Digital Twin Office for Workspace Throughput Monitoring

M.O. Candela-Leal
Tecnológico de Monterrey
Monterrey, México
milton.candela@tec.mx

M.A. Ramirez-Moreno
Tecnológico de Monterrey
Monterrey, México
mauricio.ramirezm@tec.mx

J.J. Lozoya-Santos
Tecnológico de Monterrey
Monterrey, México
jorge.lozoya@tec.mx

***Abstract*—Efficient usage of spaces and tools in a given workspace is a crucial component when maximizing throughput in an input-process-output environment; the current work presents an integrative visualization of a shared workspace with an Internet of Things (IoT) infrastructure via the design, creation, and real-time implementation of an integrative system that consists on three modules: orders reported by employees on Bitrix24, a Customer Relationship Management (CRM); machine's state and operations done according to Q-lab, an Application Programming Interface (API); employees' actions captured by Closed-Circuit Television (CCTV) cameras, processed via a Computer Vision (CV) algorithm with a Deep Learning (DL) model with an architecture of a Convolutional Neural Network (CNN). With 2D video recording of over five cameras and four actions to identify: (open machine, compact machine, change piece, add piece), the model should take a considerable amount of time given the complexity of the situation, as it takes a sequence of images as input to predict an action. However, the model would be fast when predicting an activity, given by automatic real-time video segmentation when CCTV detects a person, the conjunction of each of the systems, would then create a real-time dashboard that shows the machines in terms of usage, time of operation remaining, conflicts between data sources, and actions done by employees. This constant monitoring of a workspace, leveraged by intelligent technologies to create a Digital Twin (DT) of an office, would function as a Decision Support Tool (DSS) to assure machines' maximum throughput.**

***Index Terms*—Digital Twin, Computer Vision, IoT, Real-time Simulation, Deep Learning, CNN, Throughput, HAR**

I. Introduction

A. General context

Human Activity Recognition (HAR) is a widely complex task that has been not deeply explored [1], it commonly takes two approaches: Sensor-based (gyroscope, pressure sensors, depth-based, and hybrid, modality systems) [2] or vision-based (RBG, RBG-D) frame analysis using Deep Learning (DL) [3]. The current approach considers video-based HAR, as it uses CCTV footage from the laboratory to identify worker's actions.

Deep Learning models, [4] created a RNN + CNN model trained on 128 hours of untrimmed real-world surveillance recordings achieved 97.23% accuracy to distinguish abnormalities. On the other hand, in [5] human behaviour recognition in dynamic, static, and transition actions using accelerometer and a gyroscope using smartphones.

B. Delimitation of the object of study

Using only Bitrix24, CCTV footage from the company and Q-Lab API, to obtain a real-time dashboard that refreshes on client's needs and orders, leveraged by depth cameras to create a DT workspace in order to appreciate better the interconnectivity between machines and workers.

C. Problem Statement

Chemical machines (Q-Lab) in a given lab are needed to process some parts delivered by clients, in this case, the objective is to maximize machines' throughout using intelligent systems and biometrics in order to get a sense about machines' state and how they could be optimized in real-time given DSS of dashboard and DT.

D. Justification

Yearly analysis are being performed by the company, nevertheless, real-time decision making is poorly leveraged by intelligent devices and hence is not fully optimized on data. It is important to not only take decisions, but take informed decisions based on data, although, as machines require long hours to process pieces, and their size is limited, the task of fully optimizing the procedure gain complexity.

E. Theoretical framework

1) Digital Twin: A Digital Twin is a simulation of a space or object of interest, which receives enormous inflow of data in order to update the model that is being simulated, it reassembles real-life scenarios and are useful to run tests on that workspace and so compare how it could change the real workspace, it could also be used to have a better appreciation of the workspace constantly fed with data.

2) Computer Vision: Computer Vision (CV) refers to the action of Artificial Intelligence (AI) to predict based on images or video. It could be employed to: Measure biomechanical features from person, detect whether a person is in a video or not, detect certain objects in a given frame, and even measure risk when used in autonomous devices.

3) Human Activity Recognition: Already discussed in I-A.

F. Objectives

To maximize machines' throughput using data and a real-time dashboard, and provide a framework of IoT devices interconnected in an integrative system, which shouws a real-time DT of the workspaces in order to used it as a DSS.

