SN Computer Science (2020) 1:296
https://doi.org/10.1007/s42979-020-00304-x

ORIGINAL RESEARCH

Real-Time Processing of High-Resolution Video and 3D Model-Based Tracking for Remote Towers

Oliver J. D. Barrowclough[1] · Sverre Briseid[1] · Georg Muntingh[1] · Torbjørn Viksand[1]

Received: 5 May 2020 / Accepted: 14 August 2020 / Published online: 8 September 2020
© The Author(s) 2020

Abstract
During the past decade, a new approach to providing air traffic services to airports from a remote location has been established, known as remote or digital tower. High quality video data is a core component in remote tower operations as it inherently contains a huge amount of information on which a controller can base decisions. The total resolution of a typical remote tower setup often exceeds 25 million RGB pixels and is captured at 30 frames per second or more. It is thus a challenge to efficiently process all the data in such a way as to provide relevant real-time enhancements to the controller. In this paper we describe the development of number of improvements and discuss how they can be implemented efficiently on a single workstation by decoupling processes, implementing attention mechanisms and utilizing hardware for parallel computing.

Keywords Remote tower · Object tracking · Machine learning · Video processing · 3d Modelling

Introduction

The remote tower concept, proposed in the early part of this century, aims to provide air traffic services (ATS) to airports from a remote location by streaming sensor data over a network [21, 29]. High-resolution video that replicates the out-the-window view of a conventional tower is a core feature of the remote tower concept and is supplemented by a pan-tilt-zoom (PTZ) camera that mimics the function of binoculars in a conventional tower [8]. Other components in remote tower typically include voice communication, meteorological data, sound reproduction and radar data, where available. The motivation behind the concept is both to cut costs associated with tower maintenance and personnel, and to improve safety. Today there is a focus on validating the multiple remote tower concept, in which an air traffic controller (ATCO) is responsible for more than one airport simultaneously [16, 22]. This movement of the concept towards more complex scenarios demands new technologies that support an ATCO's situational awareness in order to ensure that their cognitive capacities are not exceeded. Such technologies may include providing visual enhancements or even fully automating parts of the ATCO's responsibilities. This requires processing of large amounts of data, including high-resolution video data, in real-time. In this work we consider up to 14 statically positioned cameras, each with resolution 1080x1920 aligned in portrait mode and stacked horizontally, for a total resolution of 15120x1920. Despite the ever-increasing capacity of modern computers, their computational power is still insufficient to allow for naive implementations when dealing with such large amounts of data. There are, however, a number of solutions, both hardware and software based, that make such processing attainable.

Our contribution in this paper is to describe the implementation of several features that enhance remote tower based on raw video data. In particular, novelties of this paper include:

– Introduction of a method for cost-effective surface tracking of objects that does not rely on object movements or active surveillance hardware, such as radar.

✉ Oliver J. D. Barrowclough
oliver.barrowclough@sintef.no

Sverre Briseid
sverre.briseid@sintef.no

Georg Muntingh
georg.muntingh@sintef.no

Torbjørn Viksand
torbjorn.viksand@sintef.no

1 SINTEF Digital, Forskningsveien 1, 0373 Oslo, Norway

- Combination of state-of-the-art object detection algorithms (YOLO) with attention mechanisms to handle high-throughput and extremely high-resolution video data.
- Consolidation of each (per frame) detection into (cross frame) tracks using combinatorial optimization.
- Design of the hardware and software components in order to balance the trade-off between accuracy and real-time performance.
- Development of an automatic real-time video exposure and white balance correction for a camera array that does not require overlap between adjacent cameras.
- Development of a software architecture in which the various components utilize parallel processes on a single workstation and communicate in such a way that failure of a non-vital component (e.g. tracking) does not affect the vital ones (i.e., video rendering).

The organization of this paper is as follows. In Sect. 2 we outline the background with respect to video processing and tracking technologies. In Sect. 3 we detail our approach to providing real-time tracking and exposure correction to very high-resolution video and describe the system as a whole. In Sect. 4 we provide both quantitative and qualitative results to evaluate our approach. Finally, we provide a summary and some concluding remarks in Sect. 5.

Background

Literature Review

The literature on the development of the remote tower concept appears to be relatively sparse. The concept seems to have gained traction during the early part of this century with initial proof-of-concept being performed in the US by NASA [21] and by DLR in Europe [8, 10, 29]. This was followed by a number of research projects that brought the concept forward to the first operational tower in Örnsköldsvik, Sweden in 2014 by LFV and Saab. An overview of the development of the remote tower concept and technologies can be found in [9].

In the more general context of video processing and tracking for surveillance systems, there exist a number of relevant sources. An early approach to real-time vehicle tracking using conventional computer vision techniques was developed by Coifman et al. [4]. Ozkurt & Camci apply video processing for estimating traffic density using neural networks, but stop short of tracking [19]. Biswas et al. present an approach to intelligent traffic monitoring by combing several sensor data sources [2], though the hardware is limited to microcontrollers. A plethora of modern approaches visual object tracking were presented during the Seventh Visual Object Tracking VOT2019 Challenge [11], but these approaches target tracking of generic objects. A more general overview of relevant methods and applications can be found in [6, 7]. Common to the approaches above is that they mainly focus on single-camera setups, where the total resolution is low compared to our scenario.

