Received: 00 Month 0000 | Revised: 00 Month 0000 | Accepted: 00 Month 0000

DOI: xxx/xxxx

ORIGINAL ARTICLE

# Panorama Construction for PTZ Camera Surveillance with the Neural Gas network

Karl Thurnhofer-Hemsi | Ezequiel López-Rubio | Enrique Domínguez | Rafael Marcos Luque-Baena | Miguel A. Molina-Cabello

Department of Computer Languages and Computer Science, University of Málaga, Bulevar Louis Pasteur, 35, 29071 Málaga, Spain

**Correspondence**
Enrique Domínguez, Department of Computer Languages and Computer Science, University of Málaga, Bulevar Louis Pasteur 35, 29071 Málaga (Spain)
Email: enriqued@lcc.uma.es

**Abstract**

The construction of a model of the background of a scene still remains as a challenging task in video surveillance systems, in particular for moving cameras. This work presents a novel approach for constructing a panoramic background model based on the Neural Gas network and a subsequent piecewise linear interpolation by Delaunay triangulation. Furthermore, an ensemble model of Neural Gas networks is also proposed. The approach can handle arbitrary camera directions and zooms for a Pan-Tilt-Zoom (PTZ) camera-based surveillance system. After testing the proposed approach on several indoor sequences, the results demonstrate that the proposed methods are effective and suitable to use for real-time video surveillance applications.

**KEYWORDS:**
PTZ cameras, panorama construction, background modeling, neural gas network

## 1 | INTRODUCTION

Pan-tilt-zoom (PTZ) cameras have become increasingly popular in monitoring public areas (Chung-Chen Chen, Yi Yao, Drira, Koschan, & Abidi 2009; Ding, Song, Morye, Farrell, & Roy-Chowdhury 2012; Konda, Conci, & De Natale 2016) due to their high mobility and zoom capability. Although omnidirectional cameras are promising candidates for monitoring multiple latent activities in the area of interest (Boult, Gao, Micheals, & Eckmann 2004), this kind of cameras have non-uniform resolution and are unable to provide close observations of particular targets.

In these cases, where PTZ cameras are needed, the combination of these two types of cameras (omnidirectional and PTZ) is proposed in order to facilitate a continuous monitoring of the whole surveillance area and detailed observations of specific targets simultaneously (C.-H. Chen et al. 2008). Nevertheless, this dual-camera system may be still an expensive and complex solution in some scenarios. For this reason, in this paper we are focusing on an active sensing approach to multiple object detection and tracking using a single PTZ camera.

For the images/videos captured by static cameras, the most common and efficient approach to moving object detection is background subtraction, that consists in maintaining an up-to-date model of the fixed background and detecting moving objects as those that deviate from such model. Compared to other approaches, such as optical flow, this approach is computationally affordable for real-time applications, is independent of moving object velocity, and is not subject to the foreground aperture problem. However, traditional background subtraction algorithms assume the cameras are static and this leads to false detection when the camera moves (Kim, Yun, Yi, Kim, & Choi 2013; Xue, Liu, Ogunmakin, Chen, & Zhang 2013). Due to this camera movement, even pixels belonging to static objects appear to move in the camera frame (called ego-motion effect).

Extensive research has been carried out regarding object detection for moving cameras. Some proposal are based on the optical flow clustering, that consists in calculating dense or sparse optical flows and clustering them to identify moving

object regions (Varcheie & Bilodeau 2011). Another methods are based on the estimation of the transformation parameters between consecutive frames (López-Rubio & López-Rubio 2015). Our approach is based on mosaicing the background (Azzari & Bevilacqua 2006; Bevilacqua & Azzari 2006), that consist in creating a mosaiced or panoramic background image and then using a background subtraction technique to extract moving object regions.

In this paper, we address the problem of moving objects detection for PTZ cameras, and propose a method based on building a panoramic background model using a type of competitive neural network. Apart of the traditional and frequently cited seminal papers related to competitive learning (Ahalt, Krishnamurthy, Chen, & Melton 1990; Uchiyama & Arbib 1994; Yair 1992), recent successful applications in the computer vision field can be found in the literature (H.-P. Chen, Shen, & Long 2016; Ozan, Kiranyaz, & Gabbouj 2016; Valente & Abrao 2016; Xie et al. 2016). In our approach, a Neural Gas network (T. Martinetz & Schulten 1991; T. M. Martinetz, Berkovich, & Schulten 1993) or an ensemble of them are used to build a panoramic background model for object detection. The Neural Gas stands as a popular unsupervised learning neural network, which is commonly employed for vector quantization purposes (Antonakakis, Dimitriadis, Papanicolaou, Zouridakis, & Zervakis 2016; Dimitriadis, Sun, Laskaris, Thakor, & Bezerianos 2016). Due to the huge volume of input data, a large number of neurons has been used and the neuron prototypes have been organized in a quad-tree in order to be quickly evaluated.

The rest of the paper is organized as follows. In section 2, a more detailed description of the proposed neural models are presented. In section 3, we present the results achieved with the implementation of the proposed approach. Finally, section 4 includes some concluding remarks.

## 2 | THE MODELS

In this section a Neural Gas network based system to learn the background of a panoramic scene from the input of a PTZ camera is designed. Furthermore, an ensemble model composed of several Neural Gas networks is also proposed. First the data acquisition procedure to transform the input video frames into input samples for the Neural Gas networks is considered (Subsection 2.1). Then the Neural Gas model is described (Subsection 2.2). After that, the ensemble model is proposed (Subsection 2.3). Finally, an interpolation procedure is designed to estimate the background from the final state of the competitive learning network, which is based on a Delaunay triangulation (Subsection 2.4).

### 2.1 | Data acquisition

The data from which we start are the acquired video frames of the PTZ camera. These are video frames with fixed width and height, and every pixel has two frame coordinates $(x, y)$. To transform these frame coordinates to another coordinate system in the panoramic image, the first step is to obtain the spherical coordinates associated to the frame coordinates and then carry out a scalar transformation to the dimensions of the panoramic image.

In order to find out required polar coordinates $(\theta, \phi)$ of an arbitrary point $A$ belonging to projection plane $(x, y)$ a pinhole camera model is used, consisting of a sphere centered on the coordinate origin and a projection plane (see Fig. 1 a), whose bounds are the width and height of the frame window in pixels: $w, h$. In this work the virtual PTZ library (G. Chen et al. 2015) is employed, so we must follow the coordinate system criteria used by that library. We start with the case of camera orientation $(pan, tilt) = (0, 0)$, which corresponds to see the positive Z axis from the origin. The general case, when $pan \neq 0, tilt \neq 0$ is obtained from the particular case by means of a coordinate system transformation.

The coordinates of vector $OA$ in coordinate system $O$ can be found as projections on the coordinate axes:

$$(vx, vy, vz) = (x - w/2, h/2 - y, r) \tag{1}$$

where $vz = r$, since points $A$ and $C$ are located on the same plane. $vx$ and $vy$ is calculated taking in account that $A$ on the projection plane is based relatively of the left top corner of the frame. Radius $r$ is calculated using the vertical field of view $(FOV)$, which is known. We only need to notice that $\widehat{FOC} = \widehat{FOV}/2$, and $FOC$ is a right triangle, so

$$\tan(\widehat{FOV}/2) = \frac{FC}{r} \Leftrightarrow r = \frac{h/2}{\tan(\widehat{FOV}/2)} \tag{2}$$

This way the camera coordinates are computed. In order to extend the particular case to the generic case and obtain the world coordinates we need to recalculate vector $OC$ from the $O$ coordinate system into the rotated coordinate system $O'$. The coordinates are found by means of multiplication by the inverse of a double rotation matrix. This matrix can be found by multiplication of two rotation matrices: around $X$ and $Z$ axes by $(pan, tilt)$:

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(tilt) & -\sin(tilt) \\ 0 & \sin(tilt) & \cos(tilt) \end{pmatrix} \cdot \begin{pmatrix} \cos(pan) & -\sin(pan) & 0 \\ \sin(pan) & \cos(pan) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$A' = (vx', vy', vz') = R^{-1} \cdot (vx, vy, vz) \tag{4}$$