G. Hypothesis

Real-time inflow of data from intelligent devices, is capable of maximizing throughput and reduce time used in a piece. It would also increase business' income, as it is capable of processing more client's orders easily and faster.

II. Proposal

A. Methodology

1) Data Auto-segmentation: Based on You Only Look Once (YOLO) CV algorithm, a data auto-segmentation was designed in order to cut the CCTV videos only when a person is being detected on the CCTV footage. This script was designed in order to make real-time predictions only when people were on the CCTV footage, but also to reduce the pre-process time, as mini-clips of people performing actions are needed to train the HAR DL model, and the auto-segmentation video obtains the data easier.

2) Deep Learning Architecture: For the deep learning architecture, it would use ConvLSTM cells in order to

Long Short-Term Memory (LSTM) cells have a different architecture than a simple cell, its structure in Fig. 1, and the set of equations required in Eq. (1). Three gates compose the cell: Forget gate, input gate, and output gate.

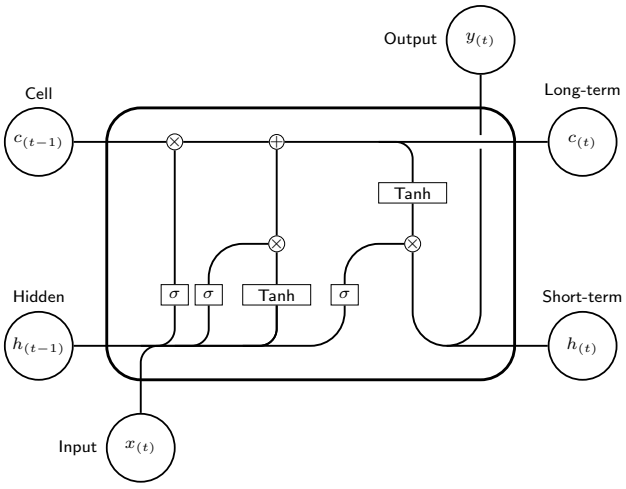


Fig. 1: Default structure of a LSTM cell, based on [6]. It has a combination of $\sigma(x)$ and $\tanh(x)$ functions, to create three main gates (forget, input, output) and so generate the short-term $h_{(t)}$ and long-term $c_{(t)}$ vectors.

The processing starts on the lower part with $x_{(t)}$ and the previously generated $h_{(t-1)}$, they are used to calculate $f(t)$ (left-hand side), via the $\sigma(x)$ function in Eq. (1b), this controls the forget gate to determine which information of the long-term state should be erased, as a $\sigma(x)$ function is used and their values range from 0 (erase all) to 1 (erase nothing). On the other hand, $x_{(t)}$ and $h_{(t-1)}$ are also used to calculate $i_{(t)}$ and $g_{(t)}$. This consists on the input gate, where the $\sigma(x)$ function in $i_{(t)}$ controls the information of $g_{(t)}$ that should be included in the long-term state. The addition of the forget gate and input gate is calculated in $c_{(t)}$, as shown in Eq. (1e), Which is further used in the output gate with $o_{(t)}$, which controls the information from the long-term state should be read and outputted. The element-wise multiplication of both $o_{(t)}$ and $c_{(t)}$ is the output $y_{(t)}$, which is equal to $h_{(t)}$ (used in the next iteration as $h_{(t-1)}$), as calculated in Eq. (1f).

$$i_{(t)} = \sigma(W_{xi}^T x_{(t)} + W_{hi}^T h_{(t-1)} + b_i) \tag{1a}$$
$$f_{(t)} = \sigma(W_{xf}^T x_{(t)} + W_{hf}^T h_{(t-1)} + b_f) \tag{1b}$$
$$o_{(t)} = \sigma(W_{xo}^T x_{(t)} + W_{ho}^T h_{(t-1)} + b_o) \tag{1c}$$
$$g_{(t)} = \tanh\left(W_{xg}^T x_{(t)} + W_{hg}^T h_{(t-1)} + b_g\right) \tag{1d}$$
$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \tag{1e}$$
$$y_{(t)} = h_{(t)} = o_{(t)} \otimes \tanh\left(c_{(t)}\right) \tag{1f}$$

TABLE I: In each RNN architecture, for each model that has a specific type of layer, the output shape would be used for each force prediction on a given dataset, where $n_{seq} = 5$ and $n_{feat} = 88$.