The closest work we have found to our research is the masters thesis of Mathiesen [18]. This research was performed in parallel with ours, and considers a similar dataset and is trained specifically on airport data. Nevertheless, that work omits reference to attention mechanisms and does not look into exposure correction in the context of image stitching.

In the following subsections we provide some more specific background on the topics of image/video stitching and tracking.

Video Processing

Live streamed video is a core component of remote tower systems, typically using light spectrum, but sometimes also infrared imaging for support in low visibility conditions. Given that the cameras are often placed some distance from the runway, high-resolution of the streamed imagery is considered an essential feature. Remote tower data is most often transferred on high bandwidth networks meaning that, in many cases, high frame rates are available in addition to high resolution. Essentially, this means that a huge amount of information is being continually updated at a fast pace.

Another feature of remote tower implementations is that they often use multiple cameras to cover angles of up to 360 degrees. The exposure of each of these cameras is typically controlled individually in order to ensure that they present optimal contrast of the scene to the ATCO. For example, if the sun is shining directly towards one camera, the exposure profile required should be completely different to the profile of the camera pointing in the opposite direction. Nevertheless, this local correction of contrast often leads to visible 'seams' between the images when presented side-by-side (see Fig. 1). It is therefore of interest to apply local filters that smoothly adjust the contrast to avoid such seams appearing as prominently (stitching).

The aim of a stitching algorithm is to produce a visually plausible mosaic, in the sense that it is as similar as possible to the input images, but in which the seam between the stitched images is invisible [15].

There exist many methods for stitching panorama images and thus reducing the visible seam if the exposures do not match up. One example of such methods that has shown good results is Laplacian pyramid blending [3] using a feathered blend. However, common to many of these methods in a panorama setting is that they operate on adjacent images with a region of overlap. In our scenario, we do

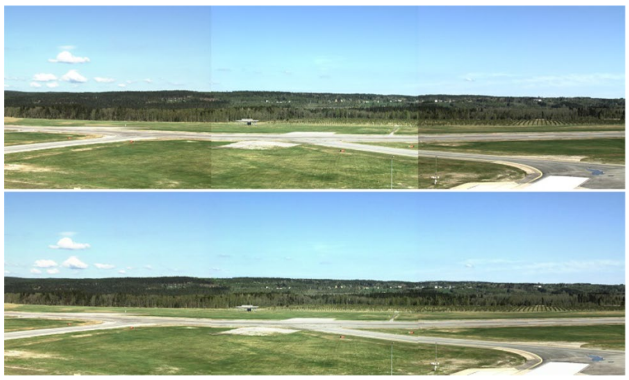


Fig. 1 Original (up) and exposure-corrected (down) images from the remote tower at Sundsvall-Timrå airport. Images courtesy LVF and Saab. This figure is not included in the article's Creative Commons licence

not have such an overlap as the cameras are intended to be perfectly aligned. In [20] the authors describe stitching of non-overlapping images. They use a pyramid blending with a Savitzky–Golay filter to smooth the transition across the seam. The input images are unrelated and the result is a blurred area across the seam, which is visually pleasing but not what we aim for. Another method is gradient domain blending [15], where the method performs the operations in the gradient domain. The authors discuss two main image stitching methods. The optimal seam algorithm involves searching for a specific curve in the overlap region, thus not applicable in our scenario. The second method is relevant for our setup, using a minimization over the seam artifacts by smoothing the transition between the images. It is, however, too compute-intensive to be run for every frame with our setup of up to 14 Full HD cameras.

Tracking

Tracking an object in a video scene is based on repeated detection and localization of the object on successive frames and obtaining a continuous association between detections through time. This association is made easier if a classification of the object is available.

In recent years, machine learning algorithms have made great strides in performance, both with respect to computational efficiency and accuracy of the results. This has resulted in some tasks that were previously considered to be insurmountable, now exhibiting superhuman performance. In particular, this is true for:

– detection—deciding whether an image contains an object;
– classification—determining the class of the dominating object in an image (in our case aircraft, vehicle, or person);
– localization—estimating the location of an object (in our case as a tight bounding box);
– tracking—locating a moving object over time.

In the context of remote tower, these tasks are of particular interest as they reflect some of the responsibilities of the ATCO.

In this paper we make extensive use of object detection as the basis for our tracking algorithm. Object detection is a wide field of research and there exist a large number of approaches dating back several decades. Traditional approaches for real-time object tracking include methods such as mean shift [5]. The interest in object detection has exploded recently, driven by developments in deep learning and convolutional neural networks (CNNs) [1, 17, 23, 24, 26]. In principle, our tracking method is agnostic to the choice of object detection method, and inherits the properties of the chosen method in terms of efficiency and accuracy. Thus, we opt for the method that provides a good balance between these requirements, with a large weight on efficiency, due to the high-resolution of the data. There exist a number of metrics for comparing approaches to object detection such as average precision (AP), mean average precision (mAP) and many variations of these. Despite, criticism of how fair these metrics actually are [24], they provide a sufficient basis for deciding which approach to use for our work. Figures 1 and 3 of [24] show a recent comparison of several methods for object detection using the AP and COCO mAP-50 metrics respectively, plotted against inference time. Due to its vastly superior performance with respect to inference time and its competitive performance with respect to AP/COCO mAP-50, we select the version of YOLO (You Only Look Once) presented in [24] as our method of choice.

