sys 1) and New System (Sys 2)

Table 1 shows that after 100 jumps, System 1 performed poorly, registering false jumps 29% of the time. In contrast, System 2 recorded only 6% false jumps, demonstrating a substantial improvement in performance. It's important to note that these metrics were obtained in a controlled environment and may vary in real-world situations where movements are more unpredictable.

Table 2 show a comparison of both systems. The old system is limited to 15 fps due to its camera setup, which affects its GPU performance. In contrast, the new system does not have this limitation, allowing it to achieve higher performance on a GPU.

However, the new system's performance is significantly impacted by the need to process two images simultaneously, rather than one. This impact is evident when running on a CPU, where the old system outperforms the new one. This means that if we used the old system with the new camera hardware, it would outperform the new system in terms of speed.

Hardware	**Old System FPS**	**New System FPS**
CPU	15	10
GPU	15	20

Table 2: FPS Comparison of Systems

4 Conclusion

This paper explored the application of stereo vision and deep learning techniques to enhance the gaming experience. By leveraging stereo vision combined with deep learning object detection, we developed a cost-effective system capable of tracking human movement in a 3D coordinate system.

Our approach integrates stereo vision technology with a deep learning model to detect and track person landmarks in real time. We addressed initial hardware limitations by transitioning to more suitable components, such as the Powerton Universal Webcam, which provides the image quality and frame rates necessary for accurate and responsive game control.

A comprehensive calibration process involving both intrinsic and extrinsic stereo calibrations ensures the precision of our 3D positioning system. Intrinsic calibration determines the internal parameters of each camera, while extrinsic calibration determines the relative position and orientation of the cameras, which is essential for accurate position triangulation in 3D space.

For object recognition, specifically identifying people and hands in the image, we utilized the Python library MediaPipe, which includes several deep-learning models for recognizing different objects. We applied our system to the Google Dino game, controlling a jumping dinosaur by moving a finger up and down. Our system demonstrated a significant improvement over the previous version, with fewer false jumps.

For performance comparison, we configured both systems for person detection and recorded the number of false jumps over 100 jumps. The results demonstrated that the new system is superior, achieving a 6% false jump rate compared to the old system's 29% false jump rate.

However, we observed a performance issue when using the system with a CPU instead of a GPU, resulting in slow jump responses. This problem is mitigated when running the system on a GPU.

Acknowledgments

This work was supported by VEGA through the Research on inertial data analysis methods for ap-

plications in rehabilitation adjuvants under Grant 1/0095/24.

References

E. Dandil and K. K. Çevik, "Computer Vision Based Distance Measurement System using Stereo Camera View," *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Ankara, Turkey, 2019, pp. 1-4, doi: 10.1109/ISMSIT.2019.8932817.

C. Loop and Zhengyou Zhang, "Computing rectifying homographies for stereo vision," *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, Fort Collins, CO, USA, 1999, pp. 125-131 Vol. 1, doi: 10.1109/CVPR.1999.786928.

Suenaga, H., Tran, H.H., Liao, H. et al. Vision-based markerless registration using stereo vision and an augmented reality surgical navigation system: a pilot study. BMC Med Imaging 15, 51 (2015). https://doi.org/10.1186/s12880-015-0089-5

Pham, H. H., Khoudour, L., Crouzil, A., Zegers, P., Velastin, S. A. (2022). Video-based Human Action Recognition using Deep Learning: A Review. arXiv preprint arXiv:2208.03775. Retrieved from https://arxiv.org/abs/2208.03775

Kulbacki, M., Segen, J., Chaczko, Z., Rozenblit, J. W., Kulbacki, M., Klempous, R., Wojciechowski, K. (2023). Intelligent Video Analytics for Human Action Recognition: The State of Knowledge. Sensors, 23(9), 4258. https://doi.org/10.3390/s23094258

Shaikh, M. B., Chai, D., Islam, S. M. S., Akhtar, N. (2024). From CNNs to Transformers in Multimodal Human Action Recognition: A Survey. ACM Transactions on Multimedia Computing, Communications, and Applications. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/3664815

Xie, Y. (2024). Deep Learning Approaches for Human Action Recognition in Video Data. arXiv preprint arXiv:2403.06810. Retrieved from https://arxiv.org/abs/2403.06810

Nico Nielsen, "Stereo Vision Depth Estimation," *GitHub repository*, 2023. Available: `https://github.com/niconielsen32/ComputerVision/tree/master/StereoVisionDepthEstimation`.

Satya Mallick, "Depth Perception Using Stereo Camera – Python C++," *LearnOpenCV*, 2023. Available: `https://learnopencv.com/depth-perception-using-stereo-camera-python-c/`.

Satya Mallick, "Stereo Vision and Depth Estimation Using OpenCV AI Kit," *LearnOpenCV*, 2023. Available: `https://learnopencv.com/stereo-vision-and-depth-estimation-using-opencv-ai-kit/`.

Hartley, R. and Zisserman, A. Multiple View Geometry in Computer Vision. 2nd ed. Cambridge University Press; (2004).

O'Shea, K. and Nash, R. An Introduction to Convolutional Neural Networks. (2015). arXiv:1511.08458. https://arxiv.org/abs/1511.08458

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A.C. SSD: Single Shot MultiBox Detector. *Lecture Notes in Computer Science*. Springer International Publishing. (2016), 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. (2014). arXiv:1311.2524. https://arxiv.org/abs/1311.2524