CENTER FOR
MACHINE PERCEPTION

CZECH TECHNICAL
UNIVERSITY

RESEARCH REPORT

ISSN 1213-2365

Motion Detection as an Application for the Omnidirectional Camera

OMNIVIEWS – Omni-directional Visual System
FP5 RTD – FET Project No: IST–1999–29017

Stefan Gächter

stefan.gachter@ieee.org

CTU–CMP–2001–07

March 9, 2001

Available at
ftp://cmp.felk.cvut.cz/pub/cmp/articles/pajdla/Gaechter-TR-2001-07.pdf

Supervisor: Tomáš Pajdla

This research was supported by the EU Fifth Framework Programme, project OMNIVIEWS No. 1999–29017, by the Czech Ministry of Education under Research Programme J04/98:212300013, decision and control for industry, and by the Grant Agency of the Czech Republic under Project GACR 102/01/0971.

Research Reports of CMP, Czech Technical University in Prague, No. 7, 2001

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Contents

1. Motion Detection

1.1. Introduction

In surveillance applications with omnidirectional cameras it is necessary to transform the omnidirectional images into panoramic ones, which are more suitable for human inspection. This transformation is redundant when using a camera with a space variant imager, i.e. with a log-polar density (SVAVISCA). Compared with a conventional camera, images taken with an omnidirectional camera have lower resolution due to the mapping of the larger field of view to a similar sized imager. The image is in general resolution non-uniform. Additionally, for omnidirectional cameras with space variant imager the resolution can not be better than for cameras with standard imagers and is often inferior for large parts in image due to bigger pixel sizes. Hence the following report investigates by simple simulation if motion detection is possible with such low-resolution images.

The report is organized as follows. The principles of omnidirectional cameras are presented in Section 1.2. Existing systems with an omnidirectional camera performing object detection and tracking are listed in Section 1.4. In Section 1.5 a similar algorithm as used in the existing systems is derived. The moving object detection is done in two steps, first by a temporal change detection presented in Section 1.5.2 and second by a background change detection presented in Section 1.5.2. Both methods use a threshold to discern between foreground and background objects. An automatic threshold selection is derived in Section 1.5.3. Further in this section the influence of the noise on the detection algorithm is discussed. Particularly considered is the case for the SVAVISCA sensor as space variant imager. Summary is given in Section 1.6.

1.2. Omnidirectional Camera

One approach for an omnidirectional camera is the combination of a curved mirror and a conventional camera in order to obtain a large field of view, *see* Figure 1. Such a camera is also known as a catadioptric sensor. Due to the symmetry of the sensor – the optical axis of the camera and the symmetric axis of the mirror coincide – the omnidirectional image in the image plane is best described by polar coordinates. For standard cameras the imagers pixel disposal is Cartesian and the uniform sampling of the imager results in a non-uniform sampling of the field of view. Hence this implicates (i) that the resolution is a function of the radius and (ii) that a mapping from polar to Cartesian coordinates is necessary to obtain panoramic images. Both is considered when using a space variant imager with pixel sizes varying with the radius and with a pixel disposal of equal numbered concentric rings. Instead of adapting the pixel sizes for a given mirror shape to equalize the image resolution, as discussed by Bruckstein and Richardson in [4], an existing imager can be used in combination

with an adapted mirror shape. Chahl and Srinivasan propose in [5] to use an imager with log-polar pixel density. Such an imager is the SVAVISCA [23, 15] showing a similar pixel distribution as the human retina receptors. For the imager retina the pixels are located by rings with equal number but linear increasing size from the centre to the rim as depicted in Figure 2. The linear increase implies a logarithmic mapping in the radius. In the fovea the pixels have a uniform distribution. With a catadioptric sensor composed of such an imager and an adapted mirror – *see* [7] for a possible design – the obtained images have uniform but lower resolution as images taken with conventional omnidirectional cameras because the varying pixel size limits the density; the considered number of pixels in the retina at present is 27720. Due to the manufacturing process of the imager the resolution in the retina cannot be better than in the fovea or as for a standard imager.

In the present case the panoramic images are obtained by simulation. Omnidirectional images taken with a catadioptric sensor composed of a conventional camera and a hyperboloidal mirror are resampled according to the pattern of the SVAVISCA sensor as described by Pajdla and Roth in [18]. Not considering the foveal part, the original images are divided in a number of equal sized sectors where the sectors are divided in linear increasing sized areas. The new images are the intensity mean values for each area spanned by polar coordinates of the centroids.

1.3. Motion Detection

In motion detection the task is to detect a *region of interest* embodied in a *region of awareness*, where the region of awareness, or in terms of the camera geometry, the field of view, is defined as the portion of environment being monitored. The region of interest is in the present case the portion of the environment with activity. For the sake of simplicity and generality, recognition-based detection is not assumed. A region of interest can be therefore a person, an animal, or an artefact; circumscribed with the term *moving objects*.

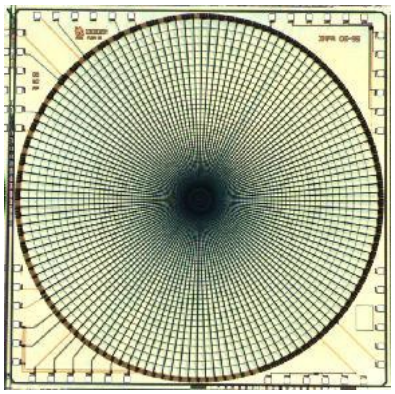
To obtain a maximum region of awareness an omnidirectional camera is used. This sensor provides the image stream necessary for detection and tracking. The detection of moving objects is done by an appropriate algorithm. The possible algorithms are based either on a *difference* or a *statistical* method. If it is assumed that the omnidirectional camera is stationary, when the motion detection can be done by taking the difference between successive images or the difference between the current image and a modelled background image. Existing algorithms for temporal and background difference methods can be found in [12, 10, 8, 27]. Statistical methods accumulate with the ongoing image stream a distribution model for each image point and discern between moving and stationary objects by hypothesis testing. Existing algorithms can be found in [13, 6, 16, 21, 11]. Often in applications both methods are merged to use advantages each.

(a)

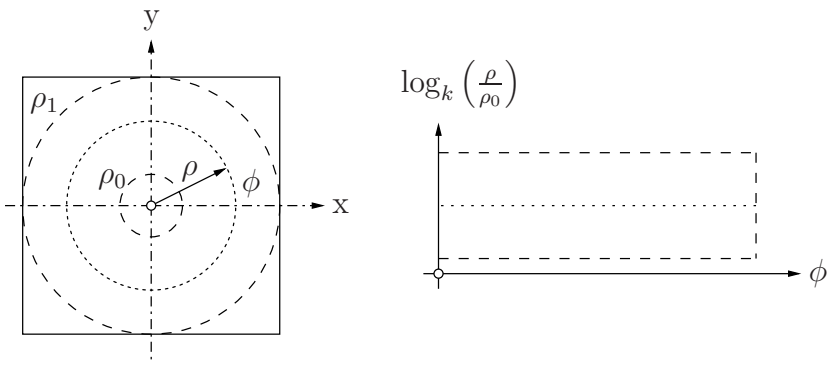
(b)

(c)

Figure 1: The setup of a catadioptric sensor consisting of a conventional camera and a hyperboloidal mirror is depicted in Figure (a). Images taken by such an omnidirectional camera are shown in (b) and (c). Figure (b) depicts the omnidirectional image. Figure (c) depicts the corresponding panoramic image.



(a)



(b)

Figure 2: An imager – similar to the SVAVISCA – with a pixel density of the form log-polar is depicted in (a). Figure (b) depicts the geometry. The fovea $\rho \leq \rho_0$ has a uniform pixel density where the retina $\rho_0 < \rho \leq \rho_1$ a logarithmic one, i.e. the pixel size is linear increasing from the centre to the rim whereas the pixels are arranged by equal numbered concentric rings.

1.4. Existing Systems

Using an omnidirectional camera for moving object detection and tracking is evident. The large field of view permits a stationary camera and it is not necessary to use multiple conventional cameras or to use a pan-tilt camera to cover the same size of field of view. A stationary camera simplifies the detection and tracking algorithms and is most suitable to difference or statistical methods.

In the last years different systems with an omnidirectional camera had been developed. In this section they are briefly presented. All of them are used for surveillance, i.e. detecting and tracking moving persons.

System developed at the Leigh University
The system was developed at the Vision and Software Technology Lab, Leigh University for a military use [3, 2]. The systems goal is to detect and track camouflaged persons in an outdoor environment. For the monitoring a catadioptric sensor is used composed of a paraboloidal mirror and an orthographic camera manufactured by Cyclovision Inc.[1]. The detection is done by background subtraction. The algorithm deals with the omnidirectional images to gain speed. Nevertheless the system also generates for each detected person a perspective panoramic image for human inspection.

System developed at the University of Massachusetts
The system was developed at the Computer Vision Research Laboratory, University of Massachusetts for rescue assistance [1]. The systems goal is to detect fire, exits and persons in a room and track their behaviour. For the monitoring an annular lens and a perspective camera is used. The algorithm for detection and tracking is not further described but it seems that it is based on a background subtraction and that it does work with panoramic images.

System developed at the Nara Institute of Science and Technology
The system was developed at the Graduate School of Information Science, Nara Institute of Science and Technology for surveillance [17]. The system has three different kinds of interfaces: a moving person is detected and tracked by a human observer, a moving person is detected by a human observer and tracked automatically, or a moving person is detected and tracked completely automatically. For the monitoring a catadioptric sensor is used composed of a hyperboloidal mirror and a perspective camera. The algorithm for detection and tracking is based on a background subtraction and does work with perspective panoramic images.