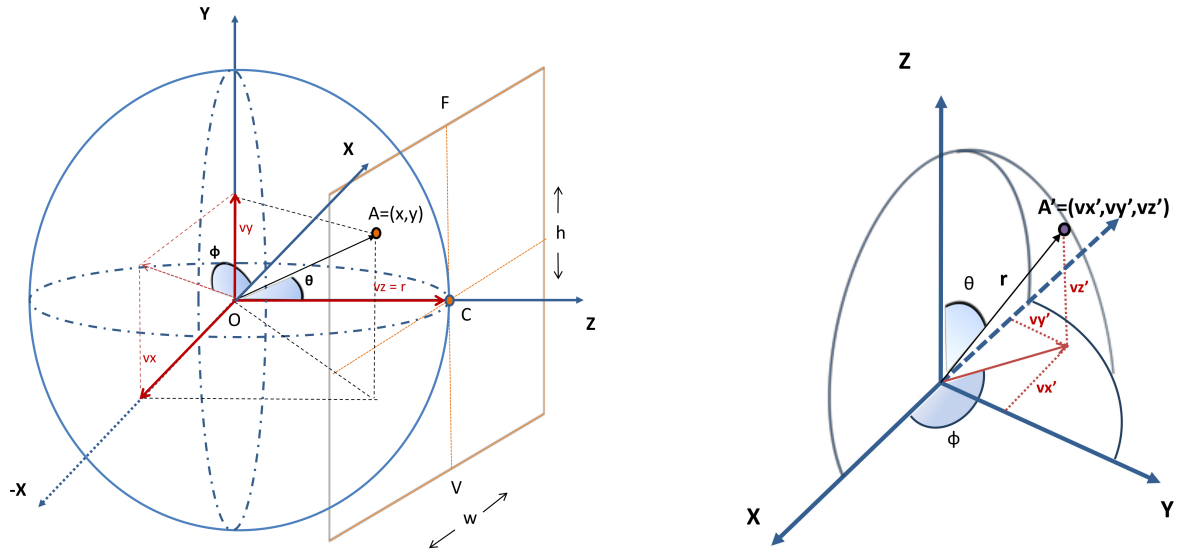



**FIGURE 1** Schematics of the PTZ camera model. a) Left: pinhole camera model with the original orientation. b) Right: reoriented scheme to obtain spherical coordinates.

Now, when the new coordinates $A'$ are computed, we only need the spherical coordinates transformation to obtain $\theta$ and $\phi$:

$$\begin{cases} vx' = R\cos(\phi)\sin(\theta) \\ vy' = R\sin(\phi)\sin(\theta) \\ vz' = R\cos(\theta) \end{cases} \tag{5}$$

By rearranging the formulas:

$$\begin{cases} \phi = \arctan(\frac{vy'}{vx'}) \\ \theta = \arccos(\frac{vz'}{R}) = \arccos(\frac{vz'}{\sqrt{vx'^2+vy'^2+vz'^2}}) \end{cases} \tag{6}$$

The obtained samples are real valued vectors, whose two first components are the spherical coordinates $\phi$ and $\theta$ of a pixel, while the third last ones are the observed RGB color values at that pixel:

$$\mathbf{x} = (\theta, \phi, r, g, b) = \left(x_1, x_2, x_3, x_4, x_5\right) \tag{7}$$

so that $\mathbf{x} \in \mathbb{R}^5$. At each time instant $t$, a new episode $S_t \subset \mathbb{R}^5$ is acquired which contains one sample for each pixel of the incoming video frame at time $t$. Therefore the cardinal of $S_t$ is the number of pixels in the video frame.

## 2.2 | Neural Gas network

Next a Neural Gas model (T. Martinetz & Schulten 1991; T. M. Martinetz et al. 1993) for panoramic scenes captured by PTZ cameras is developed. In order to learn the details of the scene, a large number of neurons $N$ is employed, which is lower but in the same order of magnitude as the number of pixels of the full panorama. Following the strategy in (Luque-Baena, López-Rubio, Domínguez, Palomo, & Jerez 2015), the input vectors $\mathbf{x} \in \mathbb{R}^5$ are divided into two sections. The first section contains the positional information in the video frame, while the second section contains the color features. In our case, the first section comprises the two spherical coordinates $\phi$ and $\theta$, while the second section contains the RGB color values. All the components of the input vectors are used to update the neuron prototypes $\mathbf{w}_i \in \mathbb{R}^5$, but only the two first ones participate in the competition. This way, only the positional information is employed to determine which neurons are closest to the input vector. Therefore, each neuron represents the average color in a receptive field which is one region of a Voronoi tesellation of the full panorama. That is, the neurons specialize on small pixel neighborhoods of the panorama. Since the positional components can have fractional values, $\left(x_1, x_2\right) \in \mathbb{R}^2$, the network can learn fractional positional information without having to round to integer pixel positions in the panorama, which avoids losing valuable information.

Each neuron contains a prototype $\mathbf{w}_i \in \mathbb{R}^5$ and also a Boolean flag $b_i \in \{true, false\}$. The flag is required to control the initialization of the neuron. Since the PTZ camera does not cover the entire panorama at a time, it is not possible to initialize all the neurons at the same time. They can only be initialized while their positional components fall into the current field of view. To this end, the Boolean flags are initialized to false. The first time that a neuron wins, its prototype is set to the input sample, and its flag is set to true. From that point on, the neuron will be updated according to the Neural Gas learning rule.

The proposed learning algorithm is as follows:

1. Draw a training sample $\mathbf{x}$ at random from from the current episode $S_t$, which has not been considered before, i.e. a random sampling without replacement is done.

2. For each neuron $i$ compute the number $k_i$ of neurons such that they are closer than $i$ to the input vector in terms of Euclidean distance according to the first two vector components:

$$k_i = \sum_{j \in \{1,...,N\}} \mathbb{I}\left(d_j < d_i\right) \tag{8}$$

$$\forall j \in \{1, ..., N\},\ d_j = \left\| \left(w_{j,1}, w_{j,2}\right) - \left(x_1, x_2\right) \right\| \tag{9}$$

$$\mathbb{I}\left(condition\right) = \begin{cases} 0 & \text{if } condition \text{ is false} \\ 1 & \text{if } condition \text{ is true} \end{cases} \tag{10}$$

   where $\mathbb{I}$ stands for the indicator function.

3. Let $q$ be the index of the closest (winning) neuron, $k_q = 0$. If the flag $b_q$ is true, then go to step 4. Otherwise, set the prototype of the winning neuron $q$ to the training sample and set its flag to true:

$$\mathbf{w}_q = \mathbf{x} \tag{11}$$

$$b_q = true \tag{12}$$

   Then go to step 4.

4. Update the neuron prototypes for the $M$ closest neurons according to the Neural Gas learning rule:

$$\mathcal{N} = \left\{ i \in \{1, ..., N\} \mid k_i < M \right\} \tag{13}$$

$$\forall i \in \mathcal{N},\ \Delta \mathbf{w}_i = \epsilon \exp\left(-\frac{k_i}{\lambda}\right)\left(\mathbf{x} - \mathbf{w}_i\right) \tag{14}$$

$$\forall i \notin \mathcal{N},\ \Delta \mathbf{w}_i = 0 \tag{15}$$

   where $\epsilon$ is a decaying learning rate and $\lambda$ is a decaying neighborhood radius.

5. If all the samples of the current episode $S_t$ have already been processed, then go to step 6. Otherwise, go to step 1.

6. If the last time instant $t$ has been reached, then stop. Otherwise, increment the time instant counter $t$, load the next episode and go to step 1.

