DOI: 10.58895/ksp/1000162754-10 erschienen in:
Proceedings. 33. Workshop Computational Intelligence - Berlin, 23. – 24. November 2023
DOI: 10.5445/KSP/1000162754 | https://www.ksp.kit.edu/site/books/e/10.5445/KSP/1000162754/

An Intelligent Camera Tracking System for Live Stage Performances

Steffen Borchers-Tigasson, Erik Rodner

Hochschule für Technik und Wirtschaft (HTW) Berlin
Wilhelminenhofstraße 75A, 12459 Berlin
E-Mail: steffen.borchers@htw-berlin.de

1 Introduction

Cameras have become increasingly ubiquitous in our daily lives, whether they are positioned in public spaces, within our households, or conveniently nestled in our pockets via smartphones. They serve diverse real-world applications, including video surveillance of human activities, observing wildlife, facilitating home care, enabling optical motion capture, and enhancing multimedia experiences. These applications typically entail a sequence of tasks, beginning with the detection of moving objects, followed by tracking and recognition. Over the past three decades, the field of computer vision has dedicated substantial research efforts to the task of detecting moving objects, resulting in a wealth of publications (cf. [8] for a review). The number of techniques dedicated to addressing scenarios involving moving cameras is steadily increasing, and this subject matter has become a significant focus for in-depth investigation, as evident from recent comprehensive reviews [14, 15, 7].

Focussing on the visual control of moving objects, several approaches have been proposed using PTZ cameras, e.g. for surveillance [2], autonomous off-road navigation and mobile robots [4], and the filming industry [6]. However, available professional systems are still very limited in functionality and flexibility.

To this end, we develop an intelligent camera tracking system suitable for theater, dancing and performances. The system consists of a remote PTZ camera, a state-of-the art real-time object detection and tracking algorithm (Yolov8), and an user interface to direct and adjust tracking. In this paper, we sketch our efforts in developing the system including hardware setup (Sect. 2), application requirements (Sect. 3), and detection and tracking algorithms (Sect. 4). We conclude with an evaluation (Sect. 5) and discussion (Sect. 6).

2 Technical Configuration

The smart camera system consists of a Panasonic AW-UE 4K camera, which allows for a pan movement denoted by x with range $-175^\circ \leq x \leq +175^\circ$, a tilt movement denoted by y with range $-30^\circ \leq x \leq +90^\circ$ and an optical zoom termed z with $1 \leq z \leq 24$. For each of the three movement coordinates, the camera allows for adjusting the movements speed in a range from $-50 \leq \{x,y,z\}_v \leq 50$. The current position and the movement commands are send and received from the camera via an http API interface. The visual data in transferred to a dedicated GPU server via an SDI cable, and the cameras are connected to the server via LAN.

The camera is mounted on a tripod and the absolute camera position remains fixed for the play.

The GPU Server (Windows 10) is equipped with a Blackmagic Capture Card to access the video signal in real time (approx. 40 frames/sec). The server features a NVIDIA RTX Titan (24 GB memory) graphic card.

3 Requirements

In this section, we briefly outline the specific requirements for the camera system. The requirements can be structured into three main building blocks of the overall system.

3.1 Camera Control

We aim to track objects throughout the stage and over the course of a play or performance. The desired position of an object in the frame is the setpoint.

- We demand setpoint control of pan (x), tilt (y) and zoom (z), the respective setpoints are denoted by x_{SP}, y_{SP}, and z_{SP}. The pan and tilt setpoints denote the coordinates where an image object is to be situated. For zoom control, the objects size is estimated from the current frame (see perception) and compared with z_{SP}.
- Setpoint control is extended by a dead zone, i.e. if the object moves though remains within the specified dead zone, no feedback is applied.
- To ensure homogeneous control performance for close and distant objects which appear on the frame at same size, the distance of the object has to be taken into account.

3.2 Perception

Advanced perception capabilities are required for reliable tracking. A basic understanding of the scene and its actors as well as background, possible projection, and backstage people is demanded.