n_{layer}	Layer	Parameters	Activation function
1	Input	(n_{seq}, height, width, 3)	
2	CNN 2D	16, ($3x3$)	ReLU
3	MP 2D	($4x4$)	
4	Dropout	0.25	
5	CNN 2D	32, ($3x3$)	ReLU
6	MP 2D	($4x4$)	
7	Dropout	0.25	
8	CNN 2D	64, ($3x3$)	ReLU
9	MP 2D	($2x2$)	
10	Dropout	0.25	
11	CNN 2D	64, ($3x3$)	ReLU
12	MP 2D	($2x2$)	
13	Flatten		
14	LSTM	32	
15	Dense	k	Softmax

3) Activation and Loss Functions: In order to measure performance on continuous variable predictions, a set of evaluation metrics are used as loss functions so that the DL-based model's performance could be evaluated and hence its weights updated. In this sense, a cost function compares model's predictions and the actual data; in a regression model, the function computes the distance between reference and predicted values [6]. The objective is to minimize the evaluation metrics, as they measure the difference between the reference values and their predictions, therefore predicting values similar to the reference values.

The only loss function used is softmax activation function, as in [6], where k is the number of total classes.

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{2}$$

Moving on to loss function, the loss function used is categorical cross-entropy, based on softmax activation function described previously in Eq. 2, where k is the number of total classes.

$$CE = -\sum_{i=1}^{k} t_i \log s(x_i) \tag{3}$$

4) Performance Metrics: A set of performance metrics would be used to evaluate model's performance, these include classification (for categorical target variables) metrics.

The first set of performance metrics correspond to classification problems: Accuracy, Precision, Recall and F1-Score. Although, additional metrics should be first described, using the confusion matrix in Table II. Depending on the combination of predicted label and true label, the classification would be categorized in one of the confusion's matrix quadrants, either TP, FN, FP or TN.

TABLE II: Confusion Matrix that describe components depending on combinations of predicted and true label.

		Predicted label	
		Positive	Negative
True label	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Moving on to our first classification metric is accuracy, which is considered as the average correct classification percentage. This metric could be represented as the number of predicted labels that match the correct classification divided by the total number of samples. In this case, based on the confusion matrix, this would be represented as the sum of TP and TN divided by the total possibilities N, which is the sum of all classifications in Eq. 4.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Additionally, precision is a classification metric that measures the number of TP classifications with respect to FP, and so determines which of the positive predicted labels are actually positive by the true label, this metric was calculated as shown in Eq. 5.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

On the other hand, recall is quite similar to precision, although this metric measures the number of TP classifications with respect to FN, and so penalizes when the models misses on predicting a positive label on a true labeled positive sample.

This metric is widely used in medicine field, as FN are dangerous when diagnosing a deadly disease, it is usually preferred to having a FP label and so do more testing in order to determine if it is TP or FP, rather than not even diagnosing it at all. The metric was calculated as shown in Eq. 6.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

This last classification metric combines both precision and recall into a single metric called F1-Score, which takes the harmonic mean of these metrics and so it is a more balanced approach with respect to using precision and recall separately. The metric was calculated as shown in Eq. 7.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{7}$$

B. Methodological proposal to use

The methodological proposal is presented in Fig. 2. The process starts with the green rectangles which represent the system's inputs; then they are processed based on each respective tool, shown with a yellow rectangle; afterwards, intermediate outputs are outputted using system's inputs and their respective tool, and shown with an orange rectangle. Each of these intermediate outputs serves as inputs on the final integrative system, which is represented with a purple rectangle; finally, the integrative system combines all intermediate outputs and develops the final outputs represented with red rectangles.

C. Techniques and technological tools used

1) Recurrent Neural Network (RNN): A type of Artificial Neural Network (ANN). Instead of having a feed-forward network (from input to output), an RNN also takes into account past output $h_{(t-1)}$ and thus is connected to the future input $x_{(t+1)}$.

The neuron is connected to itself when not considering time t, as shown in Fig. 4 on the left, which is the same structure unrolled through time on the right, where it is more apparent that output h_0 is not only the output of time 0, but it is part of the operation of time 1, this process continues until frame t, and thus an ANN can process sequences via the inclusion of past output into the next operation.

When considering Fig. 4, on the left of the figure, is the rolled through time, and on the right of the figure is the unrolled through time version, x_t represents a sample at time t, A is a neuron with a certain activation function (usually $\tanh$) that behaves like ANN's neurons, and h_t is not only the output at time t but also input on time $t + 1$, with its respective sample x_{t+1}.