System developed at the Michigan State University
The system was developed at the Computer Science and Engineering Department, Michigan State University for surveillance [9]. The system detects and

1 It seems, that Cyclovision Inc. no longer exists. Former link is replaced by `http://www.remotereality.com/`(12.2.2001).

tracks a moving person and shifts a pan-tilt camera in this direction for identification by face-recognition. For the monitoring a catadioptric sensor composed of a paraboloidal mirror and an orthographic camera manufactured by Cyclovision Inc. is used. The algorithm for detection and tracking is based on a background subtraction and does work with perspective panoramic images.

System developed at the University of Veszprém
The system was developed at the Image Processing and Neurocomputing Department, University of Veszprém for surveillance and smoke detection [14]. For the monitoring an annular lens and perspective camera is used. The algorithm for detection and tracking is based on a background subtraction and does work with panoramic images.

Two conclusions result from the presented systems. First, for all systems the monitoring ends with a human visual inspection. Therefore panoramic images obtained by mapping the omnidirectional images are necessary. One approach for the mapping is to use a catadioptric sensor consisting of a camera with a space variant imager and an appropriate mirror as mentioned in Section 1.2.

Second, all systems use the background substraction method to detect moving objects in a sequence of images. The detection is achieved by taking the difference between the current image and a modelled background image. This method is also suitable for the present case because similar conditions are on hand. A distinction must be performed between *foreground* and *background* objects, where roughly classified the foreground objects are moving and the background objects stationary. The more accurate criterion is formulated as how long an object has been stationary. In the expected scene – a room with natural and artificial illumination change – the foreground objects are persons and the background objects are the furniture and other equipment.

1.5. Concepts

The motion detection algorithm is based on *background change detection*, i.e. the difference method by background subtraction. This assumes that the background model for the expected image sequence is known in advance and that it does not change over time. Such conditions are rarely given for an indoor scene, e.g. when illumination changes occur and objects are moved around. Hence a confident adaption method is required. An *adaption region* must be specified that discerns between foreground and background objects. By definition the background change detection itself yields that distinction but is inappropriate because undetected foreground objects will be falsely adapted to the modelled background image. Therefore the discrimination must be done by a different method.

The *temporal change detection* discerns between moving and stationary objects by comparing consecutive images. But stationary foreground objects are not detected as such and to obtain a similar behaviour as the background change

detection a *temporal change history* must be accumulated. Hence background change detection classifies the scene into foreground and background objects and temporal change detection classifies into moving and stationary objects.

The temporal change history reflects regions in the image sequence that did not change for a certain time and is used as criterion for the distinction between foreground and background objects. Therefore the *temporal change region* corresponds to the *adaption region*.

An algorithm with these concepts is presented by Huwer and Niemann in [12]. This algorithm gives also the structure for the following – schematically depicted in Figure 3 – but is different in certain parts to consider the conditions when using an omnidirectional camera, especially when using the SVAVISCA imager.

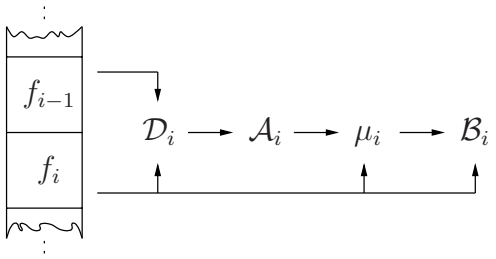


Figure 3: The different steps for motion detection by background subtraction. The background change region $\mathcal{B}_i$ is obtained by comparing the current image f_i with the background model μ_i. The model is adapted according the current image confined to the adaption region $\mathcal{A}_i$. This region is the union of a limited number of temporal change regions $\mathcal{D}_i$ obtained by differencing consecutive images.

In the present case a sequence of images $f_0, \ldots, f_{i-1}, f_i$ taken by an omnidirectional camera is available. Here f_0 is the initial image and f_i is the current image of the sequence. All images are defined on the domain $\mathcal{I} \subset \mathbb{N}^2$. An image point $f_i(m,n)$ with $(m,n) \in \mathcal{I}$ is defined on the range $\mathbb{G}$, where $\mathbb{G}$ is the greyscale with 256 normalized and discrete values. Further m respectively n denotes the vertical respectively the horizontal image coordinate.

Weighted Accumulation

The weighted accumulation is a moving average with a constant adaption length and appears in the temporal as the background change detection but for different reasons. In the first case because the difference of two consecutive images when a moving object is present results in the moving contour. The complete shape, necessary for the temporal change history, is obtained by accumulating the difference images. In the second case the weighted accumulation is used for the adaption so that the background is modelled by an average image. Therefore changes in the background are adapted with a constant rate during the ongoing process.

One function ψ that accomplishes the weighted accumulation is given by

$$\begin{aligned} \hat{f}_i &= \psi(\hat{f}_{i-1}, f_i, \tau) \\ &= \hat{f}_{i-1} \mathrm{e}^{-\frac{1}{\tau}} + f_i(1 - \mathrm{e}^{-\frac{1}{\tau}}) \ , \end{aligned} \tag{1}$$

where $\hat{f}_i$ is the accumulation image, f_i is the current image, and $\tau \in \mathbb{N}$ is the accumulation length. If the accumulation length $\tau \to 0$ when the accumulated image corresponds to the current image. Reciprocally, if $\tau \to \infty$ when the accumulated image rests unmodified. In the present case an image f_i is defined on the domain $\mathcal{I}$. However, the accumulation can be restricted to a subset $\mathcal{R} \subset \mathcal{I}$ and equation (1) becomes

$$\begin{gathered} \forall (m,n) \in \mathcal{I} : \psi(\hat{f}_{i-1}, f_i, \tau, \mathcal{R})(m,n) = \\ \begin{cases} \psi(\hat{f}_{i-1}(m,n), f_i(m,n), \tau) & ; \quad (m,n) \in \mathcal{R} \\ \hat{f}_{i-1}(m,n) & ; \quad (m,n) \in \mathcal{I} \setminus \mathcal{R} \end{cases} . \end{gathered} \tag{2}$$

1.5.1. Temporal Change Detection

The temporal change detection is carried out in two steps. First the accumulation image d_i is updated by computing $d_i = \psi(d_{i-1}, |\delta f_i|, \tau_d, \mathcal{I})$, where $|\delta f_i| = |f_i - f_{i-1}|$ is the absolute value of the difference between the consecutive images f_i and f_{i-1}. In a second step the temporal change region $\mathcal{D}_i$ is identified by comparing d_i with the threshold ε_d.

$$\mathcal{D}_i = \{(m,n) \in \mathcal{I} \mid d_i(m,n) > \varepsilon_d\} \tag{3}$$

Thus the temporal change detection depends on two parameters:

1. τ_d describing the temporal range for accumulation of the absolute difference images $|\delta f_i|$.
2. ε_d controls the sensitivity to identify the temporal change region $\mathcal{D}_i$.

1.5.2. Background Change Detection

The identification of the background change region $\mathcal{B}_i$ is done by comparing the absolute difference between the current image f_i and the mean background image μ_i with the threshold ε_b.

$$\mathcal{B}_i = \{(m,n) \in \mathcal{I} \mid |\mu_i(m,n) - f_i(m,n)| > \varepsilon_b\} \tag{4}$$

Background Adaption

For the background model update the reliable determination of the correct adaption region $\mathcal{A}_i$ is essential. The region $\mathcal{A}_i$ is defined by means of the temporal

change history as the region that did not change for a specified period of time and therefore seems to represent background. The adaption region is specified – on the basis of the temporal change regions $\mathcal{D}_k$ where $k \leq i$ – as the image region whose image points were not part of any $\mathcal{D}_k$ for the last $\eta \in \mathbb{N}$ images, where η is called the depth of adaption.

$$\mathcal{A}_i = \mathcal{I} \setminus \bigcup_{k=\max(0,i-\eta)}^{i} \mathcal{D}_k \tag{5}$$

The adaption of the background image is realized by accumulating the current background representation μ_i with f_i restricted to the adaption region $\mathcal{A}_i$.

$$\mu_i = \psi(\mu_{i-1}, f_i, \tau_b, \mathcal{A}_i) \tag{6}$$

The accumulation length $\tau_b \in \mathbb{R}$ must be chosen according to the expected types of image changes. The adaption takes place only in region where the number of the processed images is larger than the depth of adaption η. Thus the background change detection depends on three parameters:

1. τ_b describing the temporal range for background adaption in the adaption region $\mathcal{A}_i$.

2. η weighting how long a region in an image sequence is considered as a moving object.

3. ε_b controls the sensitivity to identify the background change region $\mathcal{B}_i$.

Therefore five parameters depending on the expected frame rate and image changes have to be specified. To consider is the parameters interdependence. The accumulation length τ_d has to be chosen together with the threshold ε_d such that a connected adaption region $\mathcal{A}_i$ is obtained. Consequently for fast frame rates τ_d must be large and reciprocally for slow frame rates τ_d must be small. Whereas *fast* and *slow* is relative to the expected rate of change in the image sequence. Another criterion for the parameters is the noise present in the images. The noise appears as erroneous detected regions in $\mathcal{D}_i$ and they have to be minimized by the correct selection of the threshold. Its computation is described in Section 1.5.3.

The adaption depth η depends on the chosen adaption length τ_b. For small η only moving objects are considered as foreground objects; stationary foreground objects are adapted to the background model. Hence for a small τ_b, necessary when rapid background change is expected, the foreground objects are entirely defined by η. The threshold ε_b depends on the noise as η_d does and is consequently determined in the same manner. For a further discussion about the parameter choice see Section 2.2.