- To detect objects on the current frame, bounding boxes and masks are considered. A high frame rate and low total latency is desired.
- Detection classes are head, face, body. Detected objects are represented by the respective bounding boxes. These detection classes allow for different capture settings such as close-up, knee-up, whole body, and dialog capture.
- All visible persons on the stage shall be detected. Multiple actors may be present on stage. Detected objects have to be tracked and assigned with a tracking ID number.
- The objects distance is to be estimated.

- Track losses and switches, e.g. due to occlusion, have to be considered. Fallback options have to be elaborated.
- Basic association is required for the detected objects and tracks. Objects from the three detection classes shall be associated to persons.
- To re-identify a person across multiple cameras, face identification is employed.

3.3 Interface

A flexible and intuitive Human Machine Interface is required. In short, the interface allows for:

- visualizing the detections, tracks, and associations
- adjusting setpoints and the dead zone online
- changing the tracked object, optionally with smooth transition
- modifying controllers speed
- choosing fallback options
- saving, loading, and visualizing data and configuration
- accessing and controlling all (three) cameras

4 Methods

To obtain a robust, flexible, and fast smart camera tracking system incorporating the requirements outlined in Section 3, we propose the workflow depicted in Fig. 1.

Here, the iterative loop starts with a new frame delivered by the PTZ camera. The frame is processed in the perception module subsequently, where the target object is detected, tracked, and identified. The objects information is then feed into the controller module. In combination with the user interface settings, the

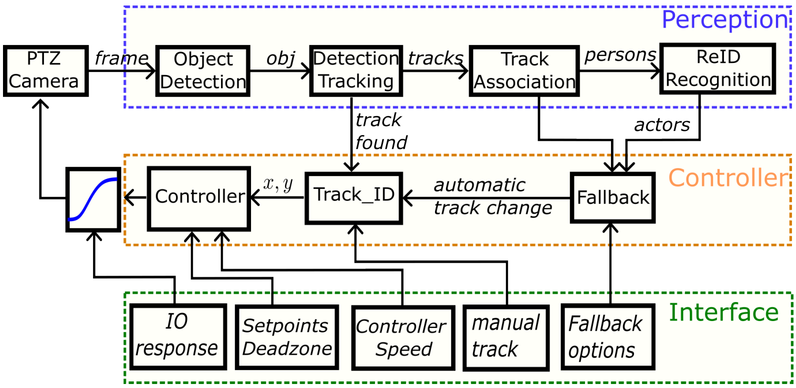


Figure 1: Key modules and the basic workflow for the proposed smart camera system.

controller computes the outputs for pan, tilt, and zoom velocities so as to close the feedback loop.

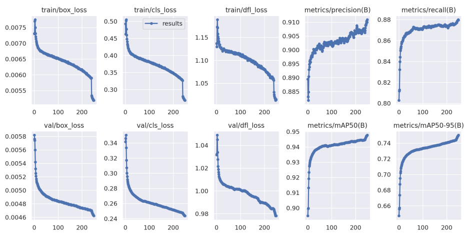
In the following, we describe the methods and approaches for the key modules seperately.

4.1 Perception Module

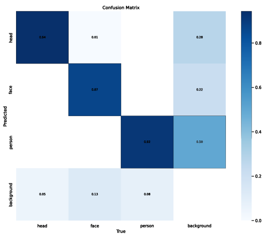
The perception module receives an image, detects all relevant objects in this image, tracks the objects from frame to frame, associates tracks with persons, and identifies them if possible.

4.1.1 Object Detection

To detect objects, namely persons, on the image, a custom model for YOLOv8 [1] has been developed. The model yields bounding boxes for three classes: head, face, and body. For training the model, several datasets such as Hollywood Heads [9], CrowdHuman [10], Facenet [11], and COCO [12] have been combined. Since the desired classes were not covered by all datasets, cross-inference was used to label the missing classes in each dataset.