2) Convolutional Neural Network (CNN): A type of ANN. They provide a more scalable approach to image classification and object recognition tasks, leveraging principles from linear algebra, specifically matrix multiplication, to identify patterns within an image.

They are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

1) Convolutional layer: A two-dimensional (2D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix;

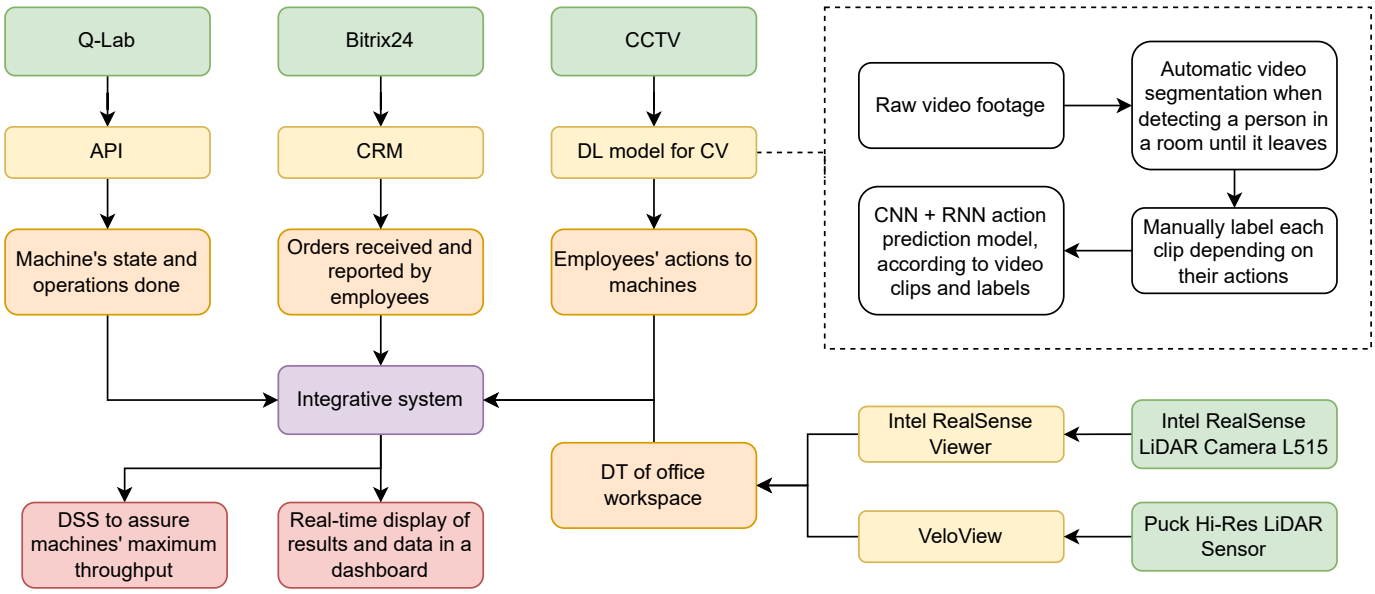


Fig. 2: Flow diagram that explains the flow and methodology followed to create the integrative system.

this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array.

2) Pooling layer: Sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array.
3) Fully-connected (FC) layer: Each node in the output layer connects directly to a node in the previous layer. Use a softmax activation function to classify inputs appropriately, producing a probability from 0 to 1.

3) You Only Look Once (YOLO): YOLO [7] is a well-renowned CV algorithm for object detection it is capable of processing frames in real-time using a CNN model and image segmentation. In this case, YOLOv4 was used to predict whether a person is in a given frame with an accuracy higher than 70, which would be used to auto-cut the CCTV footage only when a person is detected on the video.

4) VeloView: VeloView (ParaView, Kitware) is a a digital tool that performs real-time visualization and processing of live captured 3D LiDAR data from Velodyne's sensors, it can playback pre-recorded data stored in .pcap files, and can record live stream as .pcap file. The sensor sweeps an array of lasers (16, 32, 64, 128) 360° and a vertical field of view of 40°/20° with 5-20Hz and captures about a million points per second.

