ISSN No:-2456-2165

# Crowd Monitoring using HOG

Sri.N.V.Phani Sai Kumar (Asst. Professor), Y. Sriram Kalyan, U. Akshaya, V. Lakshmi Pravallika, Zayer Sakeena
Electronics and Communication Engineering SRKR Engineering College

**Abstract:- Many security and event management agencies throughout the world are beginning to understand the significance of crowd surveillance as public safety concerns increase. These organisations can avert any unforeseen mishaps or problems by estimating crowd dynamics. The goal of this research is to develop a system that can more effectively monitor crowds utilising Support Vector Machine (SVM) classifiers and Histogram of Oriented Gradients (HOG) features. According to our needs, we can interface two or more cameras to count the number of individuals in the input video of the cameras and to identify their locations in 3D space. This provides a sense of the density.**

## I. INTRODUCTION

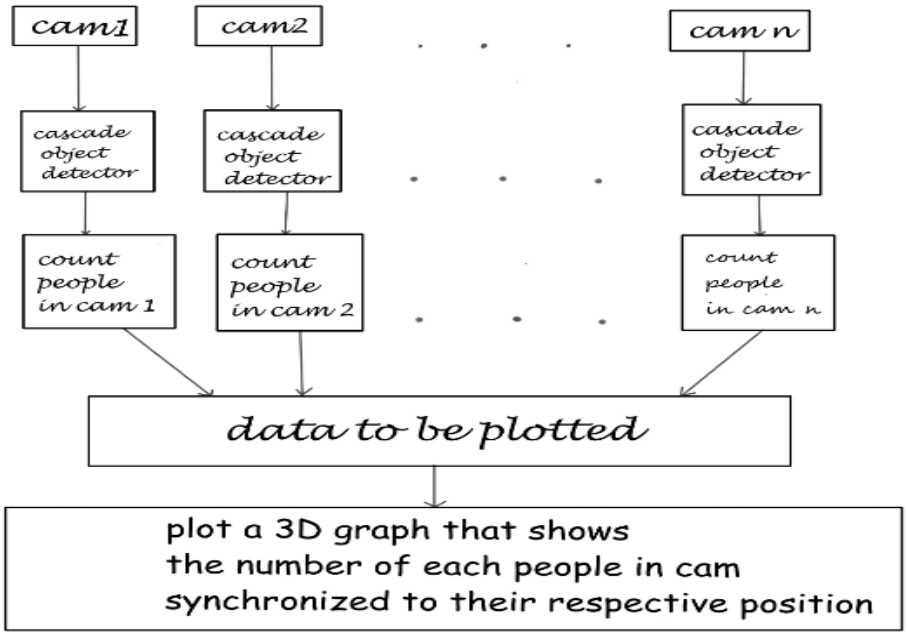
Visual surveillance systems now in use often require human operators to watch video streams from various cameras, which is impossible when more cameras are viewed. This monitoring system is also concerned with identifying suspicious activities and keeping individuals safe.

A growing number of cameras need a growing number of displays as well as a growing number of viewers, both of which increase expenditures. Furthermore, human real-time monitoring is time-consuming and exhausting yet essential for safety. This is a result of how the data from the camera is shown to the observer as well: one monitor is coupled with one or more cameras. This presents at least two issues: first, a small window of time and location may be watched, and second, changing between cameras causes the overview and context to be lost, which is intellectually taxing and wearisome for the security personnel. Operators often aren't able to maintain the appropriate degree of focus for more than 20 to 45 minutes as a consequence, which raises the cost of employees.

Automated visual surveillance systems are one solution to these issues. However, the difficulties of complicated scenarios as they arise in real-world applications are beyond the capabilities of current automated systems. They can only be used in situations where there is a low to medium density of people because to their limited dependability, which includes low recall and high false alarm rates. Additionally, the installation and upkeep of such systems sometimes still need extensive hand-tuning, which becomes increasingly impractical as the number of cameras rises.

To solve these issues, we integrate object identification with 3D visualisation and provide a graphical user interface that makes it easier to track for individuals.

## II. METHODOLOGY



ISSN No:-2456-2165

## III. IMAGE ACQUISITION

The act of obtaining an image from sources is known as image acquisition. Hardware systems like cameras, encoders, sensors, etc. can be used to do this. It is without a doubt the most important phase in the MV workflow since a bad picture would make the workflow ineffective as a whole. Gaining a picture with the proper quality and contrast is crucial since machine vision systems only evaluate the digital image of the thing that has been captured, not the actual object.