There are two phases in the learning process: first the ordering phase where $\epsilon$ experiences a linear decay (initial learning rate $\epsilon_I$); and then the convergence phase where $\epsilon$ remains constant at a small value ($\epsilon_C$). This is because the ordering phase is required for the warm-up of the algorithm only, and after that the system runs for an indefinitely long time. The change of phase is monitored by a step parameter $n$.

$$\epsilon(t) = \begin{cases} \epsilon_I(1 - t/n), \text{ if } t < n \\ \epsilon_C, \text{ otherwise} \end{cases} \tag{16}$$

$\lambda$ also experiences a linear decay as $\epsilon$ with the following equation.

$$\lambda(t) = \begin{cases} \lambda_I - (\lambda_I - \lambda_C)(1 - t/n), \text{ if } t < n \\ \lambda_C, \text{ otherwise} \end{cases} \tag{17}$$

A theoretical analysis of the above algorithm can be carried out. Since the competition is done on the first two components, the algorithm seeks a local minimum of an energy function $\mathcal{E}$ T. M. Martinetz et al. (1993), which only takes into account these components:

$$\mathcal{E} = \frac{1}{2C\left(\lambda\right)} \sum_{i \in \{1,...,N\}} \int d_i^2 \exp\left(-\frac{k_i}{\lambda}\right) p\left(x_1, x_2\right) dx_1 dx_2 \tag{18}$$

$$C\left(\lambda\right) = \sum_{k=0}^{M-1} \exp\left(-\frac{k}{\lambda}\right) \tag{19}$$

where $F_i$ is the receptive field of the $i$-th neuron:

$$F_q = \left\{ \mathbf{x} \mid q = \arg \min_{i \in \{1,...,N\}} d_i \right\} \tag{20}$$

On the other hand, the prototype update is carried out on the last three components too, so those components approximate the average color of the receptive field:

$$\left(w_{i,3}, w_{i,4}, w_{i,5}\right) \approx E\left[\left(x_3, x_4, x_5\right) \mid F_i\right] \tag{21}$$

As a last remark, it must be pointed out that the large number of neurons $N$ to be used for this application requires a considerable optimization of the competition equation (8). This is accomplished by inserting all the first sections $\left(w_{i,1}, w_{i,2}\right)$ of the neuron prototypes into a quad-tree (Frisken & Perry 2002). A quad-tree is a data structure based on a recursive decomposition in order to represent the information. It is suitable for image processing because of their properties to consider the subsets of the data which are more interesting, obtaining an enhanced execution time, so that, improving the performance of the approach where it is applied (Samet 1984). This way (8) is evaluated very quickly, even for values of $N$ in the millions.

## 2.3 | Ensemble model

As mentioned before, the number of neurons $N$ required for this application is in the millions. In order to reduce the time

required to update the network, so that faster training times are attained, an ensemble model is proposed next.

An ensemble of $Q$ Neural Gas networks will be used, each with $N$ neurons. The members of the ensemble operate in parallel, so that each network randomly draws an input vector from the input dataset independently from the other networks. This way, a grand total of $QN$ neurons are trained, while the overall training time can be close to that of a single network with $N$ neurons, provided that at least $Q$ parallel processors are available.

Once the training is finished, a unified network made of $QN$ neurons is built by joining the neurons coming from the $Q$ members of the ensemble. This way, a more accurate representation of the input distribution is obtained with reduced training time requirements.

## 2.4 | Delaunay triangulation based interpolation

Since the positions of the neuron prototypes are given by real numbers, there is no direct way to obtain the estimated color for the integer valued pixel coordinates of the panorama. In order to overcome this difficulty, we propose to build the Delaunay triangulation (Lee & Schachter 1980) of the set formed by the first sections $\left(w_{i,1}, w_{i,2}\right)$ of each neuron prototype. This way, a triangulation of the panorama is obtained. Then, for each integer valued pair of pixel coordinates $\left(y_1, y_2\right) \in \mathbb{N}^2$, the triangle which it belongs to is computed, along with its barycentric coordinates with respect to that triangle:

$$\mathbf{y} = \lambda_i \left(w_{i,1}, w_{i,2}\right) + \lambda_j \left(w_{j,1}, w_{j,2}\right) + \lambda_k \left(w_{k,1}, w_{k,2}\right) \quad (22)$$

$$i, j, k \in \{1, ..., N\} \quad (23)$$

$$\lambda_i, \lambda_j, \lambda_k \geq 0 \quad (24)$$

$$\lambda_i + \lambda_j + \lambda_k = 1 \quad (25)$$

Then the estimated color $\left(y_3, y_4, y_5\right) \in \mathbb{R}^3$ is obtained by linear interpolation with weights equal to the barycentric coordinates:

$$\left(y_3, y_4, y_5\right) = \lambda_i \left(w_{i,3}, w_{i,4}, w_{i,5}\right) +$$

$$\lambda_j \left(w_{j,3}, w_{j,4}, w_{j,5}\right) + \lambda_k \left(w_{k,3}, w_{k,4}, w_{k,5}\right) \quad (26)$$

Therefore a continuous, piecewise linear function is employed to estimate the color over the panorama.

# 3 | EXPERIMENTAL RESULTS

In this section we report the computational experiments we have carried out and their results. The software and hardware that have been used are specified in Subsection 3.1. Then, the tested video sequences are described in 3.2. The descriptions of the used parameters and the proposed methods are in Subsection 3.3, and the descriptions of the competitors in 3.4. Finally, the obtained results from the experiments are reported in Subsection 3.5.

## 3.1 | Methods

The camera control module is based on the *virtualptz library* (G. Chen et al. 2015). It simulates a PTZ camera from a panoramic video sequence, and it is accessible from its website [1]. The implementation is written in C++ and it uses the OpenCV and OpenGL libraries. This virtual camera has limitations in its vertical movement, going from 0 (up) to 180 (down) degrees.

To generate the input data for the Neural Gas network and their the competitors, a exhaustive scanning of the scene has been carried out. To simulate the real behavior of a PTZ camera in a practical setting, some limitations in the scan have been imposed. Starting at the top (vertical 0 degrees), we turn 360 degrees to the right with a step of 10 degrees, and when we come back to the initial point, we go down 10 degrees. This process is repeated until we arrive to the bottom and we start to go up again. Furthermore, a random zoom (vertical field of view) is applied to each frame, but again, we try to simulate a real situation. To this end, we generate a random number in an interval $[FOV, FOV + 5]$ and move this interval by steps of 5 degrees between a minimum and maximum value for the field of view, $[70, 140]$ degrees, in order to avoid strange images with irregularities.

A total of 3000 camera frames have been saved in binary files to read them synchronously by all the methods, which have been implemented in Matlab R2015b. If the panoramic video sequences has less than 3000 frames, then we cause a loop in the video to obtain the remaining frames. The reported experiments have been carried out on a 64-bit Personal Computer with an eight-core Intel i7 3.60GHz CPU, 32 GB RAM and standard hardware. The implementation of our approach does not use any GPU resources.

Panoramic ground truth (GT) has been calculated doing the median of the raw panoramic video frames, but only for a maximum of 600 frames since it is necessary a considerable amount of memory to process it. This is not very relevant because we are going to compare all methods with the same

[1] https://bitbucket.org/pierre_luc_st_charles/virtualptz_standalone

image, so it does not matter if objects that do not belong to the real background appear in the obtained GT. For each method, we compared the panoramic image obtained with the ground truth. Two quality measures were used to evaluate the proposed approach: the first was the Mean Squared Error ($MSE$) metric (lower is better), which is commonly used in image and video processing; the second was the Structural Similarity index (higher is better), which focuses on structural similarities between images:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (27)$$

where $\mu_x$and $\mu_y$ are the mean value of images $x$ and $y$, $\sigma_x$ and $\sigma_y$ are the standard deviation of images $x$ and $y$, $\sigma_{xy}$ is the covariance of $x$ and $y$, $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ ($L$ is the dynamic range, $k_1 = 0.01$ and $k_2 = 0.03$). The values obtained from (27) are averaged over the three RGB channels to obtain the performance for color images. Last, CPU times were measured offline for each input binary file to obtain the real running time of each algorithm per frame, without including the extra waiting time to acquire the next frame that would be required if we did it online.