YOLO frames the problem of detection and localization of objects in a scene as a regression problem that can be solved with a single evaluation of a neural network. This approach provides both better performance and is faster to evaluate than most of its predecessors. In addition to detecting and localizing objects, YOLO also provides a classification of the object and an estimated probability that the classification is correct.

In order to obtain a complete 3D surface tracking solution, we combine the video tracking with depth information obtained from calibrated 3D models of the airport terrain. Calibration involves determining both the intrinsic and extrinsic parameters of the camera. There exists a great deal of literature on calibration of cameras in a general setting, see e.g., [25, 30]. In our setting, the cameras are static, several parameters are known and the lenses have negligible distortion. There is, however, a requirement for high accuracy, particularly with respect to the camera orientation.

Methodology

Video Processing

White Balance and Exposure Correction

For a smooth transition between camera frames, narrow bands on each side of a seam should have close to identical colour spectra. Our fundamental assumption is the converse: If we are able to accurately match the colour spectra of adjacent narrow bands, then the transition will appear natural. We determined this assumption to be reasonable, as we expect the landscape to be approximately identical in these regions. This local constancy prior holds as long as the video streams are well aligned geometrically and temporally. The video processing is performed for the synchronized frames.

To obtain a smooth transition, we estimate a shared spectrum at the seams based on averaging the intensity distributions in narrow bands along the seam (see Fig. 3). To flawlessly map from the measured spectra to the shared spectra would require a highly nonlinear and high-dimensional map. However, this would take too much time to compute, as well as being too slow to apply in real-time. Moreover, the measured spectra are only approximations of the underlying spectrum of the landscape, so a simpler mapping correctly matching the essential features of these distributions would be more appropriate and prevent overfitting.

GPUs are well known for their ability to swiftly apply linear operations. To keep our algorithm as efficient as possible, we consider an approach that only depends on adjacent video streams. Considering the number of cameras in our setup, such a local approach yields a significant reduction in complexity. A further reduction is achieved by matching the spectra using a linear affine map $f_c(x) = a_c x + b_c$ for each individual colour channel $c = r, g, b$. These maps were applied from the middle of the adjacent images towards the common border (see Fig. 3). A gradual transition was obtained using convex combinations, from applying the identity map to the center of the image to the map f_c on the border.

It turned out these maps were not expressive enough to accurately transform all dominant features in the spectra. For instance, in certain cases we obtained a map yielding a seamless transition in the sky but a poor transition on the ground (and vice versa). To overcome this issue, we decided to partition the stream vertically in blocks of identical size. Another issue, manifesting itself as a local flickering, appeared when an object moved from one stream to the next. In this case the pixels of this object suddenly outshine the pixels of the background landscape, dominating the colour spectrum and violating our fundamental assumption. To resolve this issue we implemented two supplemental methods. The first method detects the movement using the thresholded absolute differences between frames (see Fig. 5), removing the corresponding pixels from consideration in the measured spectra when defining our exposure correction map. This object removal approach is viable for objects that do not dominate the domain of the local exposure correction map.

Should a moving object cover most or the whole block, there will be few or no pixels left for defining our map. For these cases we use an exponential smoothing approach, which reduces the contribution of the moving object by blending the newly computed exposure function with the exposure function from the previous frame

$$E_{\text{blend}} := (1 - \alpha) \cdot E_{\text{prev}} + \alpha \cdot E_{\text{new}}, \qquad \alpha = 0.05.$$

AI-Based Video Tracking

Detection

In this work we use YOLO for detection, localization and classification [23, 24]. YOLO is a convolutional neural network that extracts and uses the same features for classification and localization, in the form of multiple bounding box predictions. This makes the method both extremely fast and accurate, due to better generalization achieved by this multitask learning. The third version of YOLO (YOLOv3) is essentially an extension of previous versions, where 53 convolutional layers are used (applying successive 3×3 and 1×1 filters), and are organised in 23 successive residual blocks. The final residual block is followed by an average pooling before a fully connected layer and a softmax output. We refer to [24] for full details of the architecture.

Each detection consists of an object category (aircraft, vehicle, or person), axis-aligned bounding box, as well as a probability signifying the confidence of the detection. The three object classes were consistently colour-coded in their appearance as bounding boxes and close-ups in the video streams and as markers on the map (see Fig. 2). To avoid false positives, only detections whose probability exceeds a threshold $tol_{\text{detect}} = 0.65$ are processed.

In some cases, multiple detections returned by the YOLO architecture could correspond to the same object. Such superfluous detections are eliminated using greedy non-maximum suppression [28] as follows: for each category, pick the detection with highest probability, and suppress

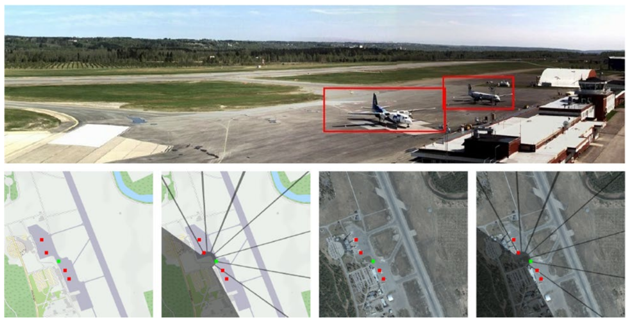


Fig. 2 Tracked objects with close-up views and their position in maps overlays. Camera views courtesy LFV, maps © OpenStreetMap CC BY-SA, orthophotos © Lantmäteriet. This figure is not included in the article's Creative Commons licence

overlapping detections within this category by setting their probabilities to zero. This process is then repeated for the remaining detections, until only detections with zero probability remain.