Background Model Reset

The so far presented algorithm has some conceptual lacks. Mostly the strong interdependence of the different parameters. Therefore the algorithm is best applicable for fast frame rates when the background is slowly varying. But this is hardly given for an indoor scene in particular when artificial illumination changes. To robusten the algorithm when the background activity abruptly increases an immediate update of the background model is done and the adaption region $\mathcal{A}_i$ is restricted to the last background change region $\mathcal{B}_{i-1}$. The background activity is measured by computing the least median of squares of the absolute difference image $|\delta f_i|$. The mean background image μ_{i-1} is updated by the multiplication with the ratio $r_i = f_i / f_{i-1}$ between consecutive images. Regions in μ_{i-1} corresponding to foreground objects are updated differently because the ratio at these regions does not represent the true ratio for the occluded background. The regions in r_i for those hold $(m,n) \in \mathcal{B}_{i-1} \setminus \mathcal{I}$ are modelled by interpolation. So when the background activity exceeds the given threshold ε_a the mean background image is updated as follows,

$$\mu_{i-1}(m,n) = \begin{cases} r_i \mu_{i-1}(m,n) & ; \quad (m,n) \in \mathcal{B}_{i-1} \setminus \mathcal{I} \\ \text{Regions in } r_i \text{ are filled by interpolating from their borders.} & ; \quad (m,n) \in \mathcal{B}_{i-1} \end{cases} . \tag{7}$$

The fill operation uses an interpolation method based on Laplaces equation. This method results in the smoothest possible fill, given the values of the ratio r_i on the region borders[2].

1.5.3. Thresholds

The thresholds ε_d and ε_b for the change detection are proportional to the noise present in the difference images. The noise is modelled by a normal distribution $N(0, \sigma^2)$ with zero mean and described by the standard deviation σ. Further details are discussed later. The standard deviation is estimated by the least median of squares of the absolute difference images. The same approach is presented by Rosin in [20, 19] in case of motion detection but using edge images.

Because the difference images contain not only noise but also foreground objects a robust estimation technique for the standard deviation is required. The least median of squares rests unaffected while the moving part in the image constitutes less than the half of all image points. As derived in [22] it is computed by determining the centroid of the shortest range in the sorted sample $\{a_k\}_{k=1}^{m \cdot n}$, i.e. $a_1 \leq \ldots a_k \leq \ldots \leq a_{m \cdot n}$ includes all image points of the absolute difference image.

2 For the interpolation function refer to the *MatLab* function `roifill.m` in the *Image Toolbox*.

$$
\begin{aligned}
a_{\mathrm{LMS}} &= \tfrac{1}{2}(a_{k_{\min}} + a_{h+k_{\min}}) \\
\text{where} \quad k_{\min} &= \arg\min_{k}(a_{k+h} - a_k) \quad \text{and} \quad h = \left[\tfrac{m \cdot n}{2}\right] + 1
\end{aligned}
\tag{8}
$$

Differencing noisy images followed by taking the absolute values produce the normal distribution $2N(0, 2\sigma^2)$ for positive values only. The least median of squares for this distribution is 0.3372 and thus the standard deviation is determined by $\sigma = a_{\mathrm{LMS}}/0.3372$. When thresholding at ε the probability of incorrectly classifying noise as foreground is given by the error function

$$
P(|.| > \varepsilon) = 1 - \frac{2}{\sqrt{\pi}} \int_0^{\frac{\varepsilon}{\sqrt{2}\sigma}} \mathrm{e}^{-t^2} dt \ . \tag{9}
$$

If assuming a probability of 1‰ then the threshold is given by $\varepsilon = 2.576\sigma$ and the thresholds ε_d and ε_b are computed as follows:

$$
\begin{aligned}
\varepsilon_d &= \tfrac{2.576}{0.3372} a_{\mathrm{LMS}} \quad \text{where} \quad a = \{|\delta f_i(m,n)|\} \quad \forall (m,n) \in \mathcal{I} \\
\varepsilon_b &= \tfrac{2.576}{0.3372} a_{\mathrm{LMS}} \quad \text{where} \quad a = \{|\mu_i(m,n) - f_i(m,n)|\} \quad \forall (m,n) \in \mathcal{I}
\end{aligned}
\tag{10}
$$

Noise

Different noise sources disturb the images. These are mainly the photonic noise, the thermal noise, and the quantization noise. For simplicity all are merged in one noise source and modelled by a normal distribution.

For the SVAVISCA sensor as a space variant imager the noise is expected to varying according to the pixel size. *At the moment no specific data is available* but simulating the sensor demonstrates a varying repartition. The simulation is performed by taking images with a catadioptric sensor using a hyperboloidal mirror and a conventional camera. The omnidirectional images are resampled with respect to the pattern of the SVAVISCA pixel density resulting in panoramic ones as described by Pajdla and Roth in [18]. (For further information about the used *MatLab* files in this section refer to Section A.1 and Section A.2.1.)

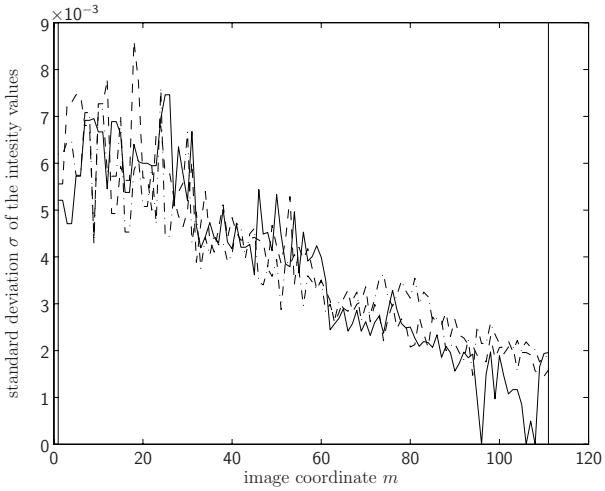
To examine the noise the standard deviation is used as a criterion. Computing $\sigma(m, n)$ for each image point in a sequence of simulated images results in a non-uniform repartition. The standard deviation in the panoramic image is decreasing with increasing image coordinate m. Referring to the SVAVISCA sensor the relation is inverse proportionally, i.e. decreasing standard deviation with increasing pixel size. The result for a sequence of 60 images is depicted in Figure 4. Figure 4(a) represents the graph of the standard deviation at three different image positions emphasized as white lines in 4(b). The decreasing standard deviation in the graph corresponds to the decreasing intensity values

in the image 4(b). The black regions in the panoramic image corresponds to saturation due to artificial and natural illumination as visible in the mean background image in 4(c). These regions have consequently a standard deviation equal zero. Further the progression of the standard deviation is not continuous. Four steps are present at the positions $m \in \{31, 61, 79, 91\}$. They are visible as bands with different intensities in the standard deviation image.

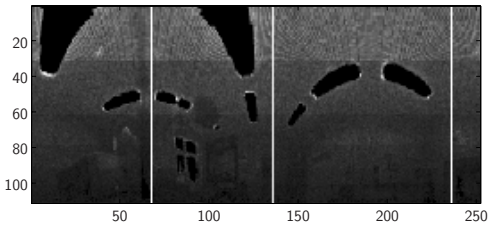
To further investigate the influence of the resampling a sequence of noise images is used. The noise in the omnidirectional images is modelled by a uniform distribution. The panoramic images are obtained by resampling. The result for the standard deviation $\sigma(m, n)$ of a sequence with 60 images is depicted in Figure 5. Figure 5(a) represents, as for the indoor scene, the graph for the standard deviation at three different image positions emphasized as white lines in 5(b). The horizontal dotted limit indicates the standard deviation for the uniform noise in the omnidirectional images, i.e. $\sigma = \sqrt{1/12} = 0.2887$. The decreasing standard deviation in the graph corresponds to the decreasing intensity values in the image 5(b). Referring to the SVAVISCA sensor the standard deviation is decreasing with increasing pixel size in a range of $0.25 \ldots 0.05$. These values are lower than the value for the imposed uniform distribution because the resampling associates to an image point in the panoramic image a mean value taken over a region in the omnidirectional image. The regions correspond to the varying pixel size of imager. The averaging results in a decrease of the standard deviation because the considered regions increase. Five different bands occurs at the same positions as for the indoor scene. They are visible as changes of the intensity values in image 5(b) and have the sizes from top to bottom $\delta m = \{31, 30, 18, 12, 20\}$. *It is not yet understood why the averaging is not continuous with respect to the pixel size.*

For the temporal and background change detection the noise is used as the criterion to determine the thresholds. Using the same threshold for the entire image results in false detected foreground objects. Such objects are present as spurious particles. In Figure 6(b) the temporal change region $\mathcal{D}_i$ when comparing the absolute difference image $|\delta f_i|$ with the threshold $\varepsilon = 2.567\sigma$, where σ is the standard deviation of $|\delta f_i(m, n)|$ for $\forall (m, n) \in \mathcal{I}$ is depicted. The particle size is increasing with increasing standard deviation of the noise in the panoramic image. Figure 6(a) represents the histogram for the particle size where the size is measured by counting the number of connected pixels.