A set of photo-sensitive sensors turn an object's incoming light wave into an electrical signal during the picture capture process. These little components provide the function of accurately describing the item to your machine vision algorithms.

It's a frequent fallacy that with an MV system, choosing the correct colours is crucial. Though it's not always the case. Colours frequently increase noise and make detection more challenging. The main objective of an image capture system is to increase contrast for the important features. The appropriate image is one in which the camera can see the object of interest. The components of an image capture system will now be covered.

There are four key components in the picture capture system. While the efficacy of the sensors and cameras may vary depending on the technology available, the lighting systems are completely within the users' control. The following list includes the principal image acquisition components:

- Trigger
- Camera
- Optics
- Illumination

Here we use the digital camera for image acquisition for our convenience.

## IV. BUILDING GUI WITH A GUIDE

Apps, or graphical user interfaces (GUIs), provide you point-and-click control over your software programmes, removing the need for others to learn a language or input commands in order to utilise the programme. Apps can be shared for usage inside of MATLAB as well as outside of it as independent desktop or online applications.

The three methods listed below can be used in any combination to develop an app in MATLAB:

*A. Convert a script into a simple app:*

When you wish to share a script with students or co-workers and let them change variables using interactive controls, use this option.

Utilize the Live Editor to turn a script into a straightforward app with interactive controls so that users can play around with the variables in your code. Without writing any code, add sliders, dropdowns, edit fields, and buttons. Indicate which sections of the script should be executed when a value is updated. To design straightforward applications and dashboards, hide the coding. Others can utilise your live script in MATLAB or MATLAB Online by sharing it with them. Interactive controls are useful for making a live script easier to use. If you wish to create a user interface with a more complex layout or want greater control over the app's behaviour, you should consider building a standalone app using App Designer.

*B. Create an app interactively:*

Pick this option if you want to design a more complex user interface for your app utilising a drag-and-drop environment.

Laying out the aesthetic components and scripting the app's functionality are the two main duties of creating an app, and App Designer is an interactive environment that merges both of these processes. It enables swift transitions between writing code in the MATLAB editor and visual design on the canvas. By utilising MATLAB Online or MATLAB Desktop, you may distribute your software for usage by others. The MATLAB Software tab may also be used to bundle App Designer apps for installation.

You may use MATLAB to build applications into independent desktop and online apps to share with non-MATLAB users. You may interactively create your layout and programme its functionality in one environment using Compiler App Designer. You may programme the entire software yourself if you'd like, including the user interface.

*C. Create an app programmatically:*

If you wish to write the code for the user interface of an app, select this option.

You may also describe the structure and functionality of your app using MATLAB functions for more flexibility in design and development. In this method, you build a figure to act as the user interface's container and programmatically add components to it. By utilising MATLAB Online or MATLAB Desktop, you may distribute your software for usage by others. Additionally, applications may be packed and installed into the MATLAB Apps page. You may utilise the MATLAB Compiler to convert your applications into independent desktop programmes to distribute to non-MATLAB users.

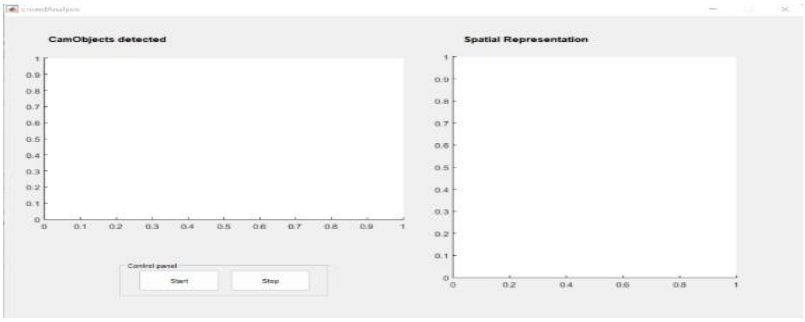



Fig. 1: GUI formed using guide

*D. Histogram of oriented gradients (HOG)*

The gradient vector for the function f (x, y) (fx, fy). Since an image is a discrete function of (x, y), the gradient of an image may also be determined. The horizontal and vertical directions are determined for each pixel in a picture gradient.

These vectors have magnitude $\sqrt{(f_x^2 + f_y^2)}$ and direction $\tan^{-1}\left(\frac{fx}{fy}\right)$

Gradients values are mapped in the range of 0 to 255. Pixels with large negative change will be black, Pixels with large positive change will be white and Pixels with no change or little change will be grey.