(a) Training metrics for the custom detection model.



(b) Confusion matrix of the custom model.

Figure 2: Metrics of the custom Yolov8 Model.

The obtained dataset consisted of approx 120.000 labelled images. Using the open source tool FiftyOne [13], we excluded clones, crowds, and removed the most similar images. Furthermore, to decrease false positive detection rates, 10 % background images (no objects no labels) were added to the dataset. The training dataset finally consists of approx. 32.000 labelled images.

Training results and the confusion matrix are depicted in Fig. 2. Accuracy is very good for head (94 %), face (87 %), and body (92 %). As we could verify in various rehearsals, the model is robust as it copes very well with different light settings.

4.1.2 Object Tracking

For tracking the objects detected by our custom Yolov8 model, we used the Kalman-based tracking algorithm Bytetrack [16]. In comparison with other SOTA multi-object-trackers (see references in [16]), Bytetrack provided the best tradeoff between inference speed and accuracy for our purpose of tracking actors on stage.

4.1.3 Association and Recognition

To associate head, face, and body objects to persons, we utilize intersection over union (IoU). Thus, we compare the extend of overlap of all detected bounding boxes. The association works pairwise: If the smaller bounding box is covered by at least 90 %, we associate the two bounding boxes to the same person.

4.1.4 Fallbacks

On runtime, it happens that a track is lost, e.g. due to occlusion, turn arounds, or dancing moves where the head and face are temporarily not visible. Elaborating usefull fallback options thus is essential to ensure reliable tracking.

Basic idea of the fallback strategy is to use the associations obtained. If e.g. the head track is lost, the fallback consists of automatically switching the track to the persons face if available, otherwise to the persons body. If at some stage later a head is reassociated with currently tracked body or face, it atomatically switches back to head tracking. Note that it may be required to smoothly transit to the new setpoints.

4.2 Camera Control Module

The control module receives the location and size of the current object of interest in the image. Given the current setpoints, the controller computes the errors and a corrective feedback. To this end, we use separate PI controllers for pan, tilt, and zoom control.

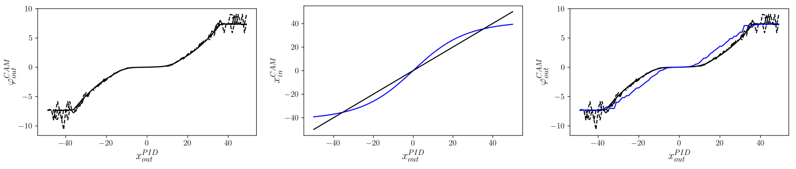
4.2.1 Output linearization

Controller output (x_{out}^{PID}) and cam output φ_{out}^{cam} are non-linearly correlated, see Fig. 3a.

The measured response Fig. 3a is corrected using a sigmoid function:

$$f_{cor}(x) = 100(\frac{1}{1+e^{-0.05x}} - 0.5)$$

(see Fig. 3b). The resulting (linearized) IO response is depicted in Fig. 3c. The response is suppressed in the range $0 \leq |x_{out}| \leq 9$, linear in the range



(a) Measure responses. (b) IO correction. (c) Corrected IO response.

Figure 3: IO Pan resonse curve and proposed correction.

$10 \leq |x_{out}| \leq 35$, and saturated for $|x_{out}| \geq 36$. The correction is applied to pan and tilt. The zoom IO response is already almost linear, so no correction is required (data not shown).

4.2.2 Distance estimation

We trained a small dense multilayer perceptron (MLP) from data acquired from the PTZ camera and laser rangefinder. The networks inputs are head width, head height, and the zoom level, the output is the distance estimate $d\ [m]$. The model provides distance predictions in real time, and validation showed acceptable performance. An analysis of it is out of scope of this paper.