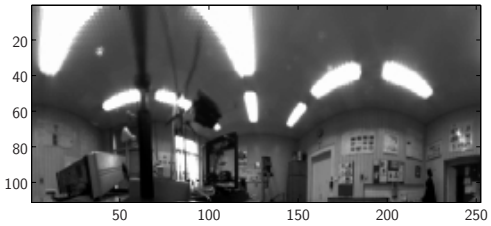
Increasing the global threshold eliminates the spurious particles but also suppresses useful information. Reminding the statistical method where for each image point a distribution model is accumulated and an individual threshold for each image point can be determined. In the present case the global threshold is deduced from spatial information in the difference image and not from temporal information in the sequence. Therefore image points of foreground objects with intensity values falling between the individual thresholds and the global threshold are not detected even though it would be possible. In especially foreground objects in the lower part of the image having the same size as the spurious parti-



(a)

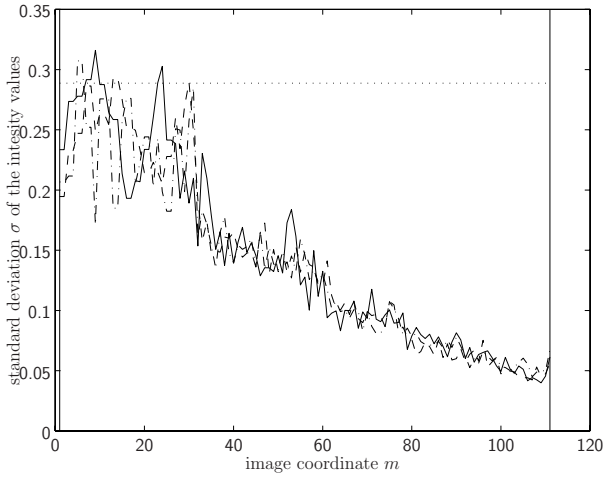


(b)

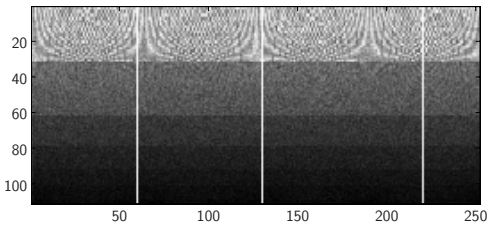


(c)

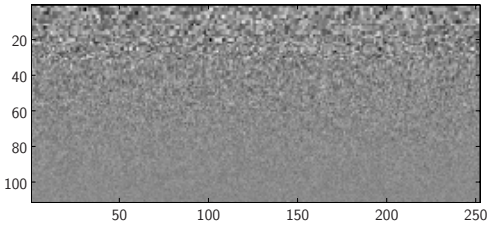
Figure 4: The standard deviation $\sigma(m, n)$ of the panoramic images for a sequence of 60 omnidirectional background images is depicted in (b). The standard deviation along the vertical image coordinate m is depicted in (a). Three different positions $n \in \{68, 136, 236\}$ are selected in (b), where the origin in (b) is at the upper left corner. The first line corresponds to the *solid* line in (a), the second to the *dashed*, and the third to the *dash dotted*. In (c) is depicted the mean of the sequence of background images. (Image sequence source `/home.stud/qqgechte/omni/movie/simulation/pan/movie_8` frame no. 139-199).



(a)



(b)



(c)

Figure 5: The standard deviation $\sigma(m, n)$ of the panoramic images for a sequence of 60 omnidirectional images with uniform noise is depicted in (b). The standard deviation along the vertical image coordinate m is depicted in (a). Three different positions $n \in \{60, 130, 220\}$ are selected in (b), where the origin in (b) is at the upper left corner. The first line corresponds to the *solid* line in (a), the second to the *dashed*, and the third to the *dash dotted*. In (c) is depicted the mean of the sequence of noise images. (Image sequence source `/home.stud/qqgechte/omni/movie/simulation/noise/pan/movie_1` frame no. 0-59).

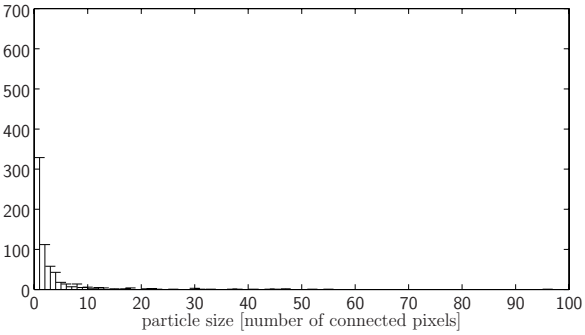
cles in the upper part are getting lost. Thus the image is split along the vertical image coordinate and each subimage is treated separately. The subimages correspond to the observed bands in Figure 5(b). In Figure 7(b) the temporal change region $\mathcal{D}_i$ when splitting the absolute difference image $|\delta f_i|$ is depicted. The splitting takes place at the positions of the five observed bands. Each part is considered as an independent image with an own threshold $\varepsilon_j = 2.567\sigma_j$ for $j \in 1, \ldots, 5$, where σ_j is the standard deviation of each independent image. The independent thresholding results in spurious particles for the lower part of the panoramic image. Furthermore less particles appear in the upper part and their size is slightly minimized. Comparing the histograms Figure 6(a) and Figure 7(a) the increase of the number of spurious particles is evident. The sensitivity for change detection is increased and is adapted to the variation in noise.

For the automatic threshold selection a normal distribution for the noise is assumed. Two histograms for absolute difference images are depicted in Figure 8. Only a clipping of a possible range $[0, 1]$ containing 256 bins is represented. The first corresponds to $|\delta f_i|$ for the temporal change detection and the second to $|\mu_i - f_i|$ for the background change detection. As depicted by the current image f_i in Figure 8(a) no foreground object is present in the scene. Therefore the differences correspond to noise. Both histograms are bounded at zero and falling off with increasing difference values. They corresponds not to a normal distribution as assumed. For the temporal change detection the differences are in the range of the greyscale values. Thus the differences appear in the histogram at bins that are multiples of $1/255 = 0.0039$. For the background change detection the difference values are spread because the background adaption results in a mean background image with intensity values in the range of $\mathbb{R}$. An erroneous peak is present at zero. This corresponds to the number of image points in saturated image regions with zero standard deviation as visible in Figure 4(b). In both cases the performance of the least median of squares computation is affected.

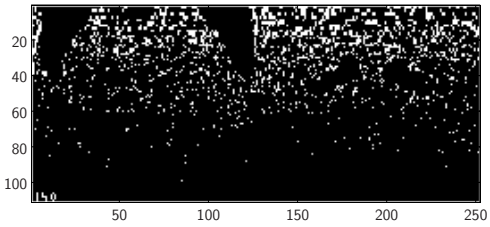
Due to the discrete intensity values of the images the automatic threshold selection fails when the noise distribution is too narrow. In such a case more than the half of the difference values is zero. Consequently the least median of squares is likewise zero instead to be greater and it is not possible to estimate correctly the standard deviation. Further the least median of squares is a distribution independent criterion and an erroneous peak that does not correspond to an outlier does influence the result. In the presence of saturated image regions the standard deviation is underestimated. For the temporal as for the background change detection the threshold must be increased. Therefore a minimal threshold is added to the noise dependent threshold given by equation (10) and the adjusted thresholds are computed as follows,

$$\begin{aligned} \varepsilon_d &= \varepsilon_{d\min} + 2.576\sigma_d \ , \\ \varepsilon_b &= \varepsilon_{b\min} + 2.576\sigma_b \ . \end{aligned} \tag{11}$$

A more sophisticated solution for the discrete case is to compute the standard

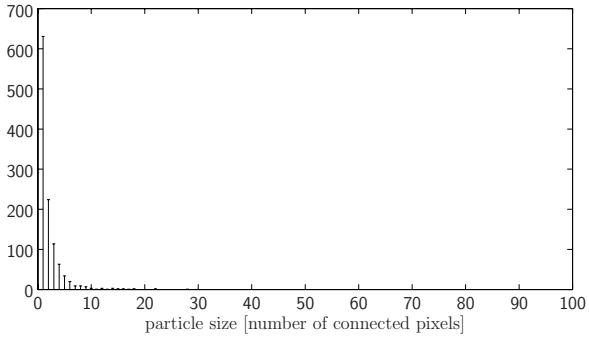


(a)

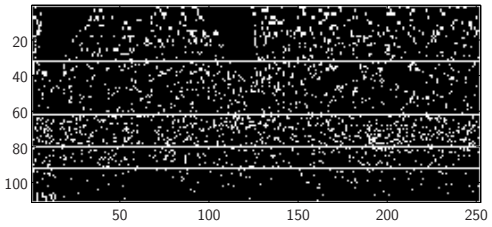


(b)

Figure 6: In (b) is depicted the thresholded image of the absolute difference image $|\delta f_i|$. The threshold is $\varepsilon = 2.567\sigma$, where σ is the standard deviation of $|\delta f_i(m,n)|$ for $\forall (m,n) \in \mathcal{I}$. In (a) is depicted the corresponding histogram of the particle sizes in the thresholded images. (Image source `/home.stud/qqgechte/omni/movie/simulation/pan/movie_8` frame no. 140).



(a)



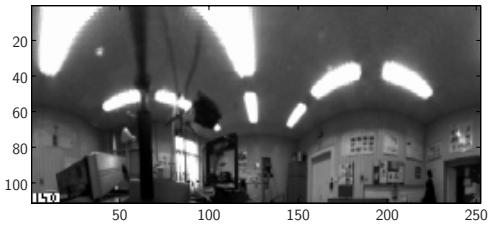
(b)

Figure 7: The thresholded and merged subimages of the absolute difference image $|\delta f_i|$ is depicted in (b). The thresholds are $\varepsilon_j = 2.567\sigma_j$, where σ_j is the standard deviation for each subimage when $j \in 1, \ldots, 5$. The corresponding histogram of the particle sizes in the merged thresholded subimages is depicted in (a). (Image source `/home.stud/qqgechte/omni/movie/simulation/pan/movie_8` frame no. 140).

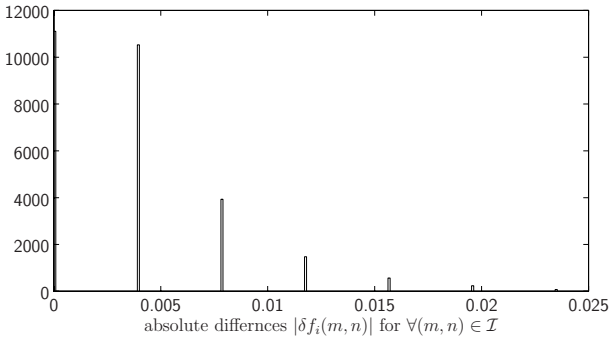
deviation directly when no foreground object is present. In the peak case an appropriate test – regions with zero standard deviation – must exclude the values for the least median of squares computation.