Overlapping of bounding boxes B, B' is quantified in terms of their Intersection over Union (IoU), defined as the quotient of the areas of their intersection and union, i.e.,

$$\mathrm{IoU}(B, B') := \frac{\mathrm{area}(B \cap B')}{\mathrm{area}(B \cup B')}.$$

It measures similarity of the boxes, taking value 0 for disjoint boxes, value 1 for identical boxes, and otherwise values in between. Overlapping detections are then suppressed whenever their IoU exceeds a threshold $tol_{\mathrm{NMS}} = 0.45$. However, YOLO cannot be applied directly to our situation, as it applies to square input images of fixed size. The image obtained by concatenating n video streams is not square; it has size $(n \cdot 1080) \times 1920$. Moreover, our high-end consumer grade GPU (GTX 1080 Ti, with 11Gb) runs out of memory, even when attempting to run YOLO on a 1600×1600 subimage.

While memory is not an issue for running YOLO on an image of size 960×960, the entire visual range can only be scanned once every couple of seconds in this manner. For the high spatial and temporal resolution of our setup, it is therefore important to develop effective attention mechanisms, i.e., strategies for deciding where to look.

Attention Mechanisms

We consider the following three mechanisms:

1. *Sliding window approach*. After concatenating the frames of all cameras in a single image, slide a fixed-sized window across this image and run a detection in each window. As an option, it is possible to use overlapping windows to avoid unfortunate cropping of objects. Another option is to (in addition) resize the image to detect objects at various scales. This strategy is computationally expensive, and therefore only run on start-up to get a good overview of the initial situation.
2. *Difference approach*. Moving objects can be detected by detecting significant local changes in the video streams. Technically, this is achieved by thresholding the absolute difference of two consecutive frames, as shown in Figure 5. Sliding a window across the resulting binary image, one runs a detection whenever the number of on-pixels (representing a significant change) exceeds a given threshold.
3. *Expectation approach*. Once we have an inventory of tracked objects with their locations and movements, we can predict its expected position in a future frame, and run a detection there.

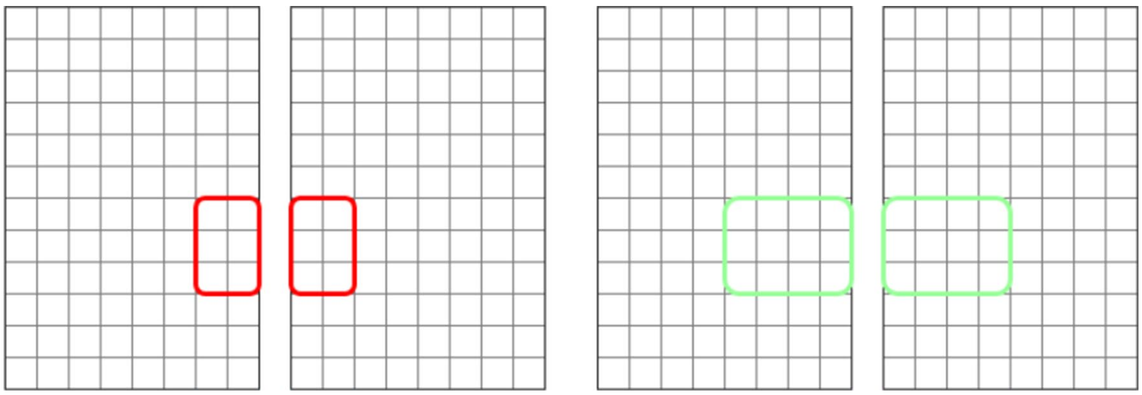


Fig. 3 The exposure correction function is defined using a narrow band along the seam of adjacent images (red) and is applied to half the image (green)

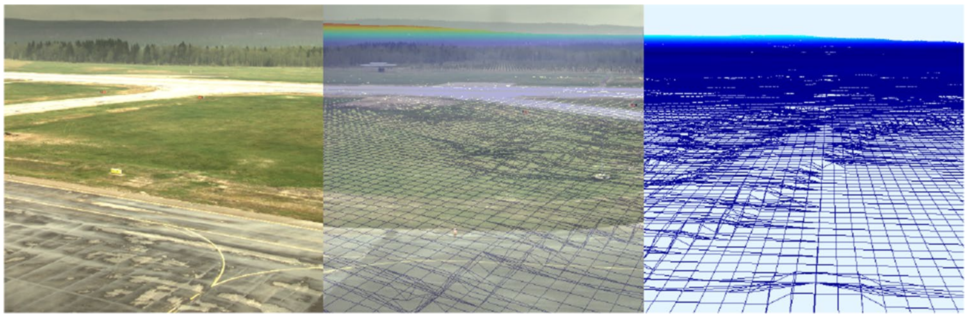


Fig. 4 The video stream (left), 3D model (right), and blend (middle) in the camera calibration application. The slight mismatch on the horizon is due to missing data at large distances. Imagery courtesy LFV. This figure is not included in the article's Creative Commons licence