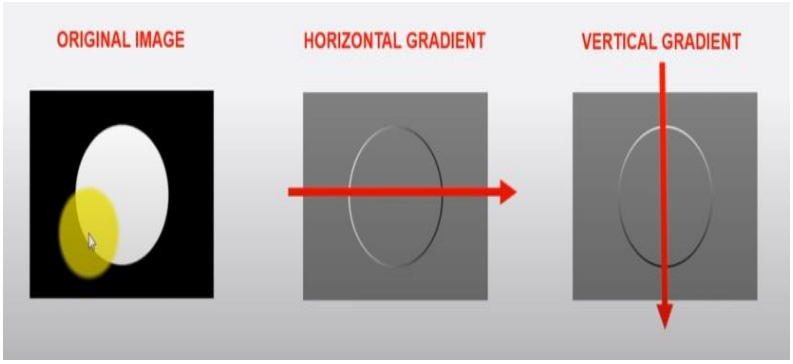



Fig. 2: gradient behavior

The first image is observed in both horizontal gradient and vertical gradient. In the horizontal gradient image, there is a transition from black to white (i.e., a 0 to 1 positive transition), which is shown with a grey colour, and then a swift transition from white to black (i.e., a 1 to 0 negative transition), which is shown with a black colour.

In the vertical gradient image, there is a transition from black to white (i.e., a 0 to 1 positive transition), which is shown with a grey colour, and then a swift transition from white to black (i.e., a 1 to 0 negative transition), which is shown with a black colour.

Let's study an example here.

The gradient value in the x direction is 120-70=50

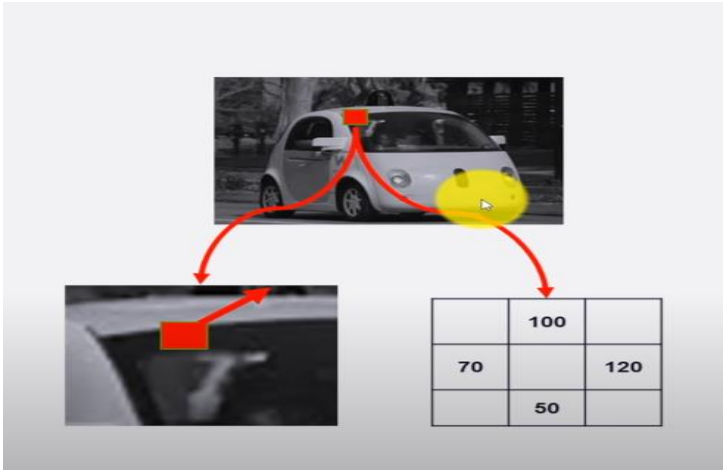



Fig. 3: Pixels of a patch

ISSN No:-2456-2165

Gradient magnitude: $\sqrt{50^2 + 50^2} = 70.1$

Gradient direction:$\tan^{-1}\left({}^{50}/_{50}\right) = 1$

Steps to be followed to get the histogram

- *Step1:*
  Use an 8 x 8pixel cell to compute gradient magnitude and direction

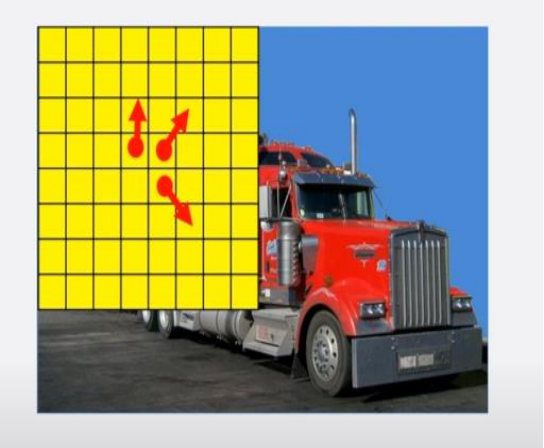


Fig. 4: Calculation of gradient

- *Step 2:*
  Create a histogram of generated 64(8 x 8) gradient vectors

  Each cell is split into angular bins each bin corresponds to a gradient direction (9 bins 0-180$^0$) (20$^0$ each bin)