## 3.2 | Sequences

For the experiments, a total of 6 different sequences obtained from two video databases are utilized. One of them has three videos which are available on the *virtualptz* library website. They are three indoor sequences, but two of them with the same scene, and they are named *scenario3* and *scenario4*, so we have only selected the first one for the final experiments. The second video that we used from this database is *scenario5*. The *scenario3* (3500x1750 pixels and 566 video frames) video shows a shopping mall and *scenario5* (3500x1750 pixels and 1957 frames) shows a room, both of them with people moving on in and doing different actions. On the other hand, we used four video sequences of the Littlstar web page[2]. We named these videos as *scenario6* (2880x1440 pixels and 1169 frames), which shows a beach with people moving and playing beach volleyball, *scenario7* (2048x1024 pixels and 1380 frames), which shows a train station with people, *scenario8* (1920x960 pixels and more than 7800 frames), which shows a big beach with some cars and people passing in the left side of the image, and *scenario9* (1920x960 pixels and more than 9000 frames), which shows a shore near a pier. Statistics were calculated using the six sequences. An important detail to consider is that the four videos from Littlstar have a dynamic background, like movements produced by the air in the train station, or the waves in the sea, and also the sun over time.

[2] https://littlstar.com/

**TABLE 1** Considered parameter values for our Neural Gas network

| Parameter | Value |
|---|---|
| Number of steps $n$ | 1000 |
| Initial learning rate $\epsilon_I$ | 0.4 |
| Convergence learning rate $\epsilon_C$ | 0.01 |
| Number of closest neurons $M$ | 2 |
| Initial neighborhood radius $\lambda_I$ | 1.0 |
| Convergence neighborhood radius $\lambda_C$ | 0.01 |

## 3.3 | Parameter selection and proposals

A set of tuned parameters is needed to define the neural gas model. This study has been done using *scenario4*, which is different of the other videos used for the final experiments. These fixed parameters are reported in Table 1 . We have based ourselves on the parameters presented in articles as T. Martinetz and Schulten (1991) and T. M. Martinetz et al. (1993) and we have tried to find a balance between quality and time efficiency (which depends on the value of $M$).

Taking into account the preliminary tests, we propose four different versions of the Neural Gas network varying the number of neurons: NG30, NG50, RENG30 and RENG50. The first two consist on the simple Neural Gas network using 30% and 50% of neurons with respect to the total number of pixel of the panoramic image. We have selected a thirty percent to show that we can obtain almost the same results that with fifty percent of neurons. In addition, we tested two ensemble models with the same quantity of neurons, called RENG.

## 3.4 | Competitors

We have compared the four proposed methods based on Neural Gas network to three other methods. The competing methods read each frame, then position it in the panoramic image matrix and finally compute the mean over all the frames.

Since data input coordinates are almost always fractional numbers, the panoramic integer coordinates where the incoming frame must be placed were calculated by rounding the original fractional coordinates. Besides, the coordinate pairs do not define a regular strictly monotonic grid, i.e., the points have no structure or order between their relative locations, so that the usual interpolation methods defined to obtain the interpolated RGB values in a panorama cannot be used. Therefore, the scattered interpolant method from MATLAB was employed to manage this situation. It provides the following interpolation variants: 'nearest' (nearest-neighbor interpolation), 'linear' (linear interpolation), and 'natural' (natural-neighbor interpolation). The same input data was provided to

all the competitors as well as our method, i.e. the 3000 camera frames saved in binary files.

## 3.5 | Results

A comparison from a quantitative point of view between the results produced by each method can be observed in Figures 2 , 3 , 4 and 5 .

Figure 2 shows the MSE of the four proposed configurations of the Neural Gas model. This figure displays the comparative evolution of the methods with respect to the number of captured PTZ camera frames, i.e. the frame index in the video sequence. For all the scenarios, the best method is RENG50, followed by NG50 (except for *scenario8*, where both roles are exchanged). The second observation is that there is no significant differences between the four methods in most scenarios. For example, RENG30 obtains results close to NG50, which means that we can use a lower quantity of neurons, i.e., a lower processing time, to obtain the same results. The tendency shows that from 2000 frames, the MSE decrease and do not change a lot in the rest of the processing.

The comparison with the competitors is presented in Figure 3 . We only draw the best of our methods (RENG50) for the sake of clarity. When a certain number of frames is reached that covers almost all the panoramic image, which is around 600 frames, our method (in green) attains the best performance values, as compared to the competing methods. In particular, lower values of MSE indicates that the obtained RGB pixel values are more precise. The Neural Gas model do not need a lot of frames to reach a good performance, in contradistinction to the interpolation methods, which need 1500 frames at least. When the number of acquired frames increases to more than 2500, all methods tend to perform similarly, but our method still remains as the best one.

SSIM results for the seven methods is displayed in Figure 4 . For the videos with fixed background, i.e., *scenario3* and *scenario5*, the Neural Gas model clearly overcomes the performance of the competing methods. Among our methods, RENG50 still remains the best. However, when an external factor is present, like wind or sun, i.e, a dynamic background, the final result is affected. In the three scenarios showing a beach, the competitors achieve a better final panoramic background, but the difference with respect to our proposal is less than 1%. If we also consider the MSE to perform a global evaluation, it is clear that the neural model is more suitable.

There is a particularity in the results of *scenario7* (Figures 2 d, 3 d and 4 d). We can appreciate two peaks around 1400 and 2800 frames. That is because this video has a duration of 1 min. 32 s., it has 1380 video frames, and in the last seventy frames, a man cover the camera with a cardboard, so the neural network treat to learn it and the comparison with the GT generates bad results. But when more frames are processed, the net has the ability to recover very fast and obtain better results than the competitors.

Last, it is important to remark that the evaluation for *scenario9* has been carried out comparing with two different ground truths. This video has a particularity: in the first 1350 frames approximately, there are not many people walking around, but then two men arrive and sit on the shore, and stay there until the end off the video. Our model finishes learning this situation but the competitors have it more smoothed, as they do the average of all the frames. When compared to the ground truth, they do a little better. Therefore, we generate another GT taking the six hundred frames to the previously mentioned, and we compared all the methods with both GTs. The original one for the 1349 frames and the second one for the rest. The results were combined in the same graphic (Figures 2 f, 3 f and 4 f). We can observe that our methods improve the competitors in the MSE, and equal the SSIM in the case of RENG50.