1.6. Summary

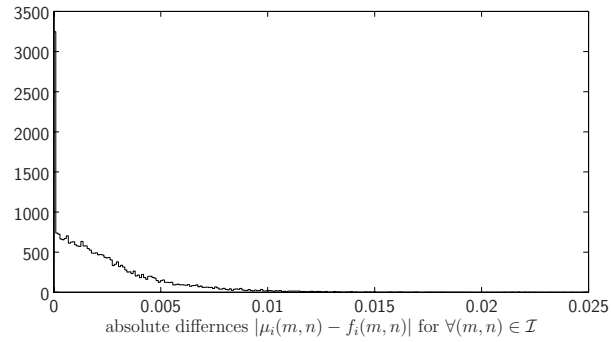
This section has presented the derivation of the motion detection algorithm for a catadioptric sensor. The algorithm performs background change detection, where the background model is adapted to stationary objects during the ongoing process. To robusten the adaption a temporal change detection is used to discern between moving and stationary objects. Further for fast background changes a reset method for the algorithm is derived. The thresholds for the temporal and background change detection are determined depending on the noise in the images, where the noise is characterized by the standard deviation estimated with the least median of squares. A minimal threshold has been introduced because the estimation fails when the noise tends to zero or saturated regions in the images are present. In the case where the panoramic images are simulated a non-uniform noise repartition results. Therefore it is necessary to adapt the threshold locally for that the detection sensitivity rests constant over the entire image.



(a)



(b)



(c)

Figure 8: The histograms for absolute difference images. The histogram for $|\delta f_i|$ in case of the temporal change detection is depicted in (b) and the histogram for $|\mu_i - f_i|$ in case of the background change detection is depicted (c). In (a) is depicted the current image f_i. (Image source `/home.stud/qqgechte/omni/movie/simulation/pan/movie_8` frame no. 140).

2. Simulation

2.1. Introduction

The kernel of the motion detection algorithm derived in Section 1.5 is not application specific. Hence the algorithm suits to a vast range of problems. In the present case the application is motion detection in an indoor scenario. Moving persons, moving and moved objects and varying illumination in account to artificial as natural light sources can be expected. To handle are these events the algorithm offers a set of parameters. That are, for the temporal and the background change detection, the accumulation lengths ε_d and ε_b, the minimal thresholds $\varepsilon_{d\min}$ and $\varepsilon_{b\min}$, and the adaption depth η. Their specification is done with the help of the simulation. The resulting reference values simplify the setup for the latter experiments. Further the performance of the algorithms extension is tested with the simulation, i.e. the reset procedure in case of abrupt illumination change. An additional parameter to specify is therefore the threshold for the background activity ε_a.

The size of the minimal detectable object is as well of interest. The motion detection algorithm does not modify the image quality. Therefore the limit is given by the image resolution and noise. In case of the resampled images both have a non-uniform repartition and it is necessary to treat the image by subdivided regions.

The section is organized as follow, the setup of the simulation is presented in Section 2.2.1. In Section 2.2.2 are discussed the results of a simulation with a image sequence, where the background update is treated in more details. Conclusions are given in Section 2.3.

2.2. Motion Detection

The different parameters for the temporal change detection depend on the expected frame rate and object motion as discussed in Section 1.5.2. The frame rate specifies the temporal succession of the image sequence. The rate of change in the scene – the moving objects velocity – has to be regarded with respect to the frame rate. In case of the temporal change detection the moving contour is proportional to that relative velocity; a slow frame rate and fast rate of change leads to a large contour, and a fast frame rate and slow rate of change to a small one. Hence to obtain a constant temporal change region $\mathcal{D}_i$ the accumulation length τ_d must be inverse proportional to the relative velocity.

For the background change detection the parameters are coupled. The adaptation depth η is proportional to the expected duration for a foreground resting state. Further the accumulation length τ_b is proportional to the velocity of background change. Therefore fixing one parameter restricts the range of the other. In case of fast background variation and long expected rest states the adaption region $\mathcal{A}_i$ is small and the background model is not adapted correctly.

A remedial is the background model update as presented in Section 1.5.2.

The remaining three parameters depend on the noise present in the difference images. For simplicity the minimal thresholds $\varepsilon_{d\min}$, $\varepsilon_{b\min}$, and ε_a correspond approximately to $6 \cdot \sigma_{\min}$, where $\sigma_{\min}$ is the smallest standard deviation of the noise present in the difference images. The smallest standard deviation is expected in regions generated with the largest pixel sizes as discussed in Section 1.5.3 and visible in Figure 6.

2.2.1. Setup

The image sequence for the simulation is generated from images taken by a conventional camera [26], a hyperboloidal mirror [24], and with conventional data acquisition equipment [25]. The process is controlled with the help of *MatLab* batch files and functions. For further information about them refer to Section A.1.1 and Section A.2.

The catadioptric sensor is placed on a level of about 1m in the centre of the monitored room. The orientation of the sensor results to a field of view that includes the upper part of the room. The omnidirectional image sequences is recorded with the fastest possible frame rate. With $MatLab$ the rate of approximately 5 frames per second is attained. The intensity values of the images are represented by the normalized greyscale with 256 steps. Different scenes are monitored. The sequences are resampled to obtain panoramic images and merged in one array to speed up the access time during the simulation. Therefore the sequence length is restricted to 200 images. The initial values for the algorithm are the mean background image μ_0 and the image f_0. The latter corresponds to the first recorded image in the sequence. The former is obtained by computing the mean value for each image point over a sequence of 60 images. During this sequence no foreground object is present in the scene. This initial mean background image will be adapted with the ongoing process.

The parameters are chosen for fast moving objects and slow background variations. For the attained frame rate walking persons in naturally illuminated environment are expected. Artificial illumination is constant or varying in an abrupt way, i.e. switched on or off. The Table 1 summarize the values deduced by experience. The images are split in bands with the widths $\delta m = \{31, 30, 18, 12, 20\}$ pixels to equalize the sensitivity, where the images have originally a width of 252 pixels and a height of 111 pixels.

2.2.2. Results

Depending on the chosen accumulation lengths and adaption depth the algorithm needs time to adjust the initial background model. For the current parameters the duration corresponds to about one third of the sequence length. During this period no reliable detection is possible. After the adjustment spurious detections are mainly due to an outdated background model. One problem

accumulation length τ_d	0.002	part of unit sequence step
adaption depth η	25	number of frames
minimal threshold $\varepsilon_{d\min}$	0.0196	part of normalized greyscale
accumulation length τ_b	5	part of unit sequence step
minimal threshold $\varepsilon_{b\min}$	0.0098	part of normalized greyscale
background activity threshold ε_a	0.0196	part of normalized greyscale

Table 1: Parameters for the temporal and background change detection.

is when the assumptions do not hold and persons are stationary for a too long time period. Hence foreground objects are falsely added to background. Another problem is when objects as equipment and furniture are displaced in the scene. The objects at the new location are not fast enough adapted and are therefore falsely detected as foreground objects. These two problems are inconsistent and are only concurrently solvable by recognition based detection; as it is for correctly, but not desirable, detected moving shadows. Consequently the algorithm with the chosen parameters performs mediocre when both cases are present in the image sequence[3].

The consecutive steps – at one position in the image sequence – from the current image to the detected moving objects are depicted in Figure 9. The images are panoramic and thus the left and right image boarder are mapped to the same place in the scene. Further they are not perspective, hence the geometry of the room is not preserved. Figure 9(a) represents the scene. In the current image two persons are observable; one is in the centre in front of the door and the second one is on the right. In Figure 9(b) the absolute difference image $|\delta f_i|$ between the current and the precedent image is depicted. For both persons their moving contours are visible. The sizes indicate a slow motion for the person in front of the door and a fast motion for the person on the right side of the image. The image is split into the different parts and for each part the threshold ε_d is computed. The thresholded parts are merged. The result is the temporal change region $\mathcal{D}_i$ depicted in Figure 9(c). The horizontal lines indicate the borders of the split parts. Further the image is slightly smoothed by morphological erosion and dilation[4]. On the basis of this image the adaption region $\mathcal{A}_i$ is updated. The temporal change history for the last 25 images is depicted in Figure 9(d). Black regions in the image correspond to robustly detected foreground objects. These regions are excluded in 9(a) for the adaption of the background model that is the mean background image as depicted in Figure 9(e). Subtracting this image 9(e) from the current image 9(a) and taking the absolute values results in Figure 9(f). The absolute difference $|\mu_i - f_i|$

3 Different animations are available at `/home.stud/qqgechte/omni/movie/animation/` which illustrate this discussion.

4 The structuring element is the minimal possible disc.

contrary to 9(b) results in complete shapes. The image 9(f) is split into the different parts and for each part the threshold ε_b is computed. The thresholded parts are merged. The result is the background change region $\mathcal{B}_i$ depicted in Figure 9(g). The horizontal lines again indicate the borders of the split parts. The image is slightly smoothed by morphological erosion and delation. Both persons, as the shadow for the person at the door, are clearly detected. Labelling this image by seeking connected regions results in Figure 9(h). Finally two objects are detected, both enclosed in a white frame, one is the person standing at the door and its shadow, the other is the person moving in the right side of the image.

Background Update

To minimize the adaption time for the background model in case of abrupt and large changes the algorithm performs an update of the mean background image and the adaption region. Without this the algorithm would fail after an interrupt for a certain period depending on the chosen parameters. These are fitted to the normal processing and therefore the period is at least equal to the initialization time.