  This reduces 64 vectors into 9 values

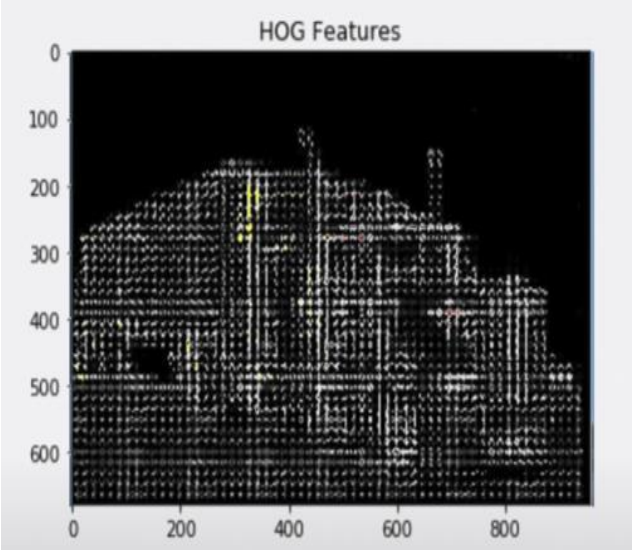



Fig. 5: HOG features

## V. FLOWCHART

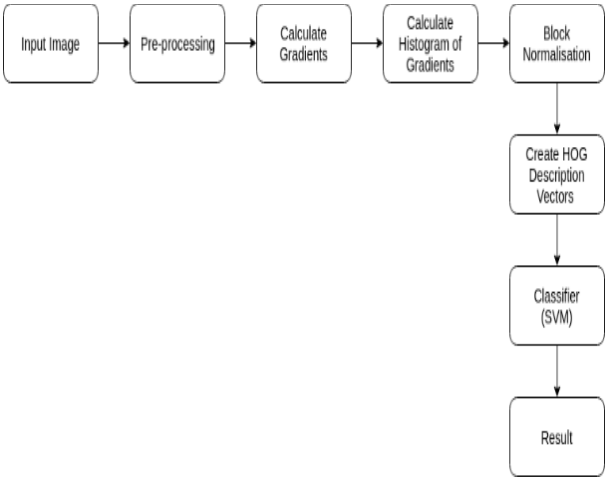



Fig. 6: Flow chart of calculating HOG

## VI. STEPS TO CALCULATE HOG FEATURES

- Start by taking the input picture for the HOG feature calculation. The picture should be resized to 128 by 64 pixels (128 pixels in height and 64 widths). The authors' main goal with this form of detection was to get better results, thus they utilised and recommended this dimension in the study.
- The image's gradient is computed. Combining the image's magnitude and angle yields the gradient. Gx and Gy are initially computed for each pixel in a block of 3x3 pixels. Making use of the formulas for each pixel value, Gx and Gy are first computed.
- After determining the gradient of each pixel, a block is created by dividing the gradient matrices (the magnitude and angle matrix) into 8x8 cells. A 9-point histogram is computed for each block. A 9-point histogram creates a histogram with 9 bins, each with a 20-degree angle range. A 9-bin histogram with values assigned following computations is shown in Figure 8. These 9-point histograms may all be shown as histograms with bins that output the gradient's strength for that bin. A block can have 64 distinct values, hence the following computation is done for all 64 magnitude and gradient values.
- We will determine the jth bin for each cell in a block before determining the value that will be given to the jth and (j+1)th bins, respectively.
- Values of Vj and Vj+1 are appended in an array at the index of the jth and (j+1)th bins calculated for each pixel. An array is used as a bin for a block.
- The 16x8x9 form of the resulting matrix from the computations above.
- After the histogram computation for each block is complete, four blocks from the 9-point histogram matrix are combined to create a new block (2x2). With an 8-pixel stride, this clubbing is done in an overlapping fashion. We combine the nine-point histograms for each of the four cells in a block to create a 36-feature vector.
- The L2 norm is used to standardise the fb values for each block:
- $$f_{bi} \leftarrow \frac{f_{bi}}{\sqrt{\|f_{bi}\|^2 + \varepsilon}}$$

- Where ε is a small value added to the square of fb to avoid zero division error.
- Prior to normalisation, the following formulas are used to determine the value of k:

$$k = \sqrt{b_1^2 + b_2^2 + b_3^2 + \ldots\ldots + b_{36}^2}$$

- $$f_{bi} = \left[\left(\frac{b_1}{k}\right), \left(\frac{b_2}{k}\right), \left(\frac{b_3}{k}\right), \ldots\ldots, \left(\frac{b_{36}}{k}\right)\right]$$
- By normalising the data, the impact of variations in contrast between photographs of the same item is lessened. Out of every brick. A feature vector with 36 points is gathered. In the horizontal direction, there are 7 blocks and in the vertical direction, there are 15 blocks so the total length of HOG features will be: 7 x 15 x 36 = 3780. HOG features of the selected image are obtained.