4.2.3 Adaptive PID controllers

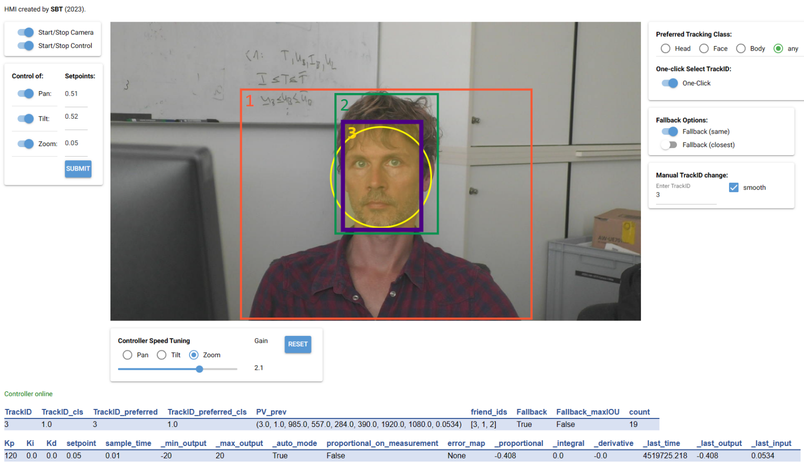
Pan, tilt, and zoom are controlled using PI controllers. The parameters have been tuned manually. The interface allows to adjust the gains online.

To compensate for a trigonometric non-linearity in pan and tilt, the camera-object distance is estimated using the trained MLP as described above. The pan and tilt controller gains are hence automatically scheduled to as:

$$K_p(d) = K_p^* \cdot \frac{3}{d},$$

where K_p^* is the presetted gain tuned for a object distance of 3 m. In the real-time setting, we filter the distance estimate of the head object averaging the last three distance estimates.

4.3 Interface module



HMI created by **SBT** (2023).

Start/Stop Camera [✓]
Start/Stop Control [✓]

Control of:	Setpoints:
Pan: [✓]	0.51
Tilt: [✓]	0.52
Zoom: [✓]	0.05

SUBMIT

Preferred Tracking Class: Head Face Body [✓ any]

One-click Select TrackID: One-Click [✓]

Fallback Options:
Fallback (same) [✓]
Fallback (closest)

Manual TrackID change:
Enter TrackID: 3
smooth [✓]

Controller Speed Tuning
Pan Tilt [✓ Zoom]
Gain 2.1
RESET

Controller online

TrackID	TrackID_cls	TrackID_preferred	TrackID_preferred_cls	PV_prev	friend_ids	Fallback	Fallback_maxIOU	count
3	1.0	3	1.0	(3.0, 1.0, 985.0, 557.0, 284.0, 390.0, 1920.0, 1080.0, 0.0534)	[3, 1, 2]	True	False	19

Kp	Ki	Kd	setpoint	sample_time	_min_output	_max_output	_auto_mode	proportional_on_measurement	error_map	_proportional	_integral	_derivative	_last_time	_last_output	_last_input
120	0.0	0.0	0.05	0.01	-20	20	True	False	None	-0.408	0.0	-0.0	4519725.218	-0.408	0.0534

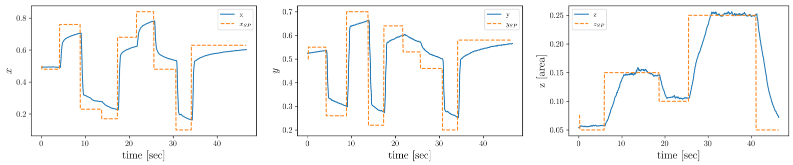
Figure 4: An overview of the interface.

The interface module is based on the python package nicegui [3]. The interface features the current image and detections, as well as interactive selection of tracks. The interface allows to turn on/off detection and the controllers, to select tracks and adjust setpoints interactively, as well as to set fallback options and tune the controller online.

5 Implementation and results

The modules are programmed using Python 3.11. Communication between the modules is established by using UDP. Associations (persons) as well as the current tuning and settings are stored using an SQLite database.