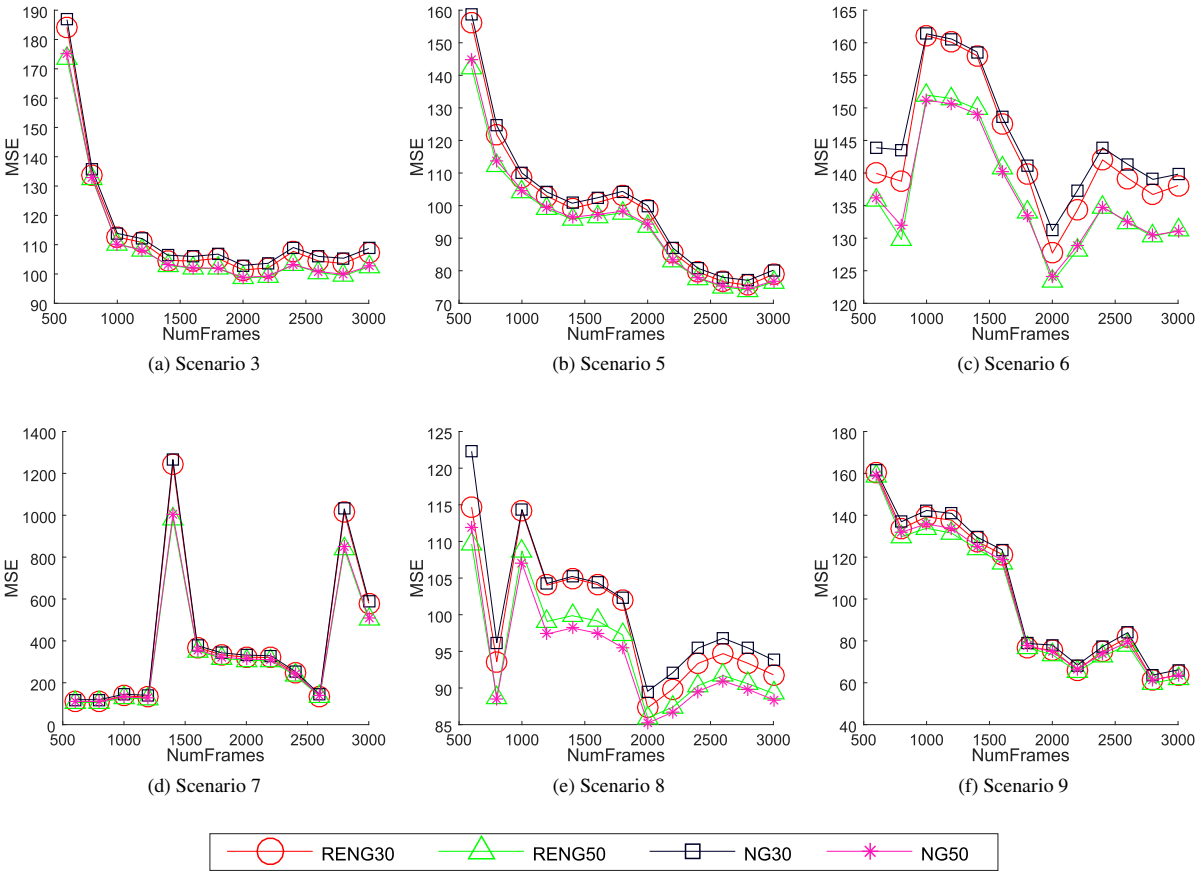
A bar graph is shown in Figure 5 to appreciate the large differences in time required to process a frame. Mean values for the 3000 frames are calculated. Since Random Ensemble methods (RENG) would require almost the same time for the calculations as their simple version (NG), only the latter has been shown. It is clear that our proposals achieve a better performance than the competing methods for any scenario. The similarity between NG30 and NG50 and the so low computing time would permit to vary the number of neurons adapting to each type of scenario to try to find the best configuration.

To summarize, we have calculated the mean and standard deviation of the six sequences for each method and for each performance measure. These qualitative results are shown in Table 2 . As we can see, the Neural Gas model clearly outperforms the competing methods in terms of the mean squared error. Higher values of SSIM confirm that our neural model produces the best approximation of the background of the scene, with special remark to the RENG50 method.

The CPU time required to process one frame is a very important feature to be assessed. Table 2 shows the mean required time to process a binary file for a frame. Our proposed methods are around 87% faster than the best of the interpolation methods. It computes the winner neurons and updates the quad-tree where they are stored in just over a second. If we consider that a movement of the virtual PTZ camera and the generation of the binary file for the current frame takes between one and three seconds, it turns out that our method is the only one that can be executed concurrently with the PTZ camera frame acquisition

**TABLE 2** Mean (standard deviation) measures for each method along the six scenarios. (*) CPU time for these methods has been established the same as their simple version method because they can be parallelized.

| | $MSE$ | $SSIM$ | $Time(sec/frame)$ |
|---|---|---|---|
| NG30 | 179.9321 (202.9430) | 0.9098 (0.0440) | **0.9211** (**0.1359**) |
| NG50 | 162.0514 (171.9265) | 0.9159 (0.0409) | 1.0281 (0.1876) |
| RENG30 (*) | 176.4957 (198.8611) | 0.9148 (0.0423) | 0.9211 (0.1359) |
| RENG50 (*) | **160.4683** (168.6281) | **0.9180** (**0.0409**) | 1.0281 (0.1876) |
| Linear | 212.7255 (**107.3339**) | 0.9138 (0.0451) | 7.0936 (1.0673) |
| Natural neighbor | 212.9676 (107.3528) | 0.9133 (0.0453) | 8.4727 (1.2829) |
| Nearest neighbor | 212.6441 (107.3512) | 0.9149 (0.0444) | 7.1806 (1.0532) |




(a) Scenario 3 (b) Scenario 5 (c) Scenario 6

(d) Scenario 7 (e) Scenario 8 (f) Scenario 9

**FIGURE 2** Comparison of the MSE (lower is better) for the four methods proposed in this paper.

process. Moreover, if we have available more than one processor, we can employ the random ensemble methods RENG30 or RENG50 to improve the quality with the same time. The competitors require seven times more time and incur in a big time delay to obtain the panoramic background. The utilization of a GPU would improve between 25 and 50 times the processing rate, since the inner loop of the proposed method is amenable to single instruction multiple data (SIMD) parallelization. Therefore, each pixel can be processed by a GPU thread. Present day GPUs can run hundreds or thousands of such threads at a time. Given the single thread CPU times shown in Figure 2 , the number of frames that can be processed on a GPU could be increased to 50 frames per second, thereby achieving real time performance.