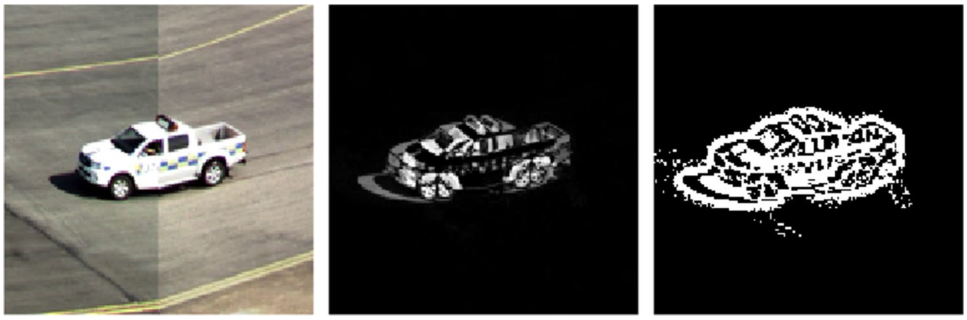


Fig. 5 In a given image region (left), changes are detected by taking the absolute difference of consecutive frames (middle) and thresholding (right). This figure is not included in the article's Creative Commons licence

These mechanisms are combined in a high-level scheduler to effectively track objects, subject to the computational resource constraints.

Tracking Algorithm

Upon start-up of the tracker, one first applies the sliding window approach to yield an initial list of detections. During the remainder of the tracking process, the difference and expectation approaches are used for deciding where to run detections. Besides being used within each YOLO detection, non-maximum suppression is used here to remove superfluous detections by the various attention mechanisms. To avoid the creation of duplicate objects (and an ensuing cascade effect), a low suppression tolerance $tol_{\text{NMS}} = 0.3$ is used here.

The problem of optimally assigning a set of m detections $D = \{d_i\}_i$ to n existing objects $O = \{o_i\}_i$ can be expressed as an *assignment problem*. For this, one first defines a cost function $C : D \times O \longrightarrow \mathbb{R}$, in which a higher cost reflects a less desirable match. The values of this function are assembled in a *cost matrix*

$$\mathbf{C} = \begin{bmatrix} C(d_1, o_1) & \cdots & C(d_1, o_n) \\ \vdots & \ddots & \vdots \\ C(d_m, o_1) & \cdots & C(d_m, o_n) \end{bmatrix} \in \mathbb{R}^{m,n}.$$

For the linear sum assignment problem, the goal is to find a one-to-one assignment $f : D \longrightarrow O$, for which the total cost

$$\sum_{i=1}^{n} C(d_i, f(d_i))$$

is minimal. This problem can be solved rapidly (in cubic running time) using the Hungarian algorithm [12]. Such an assignment problem is solved for every category separately.

Let B_d and B_o be the bounding boxes of detection d and object o measured at frame numbers f_d and f_o. To impose a penalty for dissimilarity, we consider a cost function complementary to the IoU, defined by

$$C(d, o) := 1 - \text{IoU}(B_d, B_o) \cdot a^{f_d - f_o}, \qquad a = 0.99.$$

This function imposes a higher cost for matching a detection d with an object o last observed in a distant frame, by discounting their IoU by a factor a for every frame that has since passed.

After finding the optimal assignment f, each detection d is added to the history of the object $o = f(d)$ if

$$1 - C(d, o) > tol_{\text{IoU}} := 0.05,$$

i.e., if the discounted IoU exceeds a given tolerance. If this is not the case, as well as for the unassigned detections, it is checked whether

$$1 - \min_{i=1} C(d, o_i) < tol'_{\text{IoU}} := 0.001$$

i.e., whether the detection wasn't just outmatched, but not relevant for any of the existing objects. If this is the case, it is added as a new object. This rather strict tolerance avoids the duplication of objects due to inaccurate detections.

Calibration in 3D Space

Given that the cameras are static, a number of parameters such as position could be easily determined, e.g., by referring to map data, orthophotos. The tilt and lens distortion was negligible. Aspect ratio and resolution are embedded in the image data. Thus the main challenge is determining horizontal orientation and field of view of the cameras. To aid this process, we implemented an interactive application that allows both navigation in the 3D model and blending the video dynamically in order to match features manually, see Fig. 4. After calibration, the depth of each pixel is computed by simply rendering the scene with the depth buffer active. The resulting depths can then be used to position events in 3D space.

System

The video streams enter the system as H.264 compressed video streams [27] in 1920 × 1080 resolution. In our case, 13 such streams had to be decoded and displayed in real-time. In order to achieve the required performance, we offload the decoding to the GPU using Nvidia NVDEC, which on our system with a GeForce GTX 1070 GPU was able to decode up to 14 such streams in real-time.

With the decoding being done on the GPU, and the video frames residing in GPU memory after decoding, rendering in real-time and at full resolution is easily achieved. The frames are only moved into RAM, a relatively slow operation, a few times per second in order to calculate the white balance and exposure correction on the CPU, and when the object recognition-module requests a new frame.