## VII. SUPPORT VECTOR MACHINE (SVM)

A feature vector exists for each pixel. Whether the feature vector is a face or not must be determined. This decision boundary method is the most effective. SVM determines the optimal decision bounds.

### A. *Linear Decision Boundary*

The linear decision boundary in n-D space is (n-1)-D line.

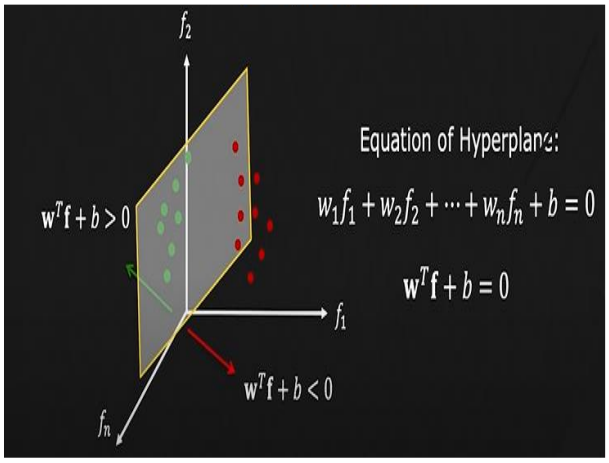



Fig. 7: Decision boundary

### B. *Optimal decision boundary*

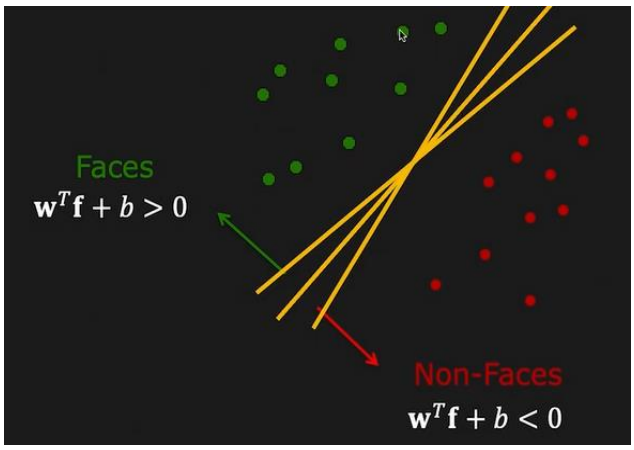



Fig. 8: Many decision boundaries

We are seeking a decision boundary for our training data, and after finding many, we mustassess which decision boundary is the best.

Before reaching a feature point, the breadth of that border might be increased.

### C. *Evaluating a decision boundary*

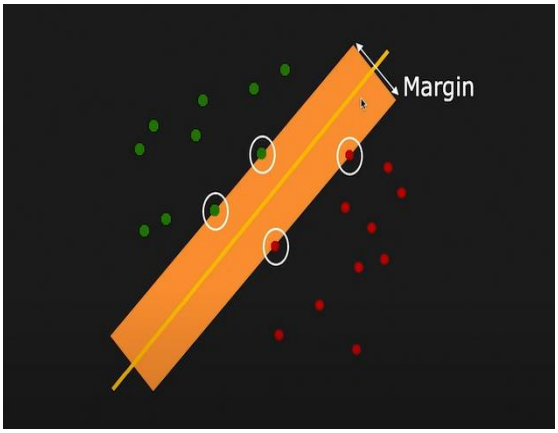



Fig. 9: Margin or safety zone

Marginor safe Zone: The width of that boundary can be increased before hitting a feature point.

The thickness around the decision boundary is known as the margin and the region around the boundary is known as the safety zone SVM classifies based on the given equations:

Compute $d = \omega^T + b$

Here

d=distance

$\rho = width\ of\ safety\ margin$

$d \geq {}^{\rho}/_{2}$ Face

$if$: $d > 0\ and\ d < \rho\ /\ 2$ Probably face

$d < 0\ and\ d > -\rho\ /\ 2$ Probably not a face

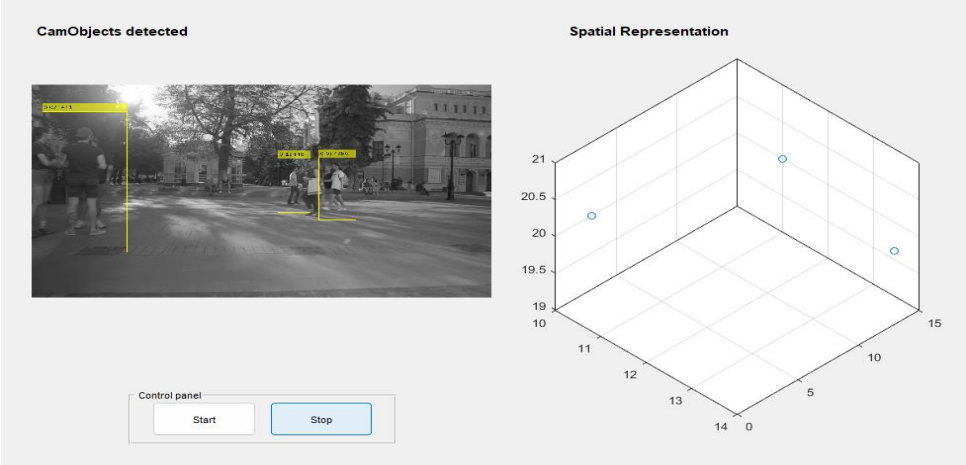
$d \leq -\rho\ /\ 2$ Not a face

## VIII. RESULT

We imagine a room or building with a population count.

Along with the individual, the dot that represents them moves.

It is possible to improve this outcome by integrating tracking.


## REFERENCES

[1.] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 Optical Flow. In Proc. British Machine Vision Conf., 2009.

[2.] P. Widhalm and N. Brandle. Learning Major Pedestrian ¨ Flows in Crowded Scenes. In Proc. Int'l Conf. on Pattern Recognition, 2010.

[3.] B. Wu and R. Nevatia. Improving part based object detection by unsupervised, online boosting. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2007.

[4.] S. Wu, B. E. Moore, and M. Shah. Chaotic invariants of Lagrangian particle trajectories for anomaly detection in crowded scenes. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2010

[5.] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2005.

[6.] P. Decamp, G. Shaw, R. Kubat, and D. Roy. An immersive system for browsing and visualizing surveillance video. In Proc. ACM Multimedia, 2010.

[7.] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. In IEEE Trans. on Pattern Analysis and Machine Intelligence, volume 32, pages 1627–1645, 2010.

[8.] B. Fritzke. A growing neural gas network learns topologies. In Advances in Neural Information Processing Systems, 1995.

A. Girgensohn, D. Kimber, J. Vaughan, T. Yang, F. Shipman, T. Turner, E. Rieffel, L. Wilcox, F. Chen, and T. Dunnigan. DOTS: Support for effective video surveillance. In Proc. Int'l Multimedia Conference, 2007.

[9.] S. Havemann. Generative Mesh Modeling. PhD thesis, Institute of Computer Graphics, Faculty of Computer Science, Braunschweig Technical University, Germany, 2005.

[10.] S. Havemann and D. W. Fellner. Seven Research Challenges of Generalized 3D Documents. IEEE Computer Graphics and Applications, 27(3):70–76, 2007. (special issue on 3D documents).

[11.] S. Havemann, V. Settgast, M. Lancelle, and D. Fellner. 3DPowerpoint - Towards a design tool for digital exhibitions of cultural artifacts. In Proc. Int'l. Symp. on Virtual Reality, Archaeology and Cultural Heritage, 2007.

[12.] V. Nair and J. J. Clark. An unsupervised, online learning framework for moving object detection. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2004. [10] P. M. Roth, S. Sternig, H. Grabner, and H. Bischof. Classifier grids for robust adaptive object detection. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2009.
[13.] O. Sebe, J. Hu, S. You, and U. Neumann. 3D video surveillance with augmented virtual enviroments. In Proc. ACM SIGMM Int'l Workshop on Video Surveillance, 2003.
[14.] V. Settgast, M. Lancelle, S. Havemann, and D. W. Fellner. Spatially Coherent Visualization of Image Detection Results using Video Textures. Proc. Workshop of the AAPR, 2009.
[15.] T. Tekusova, T. Schreck, J. Behr, J. Kohlhammer, and D. Stricker. The IGD-HEyeWall for Visual Analytics – Concepts and Applications. Proc. of Int'l Workshop on GigaPixel Displays & Visual Analytics, (Laboratory Overview), 2008.
[16.] Y. Wang, D. M. Krum, E. M. Coelho, and D. A. Bowman. Contextualized videos: Combining videos with environment models to support situational understanding. IEEE Trans. Visualization and Computer Graphics, 13(6):1568– 1575, 2007.