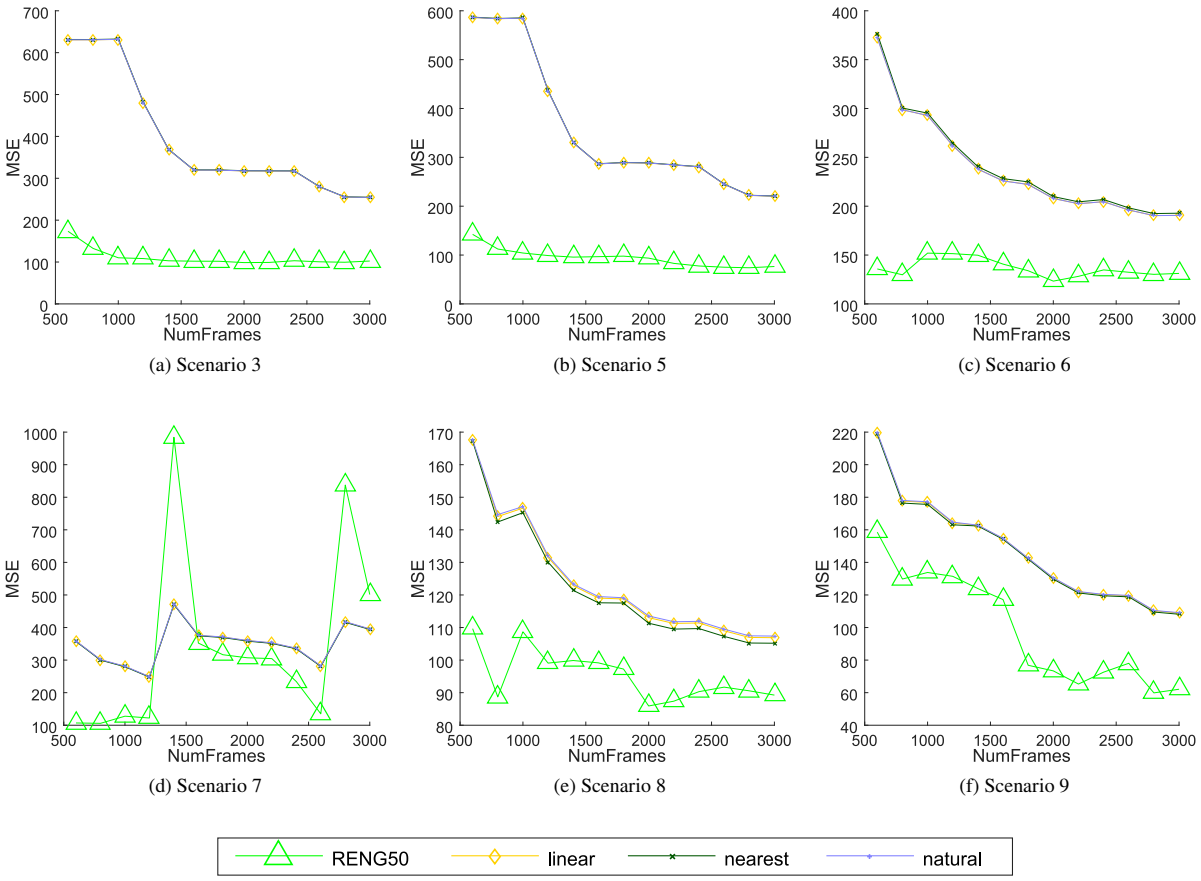



**FIGURE 3** Comparison of the MSE (lower is better) for the RENG50 method and the competitors.

A comparison from a qualitative point of view between the results produced by each method can be observed in Figures 6 , 7 and 8 . All methods remove the moving objects of the video frame efficiently and no relevant differences are noticed. An example of the result of our method is shown in Figure 6 , where a train passes and it does not appear in the panoramic background generated by RENG50. On the other hand, Figures 7 and 8 show two scenarios from the tested videos, where image 7 a represents the panoramic video frame and a selected window of 50x50 pixels (marked in red), and the rest of the images represent the ground truth and the produced result by each method in that window. The qualitative results offered by our approach are the most similar to the ground truth. This can be better appreciated on the top and the bottom of the panoramic image, where the competitors produce black pixels that should not be there. The window of *scenario5* in Figure 7 is a clear example of this. Also, in the left and the right of the images produced by the competitors there are more black pixels than in our approach. In addition, a better appreciation of the learned dynamic background can be observed in Figure 8 . Our approach changes and adapts the modeled background and the new scenario with the remaining person is learned faster than the rest of the competitors.

## 4 | CONCLUSION

In this work, a methodology to model the panoramic background of PTZ cameras is presented. It consists of an online learning method based on individual or ensemble Neural Gas networks to read each camera frame and process it to generate a panoramic image of the scene. Six scenes have been tested to check the feasibility of the system, obtaining suitable and successful results. Also, it has been demonstrated that our approaches outperform several competing methods. It should be noted that the a reduction in the number of neurons (from 50% to 30% of the pixels) implies a slight decrease of the performance metrics, although visually it is not too evident. Additionally, the ensemble models (RENG30 and RENG50)

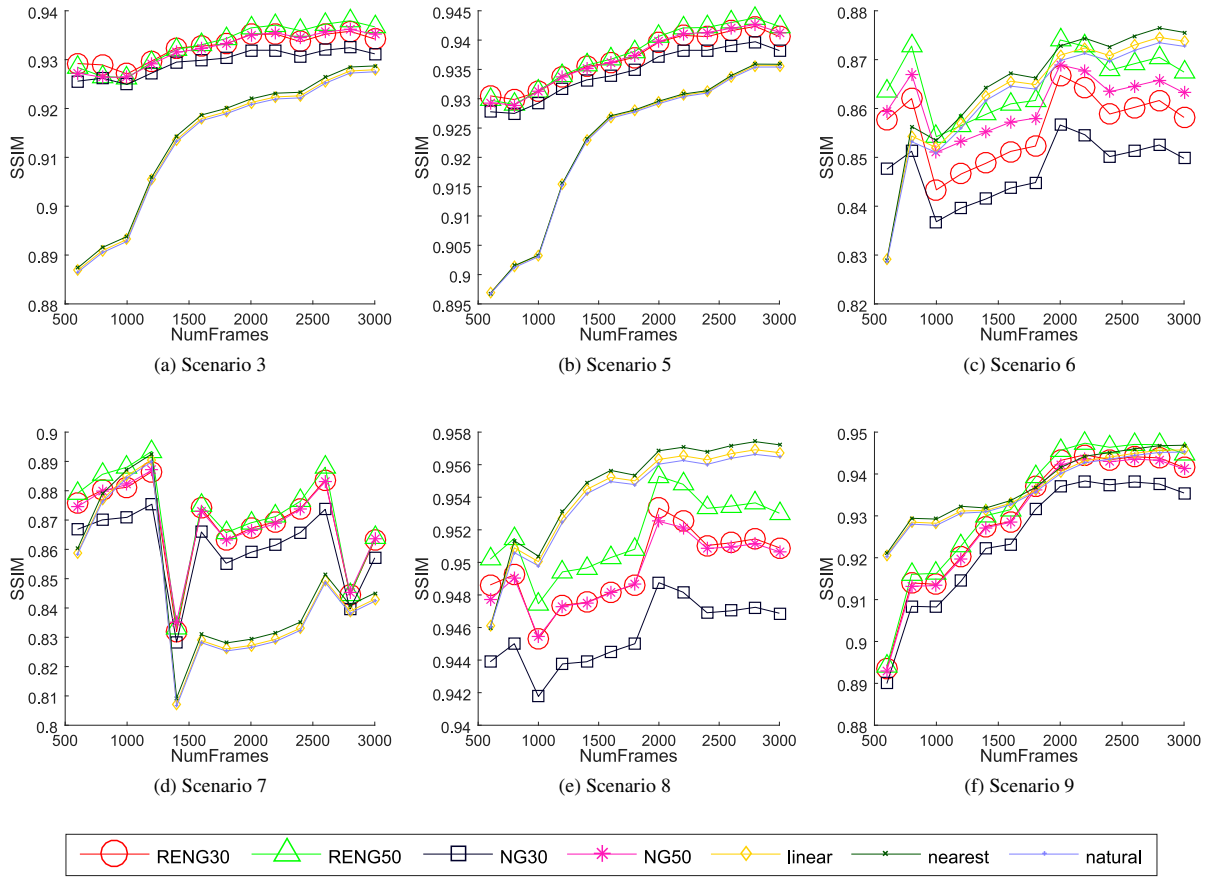



**FIGURE 4** Comparison of the SSIM (higher is better) for the the seven methods.

help to improve the background modeling, raising the performance rates. Finally, it is remarkable that the proposed models uses very little CPU time to process each input video frame, which permits an easy integration in a real time PTZ camera video surveillance system.

## ACKNOWLEDGMENTS

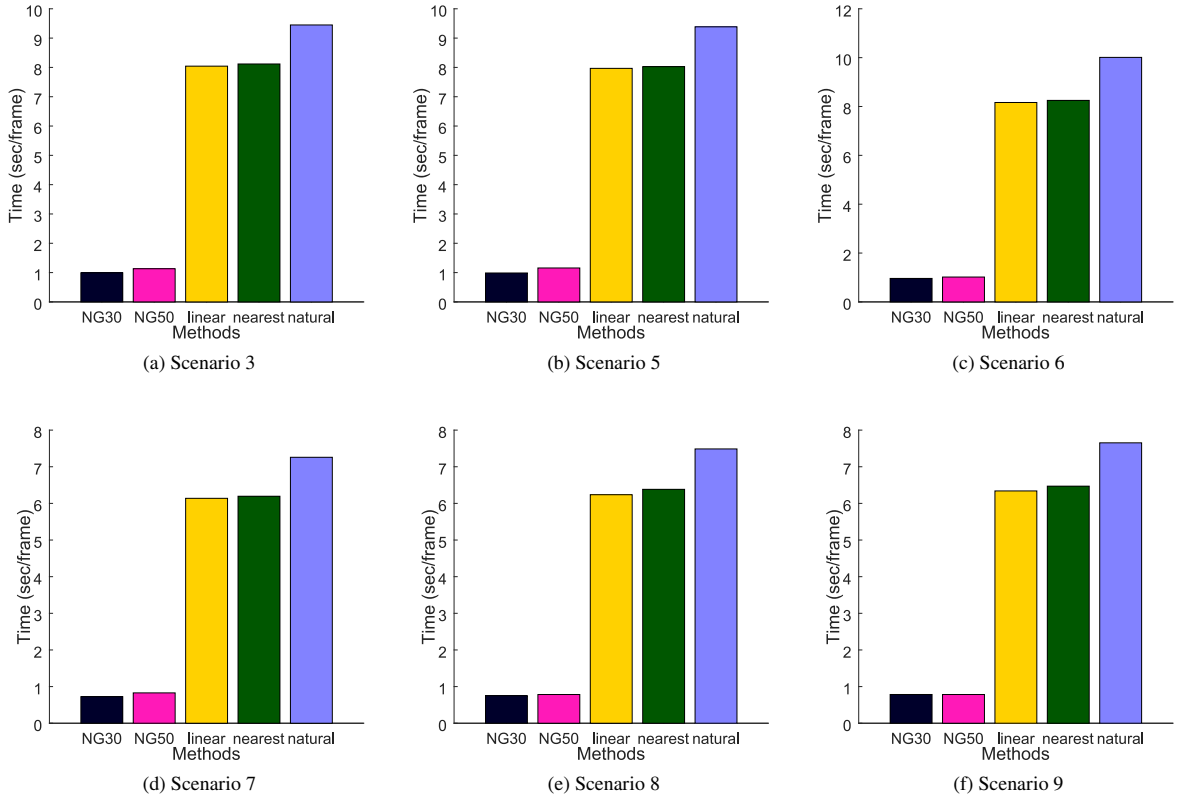
This work is partially supported by the Ministry of Economy and Competitiveness of Spain under grant TIN2014-53465-R, project name Video surveillance by active search of anomalous events, grant TIN2014-57341-R, project name metaheuristics, holistic intelligence and smart mobility, and grant TIN2016-75097-P. It is also partially supported by the Autonomous Government of Andalusia (Spain) under projects TIC-6213, project name Development of Self-Organizing Neural Networks for Information Technologies; and TIC-657, project name Self-organizing systems and robust estimators for video surveillance. All of them include funds from the European Regional Development Fund (ERDF). The authors thankfully the grant of the University of Málaga and acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Málaga. They also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU. Karl Thurnhofer-Hemsi (FPU15/06512) is funded by a PhD scholarship from the Spanish Ministry of Education, Culture and Sport under the FPU program.