With the object tracking written as a separate Python application, ZeroMQ [14] is used for inter-process communication. The object detection also runs on a GPU, and since it is essential not to degrade the performance of the live video view, a separate GPU (GeForce GTX 1080 Ti) is used for this task. This also has the benefit that if the object tracking code were to experience a crash or a slow-down, it will not inhibit the operator. She will still get a live video stream while the object recognition module recovers. For a full block diagram of the application, see Fig. 6.

Results

Exposure Correction

The exposure correction predominantly yields a mosaic with natural transitions, as visualized for sunny weather conditions in Fig. 1. This is also the case for video, in the sense that also temporal changes generally seem natural. In the presence of moving objects, the method generates natural results most of the time. However, the method can struggle when moving objects cross the image seams, sometimes resulting in a local flickering. Typically, the problem is most pronounced right before and after a full crossing of the seam, i.e., when the object is fully present in the boundary band of one of the images but not in the other. Table 1 shows the results of the proposed exposure correction methods with 16 and 64 blocks vertically,

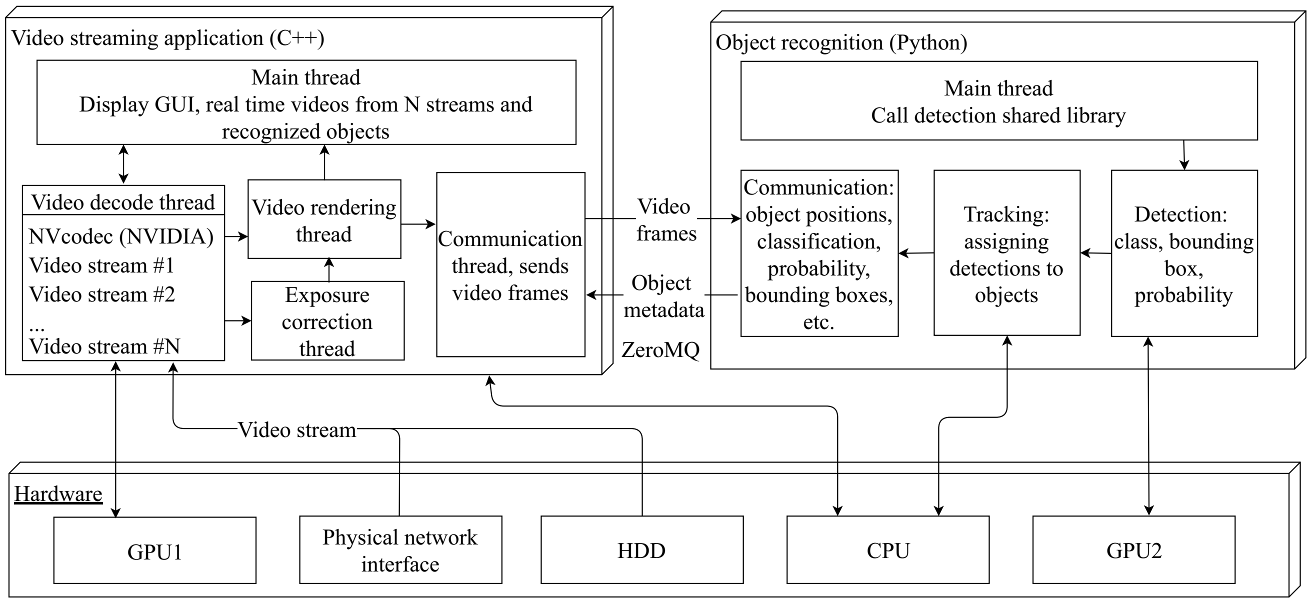
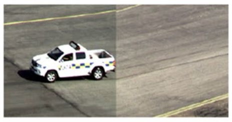
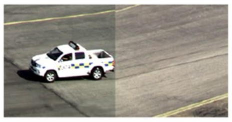
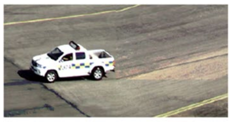
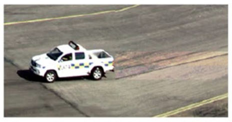
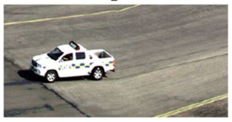
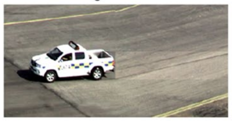
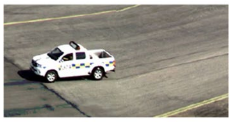
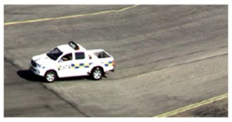

Video streaming application (C++)
Main thread
Display GUI, real time videos from N streams and recognized objects
Video decode thread
NVcodec (NVIDIA)
Video stream #1
Video stream #2
...
Video stream #N
Video rendering thread
Exposure correction thread
Communication thread, sends video frames
Video frames
Object metadata
ZeroMQ
Object recognition (Python)
Main thread
Call detection shared library
Communication: object positions, classification, probability, bounding boxes, etc.
Tracking: assigning detections to objects
Detection: class, bounding box, probability
Video stream
Hardware
GPU1
Physical network interface
HDD
CPU
GPU2

Fig. 6 Application block diagram and mapping of the software processes to different hardware components