In fact, this technological tool was used to create Fig. 8, which was generated by moving the sensor in the configuration described in Fig. 7, during a 5-minute recording by moving cautiously a moving shelf. A person move the moving shelf, although it was important that he perform the movement while squatting, otherwise, he would block the sensor and point clouds would not be collected on the direction he is standing.

D. Infrastructure

Based on a real-time framework, to create an IoT enviroment that would leverage decision making using a DT and a dashboard for visualization of workers actions and virtual environment.

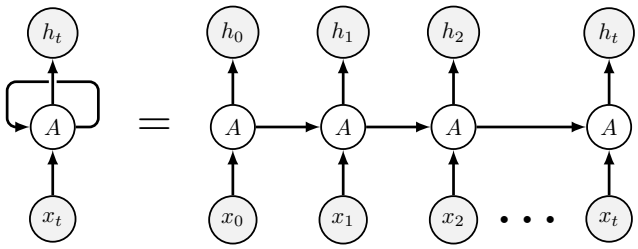


Fig. 3: The basic structure of an RNN.

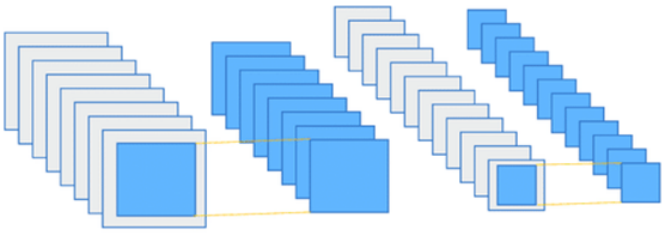
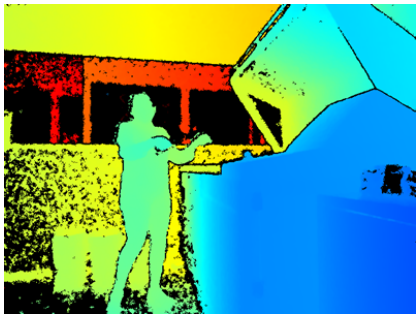


Fig. 4: The basic structure of an CNN.

(a) Normal (1920 x 1080)

(b) Infrared camera (1920 x 1080)



(c) Depth camera (1024 x 768)

Fig. 5: The Intel ®RealSense ™LiDAR Camera L515 cameras..

E. Resources used

1) Intel ®RealSense ™LiDAR Camera L515: The Intel ®RealSense ™LiDAR Camera L515 has three integrated cameras all in one in Fig. 5, it includes (a) normal high-resolution camera, (b) infrared camera with the same resolution as (a), and finally (c) depth camera with slightly less resolution when compared to the previously mentioned, but it is due to delivering depth data while keeping the same FPS.

It is a revolutionary solid state LiDAR depth camera which uses a proprietary Microelectromechanical Systems (MEMS) mirror scanning technology, enabling better laser power efficiency compared to other time-of-flight technologies. With less than 3.5W power consumption for depth streaming, the Intel RealSense LiDAR camera L515 is the world's most power efficient high-resolution LiDAR camera.

2) Puck Hi-Res Velodyne ®LiDAR: Configuration using this device is shown in Fig. 7. The Puck Hi-Res Velodyne ®Lidar is positioned on top of a moving shelf, while the laptop recording the data is positioned on the middle part of the moving shelf. The moving shelf is important due to digitization of a space requires continuous recording to create the DT.

It is designed for applications requiring greater image resolution. While retaining the Puck's surround view and best-in-class range, the Puck Hi-Res™ sensor delivers a 20° vertical field-of-view for a tighter channel distribution.

3) CCTV: The camera points to the workstation in Fig. 6. The recording is a 10-hour recording in .DAV video format, a given day at (a) morning, (b) afternoon, (c) night. It can be seen that only at (b) color is present, this might be due to lights turning full on, while (a) only has some lights turned on, it may cause errors in the HAR algorithm, so images should be pre-processed to black and white.

Fig. 7: Setup configuration.

Due to video in .DAV video format, it should be transformed to common video format (.avi or .mp4), this was accomplished using VidCoder, an open-source DVD/Blu-ray ripping and video transcoding application for Windows, which uses HandBrake as its encoding engine. The process was quite fast, as it transformed .DAV 10-hour length video file (1920x1080,

(a) 11:02 AM

(b) 2:26 PM

(c) 5:38 PM

Fig. 6: CCTV from a camera in Xperto Integral Systems.