## References

Ahalt, S., Krishnamurthy, A., Chen, P., & Melton, D. (1990). Competitive learning algorithms for vector quantization. *Neural Networks*, *3*(3), 277-290.

Antonakakis, M., Dimitriadis, S. I., Papanicolaou, A. C., Zouridakis, G., & Zervakis, M. (2016). Improving the detection of MBTI via complexity analysis in resting - state magnetoencephalography. In *IEEE international conference on imaging systems and techniques (IST)* (p. 156-160).

Azzari, P., & Bevilacqua, A. (2006). Joint spatial and tonal mosaic alignment




(a) Scenario 3 (b) Scenario 5 (c) Scenario 6

(d) Scenario 7 (e) Scenario 8 (f) Scenario 9

**FIGURE 5** Comparison of the average CPU time required to train the 3000 frames captured by the virtual PTZ camera. Only five methods are shown because Random Ensemble methods have the same processing time as the simple method.

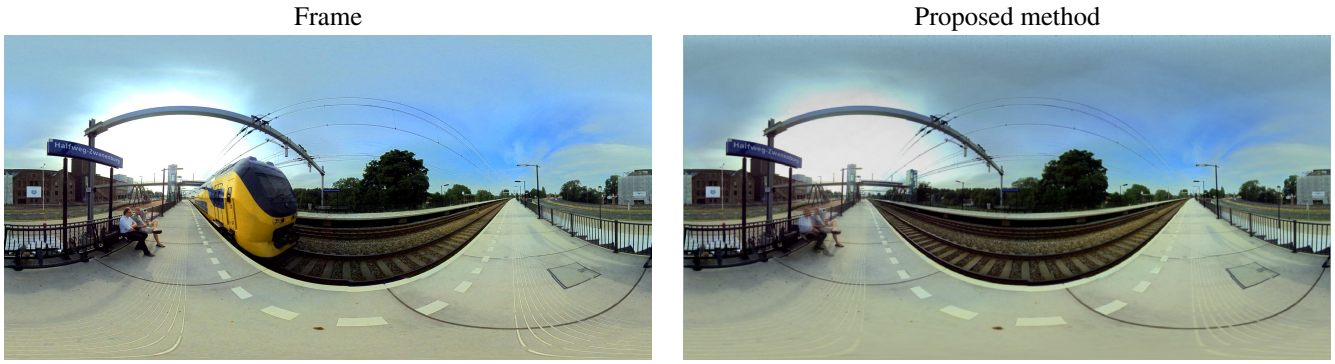



**FIGURE 6** Panoramic images of *scenario7* for frame 300 and the RENG50 proposed method, resp.

for motion detection with PTZ camera. *Lecture Notes in Computer Science*, *4142*, 764–775.

Bevilacqua, A., & Azzari, P. (2006). High-Quality Real Time Motion Detection Using PTZ Cameras. *Proceedings of the IEEE International Conference on Video and Signal Based Surveillance (AVSS 2006)*, 1–6.

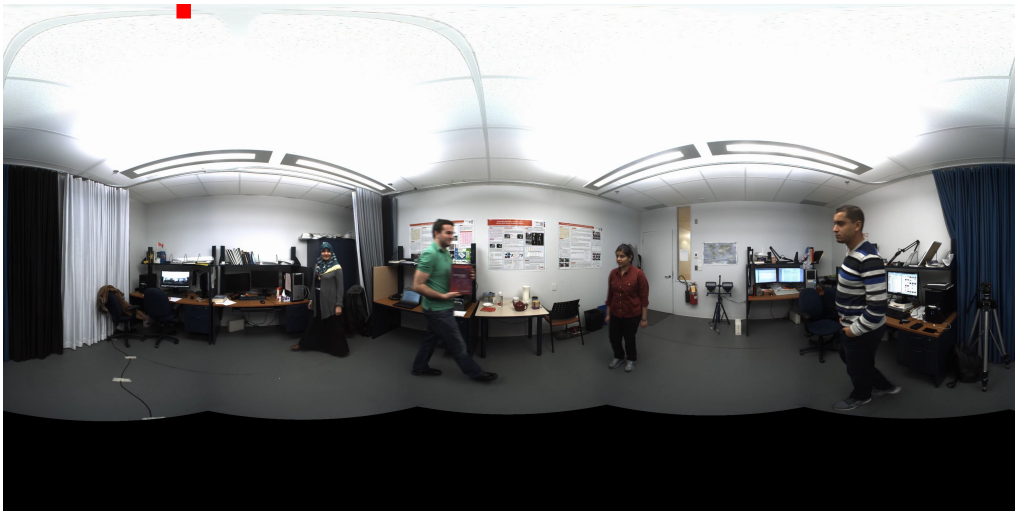
Boult, T., Gao, X., Micheals, R., & Eckmann, M. (2004). Omni-directional visual surveillance. *Image and Vision Computing*, *22*(7), 515-534.

Chen, C.-H., Yao, Y., Page, D., Abidi, B., Koschan, A., & Abidi, M. (2008). Heterogeneous fusion of omnidirectional and ptz cameras for multiple object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, *18*(8), 1052-1063.

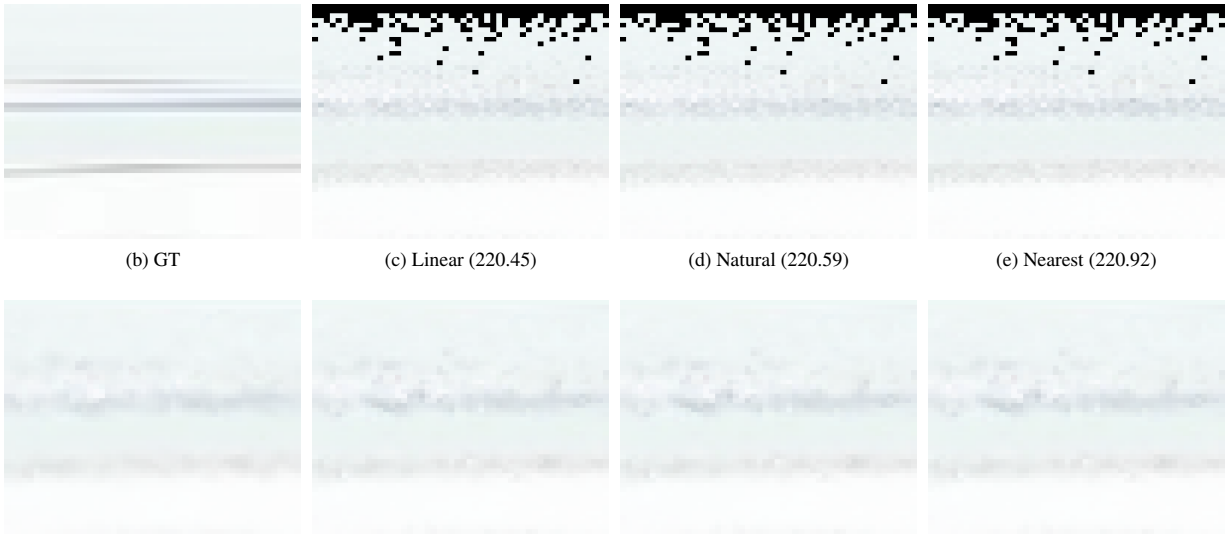
Chen, G., St-Charles, P., Bouachir, W., Joeisseint, T., Bilodeau, G., & Bergevin, R. (2015). Reproducible evaluation of pan-tilt-zoom tracking. *CoRR*, *abs/1505.04502*. Retrieved from `http://arxiv.org/abs/1505.04502`

Chen, H.-P., Shen, X.-J., & Long, J.-W. (2016). Histogram-based colour image fuzzy clustering algorithm. *Multimedia Tools and Applications*, *75*(18), 11417-11432.