Table 1 Exposure correction methods with various block sizes. The original concatenated image is shown twice for comparison with the correction methods. This table is not included in the article's Creative Commons licence

16 blocks	64 blocks
Original frame (duplicate)	
	
Standard exposure correction	
	
Exposure correction with object removal	
	
Exposure correction with exponential smoothing	
	

when applied to concatenated video streams with a moving object right after a full crossing of the seam. The standard exposure correction introduces a noticeable discolouration in the block next to the car, both for large and small blocks. The object removal approach shows natural results if the remaining number of pixels in the block is relatively high (left case). However, if the moving object fills most of the block (right case), too few pixels remain for computing a natural exposure correction. The exponential smoothing approach generally shows natural results. It does, however, add a slight delay to the update of the exposure correction. For this reason we prefer using the object removal approach when applicable.

Running the exposure correction on 14 HD cameras at 30 FPS simultaneously introduced only a minor overhead on the GPU (GTX 1070). For a quantitative evaluation of the performance of our exposure correction algorithm, we use a cost function based on the method described in [15]. Consider adjacent images I^- (left) and I^+ (right) of size $M \times N$ with columns $I^-_{-1}, I^-_0, I^+_0, I^+_1$ from left to right of the seam. A naive measure of continuity is to directly compare the columns at the seam, i.e., $\frac{1}{N}\sum_{i=1}^{N} \|I^-_{0,i} - I^+_{0,i}\|$, but this measure is sensitive to geometric misalignment and asymmetrical details. Instead, the trend at row i can be captured by measuring whether the gradients between the final (resp. initial) image column (r, g, b) pixels continues across the seam, i.e., whether

$$d_i^{\pm} := \|(I^{\pm}_{\pm1,i} - I^{\pm}_{0,i}) - (I^{\pm}_{0,i} - I^{\mp}_{0,i})\| \approx 0.$$

Hence the total discrepancy of the trend is

$$\frac{1}{N}\sum_{i=1}^{N} \frac{d_i^+ + d_i^-}{2}. \tag{1}$$

Table 2 Average values of the cost function 1

	Image interior	Image seam Uncorrected	Image seam Corrected
Sunny	8.62	49.82	19.75
Cloudy	6.37	61.19	16.11

Table 2 shows the average value of (1) for two scenes with 6 cameras and a duration of 60 seconds. The reference value in the left columns was evaluated at the middle of the input streams. The exposure corrected result is a significant improvement over the initial uncorrected seam, but is still significantly higher than the reference value. This deviation can partly be explained by a slight geometric misalignment at the seam.

Camera Calibration

Manually tuning several parameters (position, view direction, field of view, etc.) for aligning the 3D model to the video streams is a demanding process. It typically involves a field trip, expensive measuring equipment, and it can take several person-days for obtaining an accurate result. On the other hand, the camera calibration application we developed provides a virtual environment in which the calibration can be performed, requiring only video/image data and a digital surface model.

Conclusion

In this paper, we have developed and tested a number of techniques based on video processing, 3D modelling and object tracking that apply to high-resolution video arising from remote towers. It was shown that the methods can be implemented on a single workstation with commodity hardware and still retain real-time performance by efficiently exploiting hardware resources and by reducing unnecessary computations.

There remain a number of limitations of the approach which should be resolved in future work. One limitation is that this work has focused on developing a real-time system for tracking rather than optimizing the reliability of the tracking capability. In future work we would like to compare how the method performs in relation to existing approaches to tracking in terms of accuracy, precision and detection failure. Another limitation is that the bounding boxes of static objects tend to exhibit an unstable 'twitching' behaviour, which would be distracting for an ATCO in an operational setting. This instability may be down to the nature of convolutional neural networks, and could potentially be resolved with post-processing. Additionally, the tracking functionality has so far only been tested in relatively high visibility conditions during daytime. More testing is needed to see how reliable the tracking is in low visibility and night-time scenarios. One approach to improving performance in this case could be to train the network specifically on airport data gathered under various lighting and weather conditions. The attention mechanisms considered in this paper are based on detecting movements and expected locations of existing tracked objects. In the future, we could also consider where ATCOs concentrate their attention, by looking at heat maps from tracked eye movements [16] to attain better performance. Testing using infrared sensors is also subject to future work. A limitation of the exposure correction maps is that they currently act on each colour channel separately. The quality of the corrections can be expected to improve when using linear maps combining the three channels, at a negligible computational overhead. Moreover, currently the exposure correction map is defined separately for each vertical block. The transition between these maps could be improved by either using convex combinations of the adjacent maps or by adding boundary conditions. Finally, more tuning is needed for automatically selecting which of the proposed methods to use to best avoid flickering artifacts.

In a wider context, there are concerns about the use of neural networks in safety critical operations, due both to their 'black-box' nature [18] and their susceptibility to adversarial examples [13]. Whether the results of this paper can be brought forward to an industrial implementation may depend on further developments on the explainability of neural networks and methods to defend against adversarial attacks.

Acknowledgements The authors would like to thank Saab and Luftfartsverket Sweden (LFV) for their assistance in providing access to the remote tower video data from Sundsvall, Sweden. We also express our gratitude to the four ATCOs from the COOPANS partnership for their valuable feedback.