In Figure 10 the scene before and after a abrupt illumination change is depicted; the person in the room turns off the lamps in the room. After the change the scene is still illuminated by natural light. Figure 10(a) shows the detected objects before the change. The object detected immediately after the change is shown in Figure 10(b). It is the same person at the same place. Without the update it is not possible to distinguish between foreground and background objects. The background subtraction would be performed with a slightly adapted model that is similar as the image depicted in Figure 10(c) and the scene depicted in 10(b). The changes are too large and the whole scene would be detected as a foreground object. The mean background images before and after are depicted in Figure 10(c) and 10(d). The regions of the background occluded by a foreground object during the change are smoothly approximated. The adaption region $\mathcal{A}_i$ is set to the last background change region $\mathcal{B}_{i-1}$ as visible in Figure 10(e) and Figure 10(f). Hence no erroneous detected temporal change region is associated to the history.

The main problem in this approach is how to model the occluded background. The algorithm interpolates these regions from the border to the center of the shapes where the regions are identical with the last background change region $\mathcal{B}_{i-1}$. The interpolation leaves still some error appearing when the former occluded regions are left by the moving objects. In Figure 10(g) the mean background image 10 images after the interrupt is depicted. The result of the background change detection with this model is depicted in Figure 10(h). A moving person and a spurious object at his precedent position are detected.

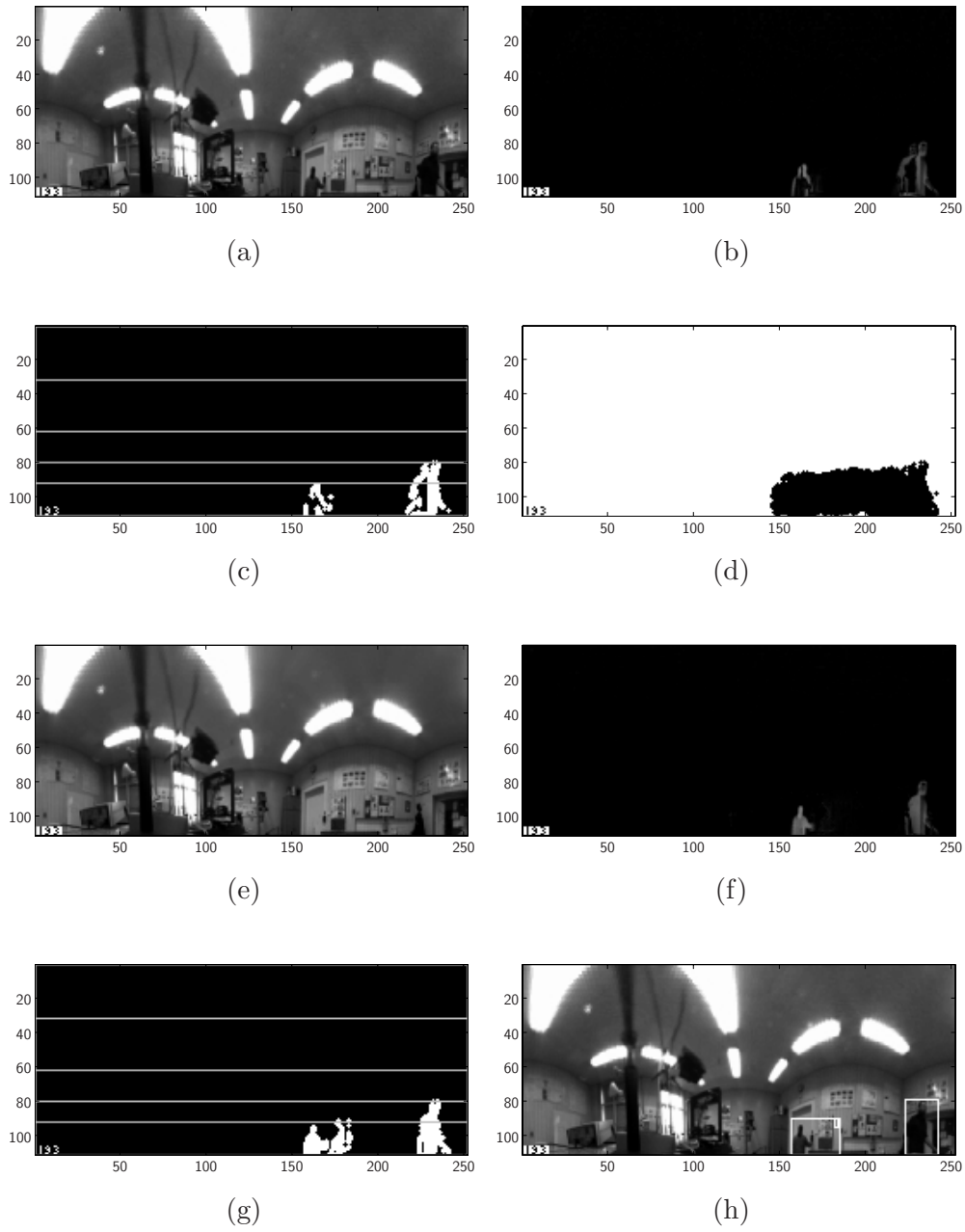


Figure 9: Phases of the algorithm – at one position in the image sequence – how to arrive from the current image to the detected moving objects. (Image source `/home.stud/qqgechte/omni/movie/simulation/pan/movie_9` frame no. 193).

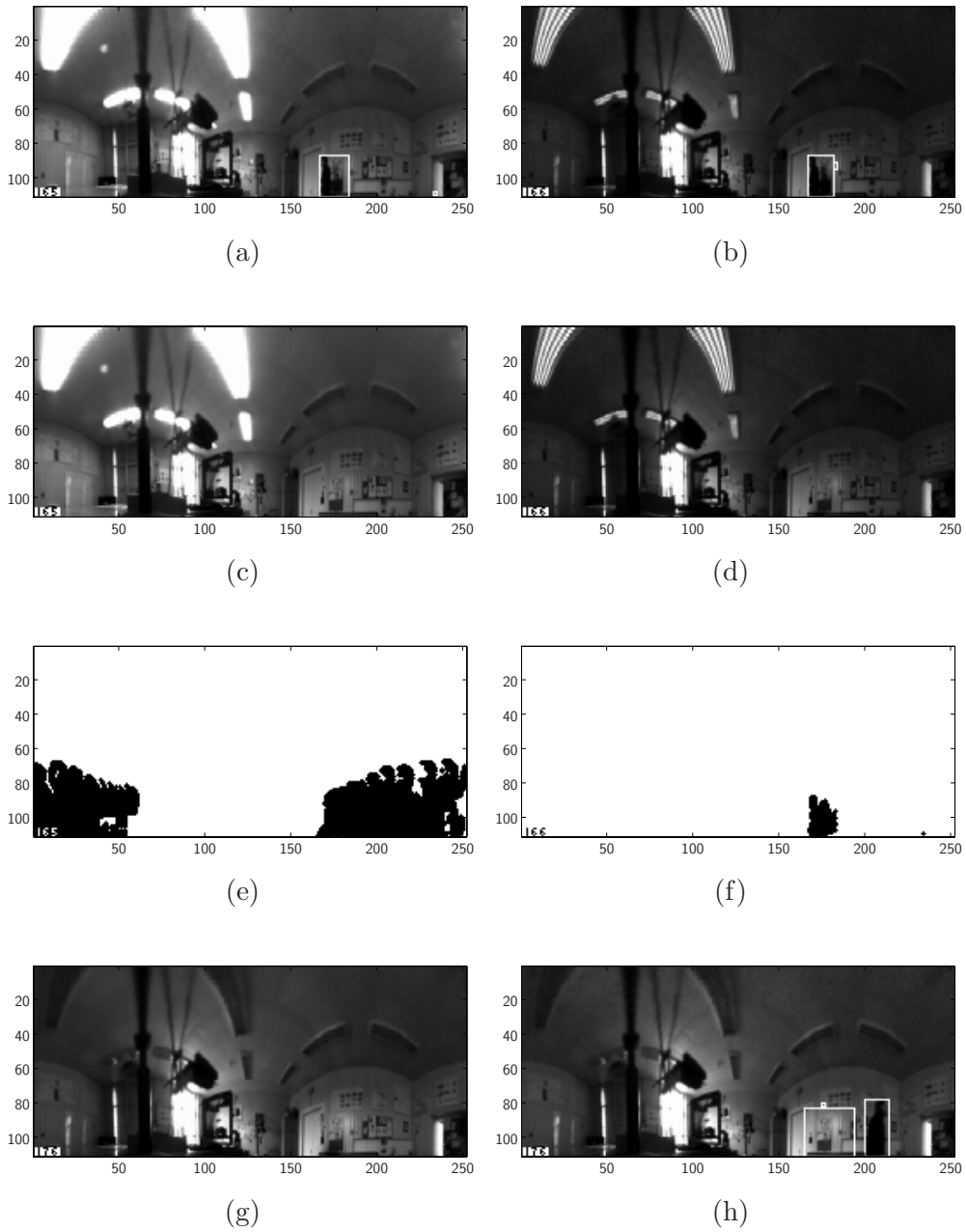


Figure 10: Images of the background model and the temporal change history in case of abrupt illumination change. (Image source `/home.stud/qqgechte/omni/movie/simulation/pan/movie_12` frame no. 193).

2.3. Conclusion

This section has presented the simulation of the motion detection in an image sequence. The image sequence consists of panoramic images obtained by resampling the omnidirectional images with the pattern of the SVAVISCA imager. Therefore the panoramic images have a similar resolution as images should have when recorded with the SVAVISCA camera. The SVAVISCA images have a lower resolution compared to conventional cameras but the simulation shows that motion detection in a room with the size of $40m^2$ is possible. A minimal object is not specified. The performance is not restricted by the object size but rather by the strong dependence between the expected scenario and the algorithm parameters. Once the parameters are chosen for a scene, e.g. for fast moving objects and slowly varying background, the algorithm performs poorly when the assumption changes, e.g. instead of fast slowly moving objects are present. However the algorithm suits to a vast range of problems but each has an independent sets of parameters. The simulation shows that the problem is eased with the reset procedure of the background model.