The system is currently subject to intensive tests. A current performance record is depicted in Fig. 5. Here, pan, tilt and zoom controllers are active, and control is induced by setpoint changes.



(a) Pan Control. (b) Tilt control. (c) Zoom control.

Figure 5: Control performance.

6 Discussion

This paper presents an intelligent camera system for stage performances based on single PTZ camera. The motivation behind this system is to capture close-ups and dynamic shots, allowing for projection on a screen on large stages, and accommodating hybrid formats. The system aims to provide maximum freedom of movement for actors, dancers, and musicians.

The camera system we developed here employs several perception methods based on recent advances in machine learning, first and foremost computer vision. The perception module delivers objects and its associated tracks in terms of coordinates on the current frame. Given the target location (setpoints), PI controllers stabilize the errors. Non-linearities resulting from the I/O response and from trigonometry are addressed.

Overall, the proposed system supports real-time, low-latency, and multi-camera setups. We handle approx. 40 frames per second, and the overall latency is approx. 80 ms. The system implementation utilizes APIs, deep learning frameworks, and interprocess communication for camera control, perception, and interface functionalities. Overall, this intelligent camera system offers an innovative solution for capturing stage performances with high precision, adaptability, and real-time capacity.

References

[1] G. Jocher et al. “Ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation”. Zenodo c7.0 doi:10.5281/zenodo.7347926 https://doi.org/10.5281/zenodo.7347926 2022.

[2] F. Z. Qureshi and D. Terzopoulos. “Surveillance in virtual reality: System design and multi-camera control.” IEEE Conference on Computer Vision and Pattern Recognition. IEEE 2007.

[3] F. Schindler and R. Trappe “NiceGUI: Web-based user interfaces with Python. The nice way.” https://github.com/zauberzeug/nicegui 2023.

[4] A. Hussein et al. “Autonomous off-road navigation using stereo-vision and laser-rangefinder fusion for outdoor obstacles detection.” in IEEE Intelligent Vehicles Symposium (IV). IEEE 2016.

[5] C. Ding et al. “Collaborative sensing in a distributed PTZ camera network.” in IEEE Transactions on Image Processing 21.7: 3282-3295 2012.

[6] J. Chen and P. Carr. “Autonomous camera systems: A survey.” in Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence 2014.

[7] AS Olagok and H. Ibrahim, and SS Teoh. “Literature survey on multi-camera system and its application.” in IEEE Access 8 (2020): 172892-172922 2020.

[8] M. Chapel and T. Bouwmans. “Moving objects detection with a moving camera: A comprehensive review.” in Computer science review 38 (2020): 100310.

[9] T. Vu and A. Osokin, and I. Laptev. “Context-aware CNNs for person head detection.” in Proceedings of the IEEE International Conference on Computer Vision. 2015.

[10] S. Shao et al. “Crowdhuman: A benchmark for detecting human in a crowd.” in arXiv preprint:1805.00123 (2018).

[11] F. Schroff, and D. Kalenichenko, and J. Philbin. “Facenet: A unified embedding for face recognition and clustering.” in Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[12] T. Lin et al. “Microsoft coco: Common objects in context.” in Proceedings of the ECCV: 13th European Conference 2014.

[13] BE Moore and JJ Corso “FiftyOne” in GitHub https://github.com/voxel51/fiftyone 2020.

[14] M. Cristani, M. Farenzena, D. Bloisi and V. Murino “Background subtraction for automated multisensor surveillance: A comprehensive review” in EURASIP Journal on Advances in Signal Processing 2010(24).

[15] E. Komagal, B. Yogameena “Foreground segmentation with PTZ camera: a survey” in Multimedia Tools and Applications 77 (17) (2018) 22489–22542.

[16] , Y. Zhang et al. “ByteTrack: Multi-Object Tracking by Associating Every Detection Box” in Proceedings of the European Conference on Computer Vision (ECCV), 2022.