Funding Open Access funding provided by SINTEF AS. This project has received funding from the SESAR Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement No 730195.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Preprint A preprint of this manuscript is available from https://arxiv.org/abs/1910.03517.

References

1. Bertinetto L, Valmadre J, Henriques JF, Vedaldi A, Torr PH. Fully-convolutional Siamese networks for object tracking. In: European conference on computer vision, pp. 850–865. Springer; 2016.
2. Biswas SP, Roy P, Patra N, Mukherjee A, Dey N. Intelligent traffic monitoring system. In: Proceedings of the second international conference on computer and communication technologies, pp. 535–545. Springer; 2016.
3. Burt PJ, Adelson EH. A multiresolution spline with applications to image mosaics. ACM Trans Graph (TOG). 1983;2(4):217–36.
4. Coifman B, Beymer D, McLauchlan P, Malik J. A real-time computer vision system for vehicle tracking and traffic surveillance. Transport Res Part C Emerg Technol. 1998;6(4):271–88.
5. Comaniciu D, Ramesh V, Meer P. Real-time tracking of non-rigid objects using mean shift. In: Proceedings IEEE conference on computer vision and pattern recognition. CVPR 2000 (Cat. No. PR00662), vol. 2, pp. 142–149. IEEE; 2000.
6. Dey N, Ashour A, Acharjee S. Applied video processing in surveillance and monitoring systems. USA: IGI Global; 2016.
7. Dey N, Ashour A, Patra PK. Feature detectors and motion detection in video processing. USA: IGI Global; 2016.
8. Eier D, Huber H, Kampichler W. Advanced ground surveillance for remote tower. In: 2008 Integrated communications, navigation and surveillance conference, pp. 1–9. IEEE; 2008.
9. Fürstenau N. Virtual and remote control tower. Switzerland: Springer; 2016.
10. Fürstenau N, Rudolph M, Schmidt M, Werther B. Wettbewerb der visionen 2001–2004. DLR: Tech. rep; 2004.
11. Kristan M, Matas J, Leonardis A, Felsberg M, Pflugfelder R, Kamarainen JK, Cehovin Zajc L, Drbohlav O, Lukezic A, Berg

A, et al. The seventh visual object tracking VOT2019 challenge results. In: Proceedings of the IEEE international conference on computer vision workshops; 2019.
12. Kuhn HW. The Hungarian Method for the assignment problem. Naval Res Logist Q. 1955;2:83–97.
13. Kurakin A, Goodfellow I, Bengio S. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533; 2016.
14. Lauener J, Sliwinski W. How to design and implement a modern communication middleware based on ZeroMQ. In: Proceedings, 16th international conference on accelerator and large experimental physics control systems (ICALEPCS 2017); 2018.
15. Levin A, Zomet A, Peleg S, Weiss Y. Seamless image stitching in the gradient domain. In: Eight European conference on computer vision (ECCV 2004), pp. 377–389; 2004.
16. Li WC, Kearney P, Braithwaite G, Lin JJ. How much is too much on monitoring tasks? Visual scan patterns of single air traffic controller performing multiple remote tower operations. Int J Ind Ergon. 2018;67:135–44.
17. Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988; 2017.
18. Mathiesen J. Low-latency detection and tracking of aircraft in very high-resolution video feeds. : Master's thesis, Linköping University; 2018.
19. Ozkurt C, Camci F. Automatic traffic density estimation and vehicle classification for traffic surveillance systems using neural networks. Math Comput Appl. 2009;14(3):187–96.
20. Pandey A, Pati UC. A novel technique for non-overlapping image mosaicing based on pyramid method. In: Proceedings of annual IEEE India conference (INDICON), pp. 1–6; 2013.
21. Papasin R, Gawdiak Y, Maluf DA, Leidich C, Tran PB. Airport remote tower sensor systems. NASA Ames Research Center: Tech. Rep. SAE-2001-01-2651; 2001.
22. Papenfuss A, Friedrich M. Head up only—a design concept to enable multiple remote tower operations. In: 2016 IEEE/AIAA 35th digital avionics systems conference (DASC), pp. 1–10. IEEE; 2016.
23. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788; 2016.
24. Redmon J, Farhadi A. YOLOv3: an incremental improvement. arXiv preprint; 2018. arXiv:1804.02767.
25. Remondino F, Fraser C. Digital camera calibration methods: considerations and comparisons. Int Arch Photogramm Remote Sens. 2006;36(5):266–72.
26. Ren S, He K, Girshick R, Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp. 91–99; 2015.
27. Richardson IE. The H.264 advanced video compression standard. 2nd ed. UK: Wiley Publishing; 2010.
28. Rothe R, Guillaumin M, Gool LV. Non-maximum suppression for object detection by passing messages between windows. In: Cremers D, Reid I, Saito H, Yang MH, editors. Asian conference on computer vision (ACCV). Cham: Springer; 2014. p. 290–306.
29. Schmidt M, Rudolph M, Werther B, Fürstenau N. Remote airport tower operation with augmented vision video panorama hmi. In: Proceedings 2nd international conference on research in air traffic management (ICRAT 2006), pp. 221–230. DS Public, doo, Belgrade; 2006.
30. Zhang Z. A flexible new technique for camera calibration. IEEE Trans Pattern Anal Mach Intell. 2000;22(11):1330–4.