Because of the automatic threshold selection the motion detection is robust. Spurious particles appear seldom, errors are mainly due to falsely adapted objects. For the simulation short sequences are used (40s), hence these errors can be reduced in a real time application by adjusting the accumulation length and adaption depth.

References

[1] Computer Vision Research Laboratory at University of Massachusetts in Amherst. Safer. `http://vis-www.cs.umass.edu/projects/safer/index.html` (29th December 2000).

[2] T. Boult, R. Micheals, X. Gao, P. Lewis, C. Power, W. Yin, and A. Erkan. Frame-rate omnidirectional surveillance and tracking of camouflaged and occluded targets. In *Second Workshop of Visual Surveillance at CVPR*, pages 48–58, 1999.

[3] T. Boult, C. Qian, W. Yin, A. Erkin, P. Lewis, C. Power, and R. Micheals. Application of omnidirectional imaging: Multi-body tracking and remote reality. In *Proceedings of the IEEE Workshop on Computer Vision Applications*, pages 242–3, October 1998.

[4] A. Bruckstein and T. Richardson. Omniview cameras with curved surface mirrors. In *IEEE Workshop on Omnidirectional Vision*, June 2000.

[5] J. Chahl and M. Srinivasan. Reflective surfaces for panoramic imaging. *Applied Optics*, 36(31):8275–85, November 1997.

[6] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *Proceedings of the International Conference on Computer Vision*, page `http://www.eecs.lehigh.edu/FRAME/Elgammal/bgmodel.html`, 1999.

[7] S. Gächter and T. Pajdla. Mirror design for an omnidirectional camera with a uniform cylindrical projection when using the SVAVISCA sensor. Technical Report CTU-CMP-2001-03, Centre for Machine Perception, Czech Technical University in Prague, 2001.

[8] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley, 1993.

[9] D. Gutchess, A. Jain, and S. Chen. Automatic surveillance using omnidirectional and active camera. In *Proceedings of the Asian Conference on Computer Vision*, 2000.

[10] I. Haritaoglu, D. Harwood, and L. Davis. Active outdoor surveillance. In *Proceedings of International Conference on Image Analysis and Processing*, pages 1096–99, 1999.

[11] T. Horprasert, D. Harwood, and L. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proceedings of the International Conference on Computer Vision*, 1999.

[12] S. Huwer and H. Niemann. Adaptive change detection for real-time surveillance applications. In *Proceedings of Third IEEE International Workshop on Visual Surveillance*, pages 37–45. IEEE Computer Society Press, 2000.

[13] P. Iliev and Tsekov L. Motion detection using image histogram sequence analysis. *Signal Processing*, 30(3):373–84, 1993.

[14] Iván Kopilović, Balázs Vágvölgyi, and Tamás Szirányi. Application of panoramic annular lens for motion analysis task: Surveillance and smoke detection. In *Proceedings of the International Conference on Pattern Recognition*, pages 714–17, 2000.

[15] A. Mannucci, P. Questa, and D. Scheffer. D1-Document on Specification, Esprit Project N. 31951 - SVAVISCA, Version 1.0. 1999.

[16] A. Neri, S. Colonnese, G. Russo, and P. Talone. Automatic moving object and background separation. *Signal Processing*, 66(2):219–32, 1998.

[17] Y. Onoe, N. Yokoya, K. Yamazawa, and H. Takemura. Visual surveillance and monitoring system using an omnidirectional video camera. In *Proceedings of the International Conference on Pattern Recognition*, pages 588–93, 1998.

[18] T. Pajdla and H. Roth. Panoramic imaging with SVAVISCA camera - simulation. Technical Report CTU-CMP-2000-16, Centre for Machine Perception, Czech Technical University in Prague, 2001.

[19] P. Rosin. Edges: Saliency measures and automatic thresholding. *Machine Vision and Application*, 9:139–59, 1997.

[20] P. Rosin. Thresholding for change detection. Technical Report ISTR-97-01, Department of Information Systems and Computing, Brunel University in Uxbridge, 1997.

[21] P. Rosin and T. Ellis. Image difference threshold strategies and shadow detection. In *Proceedings of Brittish Machine Vision Conference*, volume 1, pages 347–56. The British Machine Vision Association and Society for Pattern Recognition, 1995.

[22] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons, 1987.

[23] G. Sandini. OMNIVIEWS – Omni-directional Visual System. `http://cmp.felk.cvut.cz/` – Projects – OMNIVIEWS (22$^{\text{th}}$ February 2001).

[24] T. Svoboda. *Central Panoramic Cameras Design, Geometry, Egomotion.* PhD thesis, Center for Machine Perception, Czech Technical University in Prague, 1999.

[25] D. Večerka. SW for data translation frame grabbers. `http://cmp.felk.cvut.cz/cmp/hardware/grabbers.html` (20^{th} February 2001).

[26] R. Šára. Settings of DT3152 for OSCAR and other CCIR cameras. `http://cmp.felk.cvut.cz/cmp/hardware/oscar.html` (10^{th} January 2001).

[27] M. Šonak, Hlaváč, and R. Boyle. *Image Processing, Analysis and Machine Vision*. PWS, 1999.