15 FPS) to 10-hour .mp4 (640x480, 15 FPS) in 2 hours.

It is based on FFmpeg, is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge. No matter if they were designed by some standards committee, the community or a corporation. It is also highly portable: FFmpeg compiles, runs, and passes the testing infrastructure FATE across Linux, Mac OS X, Microsoft Windows, the BSDs, Solaris, etc. under a wide variety of build environments, machine architectures, and configurations.

III. Results & Conclusions

A. Results

The DT of the workspace is already created in Fig. 8, the digitization of the workspace was done moving cautiously the configuration described in Fig. 7, as well as recording via VeloView in Sect. II-C4, using Puck Hi-Res Velodyne ®LiDAR in Sect. II-E2.

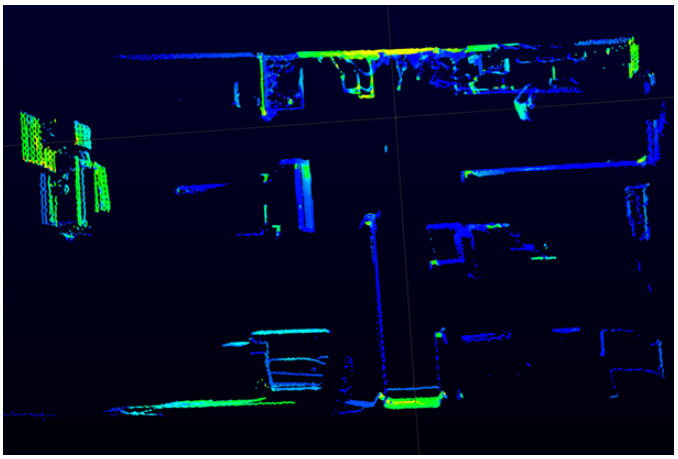


Fig. 8: 3D visualization of the working lab.

In order to speed up the process of gathering data to train the HAR model, an automatic video-segmentation algorithm was designed in order to generate video clips based on the YOLO algorithm that detects persons. The Algorithm 1 describes the function "cut_video_on_person_detected()" with auxiliar functions: "detect_person()" to assess whether a person is in a given frame leveraged by YOLOv4, and "cut_video()" to cut video using Python package *moviepy*.

The YOLOv4 algorithm in a 416x416 image, 2 GB VRAM NVIDIA GeForce RTX 3060 took approximately 10 seconds to process 20 frames, given that video Frames Per Second (FPS) is 15, then it takes approximately 10 seconds to process 1 video second, assuming linearity, an equation could be formulated to obtain the processed time given video time.

First, Eq. 8 relationship between video frames f_v as a function of video time in seconds $t_{v,s}$, this equation requires video FPS, which would be represented as α. The use of α is essential, as it a constant that relates frames to seconds by establishing Frames Per Second.

Algorithm 1 cut_video_on_person_detected(t_s, s_f, l, α)

Input t_s is the tolerance seconds when no human found.
Input s_f is the skip frames on video analysis.
Input l is the length of the video in seconds.
Input α is the video's FPS.

Require: $t_f \geq 0 \vee s_f \geq 0 \vee l \geq 0 \vee \alpha > 0$

$n_f \leftarrow \alpha \times l$ # Number of frames
$t_n \leftarrow t_s \div \frac{s_f}{\alpha}$ # Iterations tolerated
$n_d \leftarrow 0$ # Iterations no person detected
$b_f \leftarrow 0$ # Beginning frame
$e_f \leftarrow 0$ # Ending frame
for $f = 0 \rightarrow n_f$ **do**
  if $f \bmod s_f == 0$ **then**
    $p_d \leftarrow$ detect_person(f) # Person detected on f
    if p_d **then**
      $n_d \leftarrow 0$
      if $b_f == 0$ **then** # If b_f not initialized
        $b_f \leftarrow f$
      end if
    else
      $n_d \leftarrow n_d + 1$
      if $n_d > t_n$ & $b_f \neq 0$ & $e_f == 0$ **then**
        $e_f \leftarrow f$
      end if
    end if
    if $b_f \neq 0$ & $e_f \neq 0$ **then**
      cut_video(b_f, $e_f - s_f$) # Cut video on e_f, b_f
      $b_f \leftarrow 0$
      $e_f \leftarrow 0$
    end if
  end if
end for

$$f_v(t_{v,s}) = t_{v,s} \cdot \alpha \tag{8}$$

Now, the linear relationship between processing time and frames read is stated in Eq. 9, where $t_{p,s}$ refers to processing time in seconds, and f_v to video frames. Given the two points extracted (20 frames took 10 seconds to process), the equation could be simply expressed by dividing f_v by the constant 2.