Chung-Chen Chen, Yi Yao, Drira, A., Koschan, A., & Abidi, M. (2009).



(a) Frame 588



(b) GT (c) Linear (220.45) (d) Natural (220.59) (e) Nearest (220.92)

(f) NG30 (80.17) (g) NG50 (76.86) (h) RENG30 (79.53) (i) RENG50 (76.31)

**FIGURE 7** Graphical depiction of the operation of the seven methods evaluated with *scenario5*. MSE (lower is better) of the whole panoramic image is shown in parentheses.

Cooperative mapping of multiple PTZ cameras in automated surveillance systems. In *2009 ieee conference on computer vision and pattern recognition* (pp. 1078–1084). IEEE.

Dimitriadis, S., Sun, Y., Laskaris, N., Thakor, N., & Bezerianos, A. (2016). Revealing cross-frequency causal interactions during a mental arithmetic task through symbolic transfer entropy: A novel vector-quantization approach. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *24*(10), 1017-1028.

Ding, C., Song, B., Morye, A., Farrell, J., & Roy-Chowdhury, A. (2012). Collaborative sensing in a distributed ptz camera network. *IEEE Transactions on Image Processing*, *21*(7), 3282-3295.

Frisken, S. F., & Perry, R. N. (2002). Simple and efficient traversal methods for quadtrees and octrees. *Journal of Graphics Tools*, *7*, 1-11.

Kim, S., Yun, K., Yi, K., Kim, S., & Choi, J. (2013). Detection of moving objects with a moving camera using non-panoramic background model. *Machine Vision and Applications*, *24*(5), 1015-1028.

Konda, K., Conci, N., & De Natale, F. (2016). Global coverage maximization in ptz-camera networks based on visual quality assessment. *IEEE Sensors Journal*, *16*(16), 6317-6332.

Lee, D., & Schachter, B. (1980). Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, *9*(3), 219-242. doi: 10.1007/BF00977785

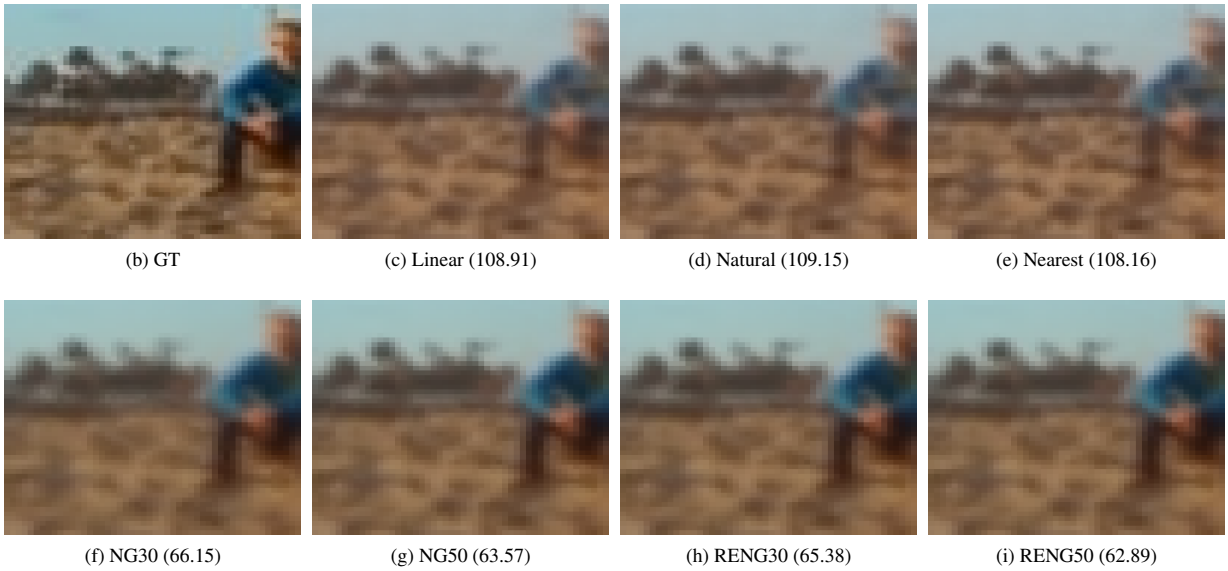
López-Rubio, F. J., & López-Rubio, E. (2015). Foreground detection for moving cameras with stochastic approximation. *Pattern Recognition Letters*, *68*, 161-168.

Luque-Baena, R. M., López-Rubio, E., Domínguez, E., Palomo, E. J., & Jerez, J. M. (2015). A self-organizing map to improve vehicle detection in flow monitoring systems. *Soft Computing*, *19*(9), 2499-2509. doi: 10.1007/s00500-014-1575-3

Martinetz, T., & Schulten, K. (1991). Artificial neural networks. In T. Kohonen, K. Mäkisara, O. Simula, & J. Kangas (Eds.), (Vol. 1, p. 397-402). Elsevier.

Martinetz, T. M., Berkovich, S. G., & Schulten, K. J. (1993). 'Neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, *4*(4), 558-569.

(a) Frame 1500



(b) GT (c) Linear (108.91) (d) Natural (109.15) (e) Nearest (108.16)

(f) NG30 (66.15) (g) NG50 (63.57) (h) RENG30 (65.38) (i) RENG50 (62.89)

**FIGURE 8** Graphical depiction of the operation of the seven methods evaluated with *scenario9*. MSE (lower is better) of the whole panoramic image is shown in parentheses.


Ozan, E., Kiranyaz, S., & Gabbouj, M. (2016). Competitive quantization for approximate nearest neighbor search. *IEEE Transactions on Knowledge and Data Engineering*, *28*(11), 2884-2894.

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, *16*(2), 187–260.

Uchiyama, T., & Arbib, M. (1994). Color image segmentation using competitive learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*(12), 1197-1206.

Valente, R., & Abrao, T. (2016). MIMO transmit scheme based on morphological perceptron with competitive learning. *Neural Networks*, *80*, 9-18.

Varcheie, P. D. Z., & Bilodeau, G. A. (2011). Adaptive fuzzy particle filter tracker for a PTZ camera in an IP surveillance system. *IEEE Transactions on Instrumentation and Measurement*, *60*(2), 354–371.

Xie, H., Luo, X., Wang, C., Liu, S., Xu, X., & Tong, X. (2016). Multispectral remote sensing image segmentation using rival penalized controlled competitive learning and fuzzy entropy. *Soft Computing*, *20*(12), 4709-4722.

Xue, K., Liu, Y., Ogunmakin, G., Chen, J., & Zhang, J. (2013). Panoramic gaussian mixture model and large-scale range background substraction method for ptz camera-based surveillance systems. *Machine Vision and Applications*, *24*(3), 477-492.

Yair, E. (1992). Competitive learning and soft competition for vector quantizer design. *IEEE Transactions on Signal Processing*, *40*(2), 294-309.