$$t_{p,s}(f_v) = \frac{f_v}{2} \tag{9}$$

Moreover, frame rate expressed in video seconds in Eq. 8 could be substituted in Eq. 9 in order to find a expression that obtains processing time in seconds $t_{p,s}$ via video time in seconds $t_{v,s}$, this equation is expressed in Eq. 10

$$t_{p,s}(t_{v,s}) = \frac{t_{v,s} \cdot \alpha}{2} \tag{10}$$

Now, given a 10-hour video length, and using the Eq. 10, processing time in seconds $t_{p,s} = \frac{10 \cdot 60 \cdot 60 \cdot 15}{2} = 270,000$, now converted to processing time in hours $t_{p,h} = \frac{270,000}{60 \cdot 60} = 75$.

75 hours is quite big time to process a 10-hour video length, that number should be reduced in order to make the video

auto-segmentation practical and not wait over 3 days of non-continuous processing to generate mini video clips based on one camera recording one work day.

In order to reduce computation time, a skip frames s_f parameter would be added so not to make frame-per-frame analysis, but analyzing frames each s_f. Hence, when $s_f = 5$, then each 5 frames would be analyzed, and when $s_f = \alpha$, then the image analysis would be done on each video second.

Given that 20 frames could be analyzed in 10 seconds, then $s_f = \alpha$ would mean that $f_v = t_{v,s}$ in Eq. 9, and hence the simplified expression on per second analysis, relating s_f parameter in Eq. 11.

$$t_{p,s}(t_{v,s}) = \frac{t_{v,s}}{2 \cdot \frac{s_f}{\alpha}} \tag{11}$$

Hence, having $s_f = 75$ or 5 seconds, then 10-hour video length processing would take 1 hour to process, as $t_{p,h}(t_{v,h}) = \frac{10}{2 \cdot \frac{75}{15}} = 1$. This is a more reasonable time when expecting 10-hour length video from 5 different cameras, as it takes half the time recorded to process all the cameras and generate the mini-clips of only when a person is on the camera.

As a result, processing a 10-hour raw video would require circa 3 hours of processing time including pre-processing to convert the 1920x1080 DAV file to 640x480 MP4 using VidCoder and mini-clip generation using YOLO in Python.

B. Conclusions

More actions are still required in order to create the real-time dashboard, as the integration of the CRM and machines' API is still missing. Although, auto-segmentation video is currently running and generating video clips that would be used to train the HAR model.

Due to DL approaches requiring a lot of data in order to generalize and create a great prediction model, there is still data to be processed into the auto-segmentation auto-cut algorithm created using Python and YOLO. Afterwards, manual data labeling should be done in order to create the dataset regarding the actions to be predicted.

REFERENCES

[1] L. Minh Dang, K. Min, H. Wang, M. Jalil Piran, C. Hee Lee, and H. Moon, "Sensor-based and vision-based human activity recognition: A comprehensive survey," *Pattern Recognition*, vol. 108, p. 107561, 2020.

[2] M. Cornacchia, K. Ozcan, Y. Zheng, and S. Velipasalar, "A survey on activity detection and classification using wearable sensors," *IEEE Sensors Journal*, vol. 17, no. 2, pp. 386–403, 2017.

[3] "Rgb-d-based human motion recognition with deep learning: A survey," *Computer Vision and Image Understanding*, vol. 171, pp. 118–139, 2018.

[4] V. Singh, S. Singh, and P. Gupta, "Real-time anomaly recognition through cctv using neural networks," *Procedia Computer Science*, vol. 173, pp. 254–263, 2020. International Conference on Smart Sustainable Intelligent Computing and Applications under ICITETM2020.

[5] G. Gao, Z. Li, Z. Huan, Y. Chen, J. Liang, B. Zhou, and C. Dong, "Human behavior recognition model based on feature and classifier selection," *Sensors*, vol. 21, no. 23, 2021.

[6] A. Géron, *Hands-On Machine Learning with Scikit-Learn & TensorFlow*. O Reilly, 2nd edition ed., 2019.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.