A. Files

A.1. MatLab Files

A.1.1. Simulation

```
%main_simulation.m  - Author: Stefan Gachter
%
% Simulation of the motion detection algorithm.
%
%      batch file
%
%      See also: acc_still_bgd_region, bmp_to_gif, difference_image,
%               display_frame, display_frame_in_figure,
%               display_histogram, fix_format_int_str,
%               insert_frame, insert_number, label_mov_objects,
%               merge_frame, n2s, reset_mean_bgd_image, scalar_lms,
%               split_frame, weighted_accumulation

%      Author      : Stefan Gachter, stefan.gachter@ieee.org
%                    00 Center for Machine Perception,
%                    Czech Technical University, Prague
%      Documentation: Gaechter-TR-2001-07.pdf
%      Language    : Matlab  5.3.1.29215a (R11.1), (c) MathWorks
%      Last change  : 17/02/2001
%      Status      : Ready
%

clear all;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% image sequence parameters

nb_movie=14;                          % number of the image sequence
height=[31 30 18 12 20];       % heights of the subimages
%height=1;

nb_frames=200;                    % total number of images in the sequence
nb_frame_start_display=1;           % begin display at this image number
nb_frame_stop_display=nb_frames;    % end display at this image number

animation=1;                      % generate animated gif
nb_frame_start_animation=1;         % begin animation at this image number
nb_frame_stop_animation=200;        % end animation at this image number

% temporal change detection parameters

acc_length_d=0.002;                % accumulation length
adapt_depth=25;                    % adaption depth
min_sensitivity_threshold_d=5/255;  % minimal threshold
```

```
% background change detection parameters

acc_length_b=5;                    % accumulation length
min_sensitivity_threshold_b=2.5/255;  % minimal threshold

activity_threshold=5/255;           % background activity threshold

% morphological filter parameters

struct_element= ...
       [0 1 0;
        1 1 1;
        0 1 0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% path specifications

load_path=['/home.stud/qqgechte/omni/movie/' ...
         'experiment/pan/movie_' n2s(nb_movie) '/'];
load_init_path=['/home.stud/qqgechte/omni/movie/' ...
             'experiment/pan/init/movie_' n2s(nb_movie) '/'];
save_path=['/home.stud/qqgechte/omni/movie/' ...
         'animation/movie_' n2s(nb_movie) '/'];

% file name specifications

stream_name=['stream_pan_' n2s(nb_movie) ;];
animation_name=['mov_objects_movie_' n2s(nb_movie) ;];
init_mean_bgd_img_name='init_mean_bgd_img';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% misc

map_bw=[0 0 0;1 1 1];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% motion detection simulation

%% image sequence
file_name=[load_path stream_name '.mat'];
eval(['load ' file_name ' stream map']);

frame_size=[size(stream,1),size(stream,2)/nb_frames];
nb_subframes=length(height);

%% initialization
acc_frame_j=zeros(frame_size);
tmp_chg_region_hist=zeros([frame_size,adapt_depth]);
```

```
tmp_change_region=zeros(frame_size);

file_name=[load_init_path init_mean_bgd_img_name '.mat'];
eval(['load ' file_name ' init_mean_bgd_img']);
mean_bgd_img_j=init_mean_bgd_img;

bgd_change_region=zeros(frame_size);

reset=0;
frame_j=stream(:,1:frame_size(2));
for i=2:nb_frames

  display=0;
  if ((i-1)>=nb_frame_start_display)&((i-1)<nb_frame_stop_display)&~animation
    display=1;
  end

  if animation|~display
    disp(['Frame no.' fix_format_int_str(i-1,3) '.']);
  end

  %% temporal change detection
  frame_i=stream(:,(i-1)*frame_size(2)+1:i*frame_size(2));
  abs_diff_image_d=abs(difference_image(frame_i,frame_j));
  acc_frame_i=weighted_accumulation(acc_frame_j,abs_diff_image_d,acc_length_d);

  acc_subframe_i=split_frame(acc_frame_i,height);
  abs_diff_subimage_d=split_frame(abs_diff_image_d,height);

  for k=1:nb_subframes;
    eval(['lms_abs_diff_subimage_d=' ...
        'scalar_lms(abs_diff_subimage_d.no' n2s(k) '(:));']);
    eval(['sensitivity_threshold_d.no' n2s(k) '=' ...
        '2.576*lms_abs_diff_subimage_d/0.3372+min_sensitivity_threshold_d;']);
    eval(['tmp_change_subregion.no' n2s(k) '=' ...
        'acc_subframe_i.no' n2s(k) '>sensitivity_threshold_d.no' n2s(k) ';']);
  end

  [tmp_change_region,border_coordinates_tmp]= ...
    merge_frame(tmp_change_subregion,nb_subframes,'border');

  tmp_change_region=erode(tmp_change_region,struct_element);
  tmp_change_region=dilate(tmp_change_region,struct_element);

  bgd_activity=scalar_lms(abs_diff_image_d);
  if bgd_activity<activity_threshold
    reset=0;
  end
  if (bgd_activity>=activity_threshold)&(~reset)
    reset=1;
    disp(['Reset at frame no.' fix_format_int_str(i-1,3) ...
         ' when level ' n2s(bgd_activity) '.']);
```

```
  mean_bgd_img_j= ...
    reset_mean_bgd_image(mean_bgd_img_j,frame_j,frame_i,9,bgd_change_region);
  tmp_change_region=zeros(frame_size);
end

[still_bgd_region_mask,tmp_chg_region_hist]= ...
    acc_still_bgd_region(tmp_change_region,tmp_chg_region_hist);

%% background change detection
mean_bgd_img_i= ...
  weighted_accumulation(mean_bgd_img_j,frame_i,acc_length_b,still_bgd_region_mask);
abs_diff_image_b=abs(difference_image(mean_bgd_img_i,frame_i));

abs_diff_subimage_b=split_frame(abs_diff_image_b,height);
for k=1:nb_subframes;
  eval(['lms_abs_diff_subimage_b=' ...
    'scalar_lms(abs_diff_subimage_b.no' n2s(k) '(:));']);
  eval(['sensitivity_threshold_b.no' n2s(k) '=' ...
    '2.576*lms_abs_diff_subimage_b/0.3372+min_sensitivity_threshold_b;']);
  eval(['bgd_change_subregion.no' n2s(k) '=' ...
    'abs_diff_subimage_b.no' n2s(k) '>sensitivity_threshold_b.no' n2s(k) ';']);
end

[bgd_change_region,border_coordinates_bgd]= ...
  merge_frame(bgd_change_subregion,nb_subframes,'border');

bgd_change_region=erode(bgd_change_region,struct_element);
bgd_change_region=dilate(bgd_change_region,struct_element);

%% labeling
frame_coordinates=label_mov_objects(bgd_change_region,'frame');

%%% display
fig_frame_i=insert_number(frame_i,i-1,3,[2,2],'inverse');
fig_abs_diff_image_d=insert_number(abs_diff_image_d,i-1,3,[2,2],'inverse');
fig_acc_frame_i=insert_number(acc_frame_i,i-1,3,[2,2],'inverse');
[fig_tmp_change_region,map_tmp_change]= ...
  insert_frame(uint8(tmp_change_region),border_coordinates_tmp,'red',map_bw);
fig_tmp_change_region= ...
  insert_number(fig_tmp_change_region,i-1,3,[2,2],'transparent');
fig_still_bgd_region_mask= ...
  insert_number(still_bgd_region_mask,i-1,3,[2,2],'transparent');
fig_mean_bgd_img_i=insert_number(mean_bgd_img_i,i-1,3,[2,2],'inverse');
fig_abs_diff_image_b=insert_number(abs_diff_image_b,i-1,3,[2,2],'inverse');
[fig_bgd_change_region,map_bgd_change]= ...
  insert_frame(uint8(bgd_change_region),border_coordinates_bgd,'red',map_bw);
fig_bgd_change_region= ...
  insert_number(fig_bgd_change_region,i-1,3,[2,2],'transparent');
[fig_labeled_change_region,map_change_region]= ...
  insert_frame(bgd_change_region,frame_coordinates,'red',map_bw);
fig_labeled_change_region= ...
  insert_number(fig_labeled_change_region,i-1,3,[2,2],'transparent');
```

```
[fig_labelled_frame_i,map_frame]= ...
  insert_frame(frame_i,frame_coordinates,'green',map);
fig_labelled_frame_i= ...
  insert_number(fig_labelled_frame_i,i-1,3,[2,2],'inverse');

if display

%display_frame(uint8(255*fig_frame_i),map,1);
%display_frame(uint8(255*fig_abs_diff_image_d),map,2);
%display_frame(uint8(255*fig_acc_frame_i),map,3);
%display_frame(uint8(fig_tmp_change_region),map_tmp_change,4);
%display_frame(uint8(fig_still_bgd_region_mask),map_bw,5);
%display_frame(uint8(255*fig_mean_bgd_img_i),map,6);
%display_frame(uint8(255*fig_abs_diff_image_b),map,7);
%display_frame(uint8(fig_bgd_change_region),map_bgd_change,8);
%display_frame(uint8(fig_labeled_change_region),map_change_region,9);
%display_frame(uint16(255*fig_labelled_frame_i),map_frame,10);

frames=sequence_to_struct(uint8(255*fig_frame_i), ...
            uint8(255*fig_abs_diff_image_d), ...
            uint8(fig_tmp_change_region), ...
            uint8(fig_still_bgd_region_mask), ...
            uint8(255*fig_mean_bgd_img_i), ...
            uint8(255*fig_abs_diff_image_b), ...
            uint8(fig_bgd_change_region), ...
            uint16(255*fig_labelled_frame_i));

map_fig= ...
  sequence_to_struct(map,map,map_tmp_change,map_bw, ...
                map,map,map_bgd_change,map_frame);

display_frame_in_figure(frames,map_fig,8,[4 2],11);

display_histogram(abs_diff_subimage_d,nb_subframes, ...
  sensitivity_threshold_d,[0 0.05],12);
xlabel('absolute difference for two consecutive images');
display_histogram(abs_diff_subimage_b,nb_subframes, ...
  sensitivity_threshold_b,[0 0.05],13);
xlabel('absolute difference for background subtraction');

pause;
end

%%% animation
if ((i-1)>=nb_frame_start_animation)&((i-1)<nb_frame_stop_animation)&animation

%red_map_frame=[gray(255);name_to_rgb('green')];
%frame=imapprox(uint16(255*fig_labelled_frame_i),map_frame,red_map_frame);

frames=sequence_to_struct(uint8(255*fig_frame_i), ...
            uint8(255*fig_abs_diff_image_d), ...
            uint8(fig_tmp_change_region), ...
```

```
255             uint8(fig_still_bgd_region_mask), ...
                uint8(255*fig_mean_bgd_img_i), ...
                uint8(255*fig_abs_diff_image_b), ...
                uint8(fig_bgd_change_region), ...
                uint16(255*fig_labelled_frame_i));
260
  map_fig=sequence_to_struct(map,map,map_tmp_change,map_bw, ...
                         map,map,map_bgd_change,map_frame);

  fig_rgb=display_frame_in_figure(frames,map_fig,8,[4 2],'not_display');
265  red_map_frame=[gray(254);name_to_rgb('red');name_to_rgb('green')];
  frame=rgb2ind(fig_rgb,red_map_frame);

  file_name=[save_path animation_name '_' fix_format_int_str(i-1,3) '.bmp'];
  imwrite(frame,red_map_frame,file_name,'bmp');
270  end

  %% update
  frame_j=frame_i;
  acc_frame_j=acc_frame_i;
275  mean_bgd_img_j=mean_bgd_img_i;

end

%%% animation
280 if animation
  start_stop=[nb_frame_start_animation nb_frame_stop_animation-1];
  bmp_to_gif(save_path,animation_name,start_stop, ...
    'animate_whirlgif','erase_bmp','erase_gif');
end
```

Temporal Change History

```
function [still_bgd_region_mask,tmp_chg_region_hist]= ...
   acc_still_bgd_region(tmp_change_region,tmp_chg_region_hist);

% [still_bgd_region_mask,tmp_chg_region_hist]= ...
5 %   accumulate_still_background_region(tmp_change_region,tmp_chg_region_hist);
%
% Creates region that did not change for a period of time.
%
% stefan.gachter@ieee.org
10
hist_size=size(tmp_chg_region_hist);
adapt_depth=hist_size(3);

tmp_chg_region_hist(:,:,1:adapt_depth-1)=tmp_chg_region_hist(:,:,2:adapt_depth);
15 tmp_chg_region_hist(:,:,adapt_depth)=tmp_change_region;

still_bgd_region_mask=sum(tmp_chg_region_hist,3)<1;
```

Difference Images

```
function diff_img=difference_image(frame_i,frame_j);

% diff_img=difference_image(frame_i,frame_j)
%
% Computes the difference image between the frame i
% and the frame j.
%
% stefan.gachter@ieee.org

diff_img=frame_i-frame_j;
```

Label Moving Objects

```
function varargout=label_mov_objects(change_region,varargin);

% [{labeled_change_region,frame_coordinates}]= . . .
%                       label_mov_objects(change_region,{'frame'})
%
% Label objects in the change region and generates the coordinates of the
% frames around each object.
%
% stefan.gachter@ieee.org


frame=0;
for k=2:nargin
  if isstr(varargin{k-1})
    frame=strcmp(varargin{k-1},'frame');
  end
end

[labeled_change_region,nb_objects]=bwlabel(change_region);

if (nargout==1)&~frame
  varargout{1}=labeled_change_region;
  return
end

if frame
  frame_coordinates=[];
  frame_size=size(change_region);
  for i=1:nb_objects;
    [coord_m,coord_n]=find(labeled_change_region==i);
    max_m=max(coord_m);
    min_m=min(coord_m);
    max_n=max(coord_n);
    min_n=min(coord_n);
    left=min_n;
    bottom=frame_size(1)-max_m+1;
```

```
    width=max_n-min_n+1;
    height=max_m-min_m+1;
    frame_coordinates=[frame_coordinates;[left bottom width height]];
  end
  if nargout==1
    varargout{1}=frame_coordinates;
  return
  end
  if nargout==2
    varargout{1}=labeled_change_region;
    varargout{2}=frame_coordinates;
  return
  end
end
```

Merge Frame

```
function [frame,varargout]=merge_frame(varargin);

% [frame,{border_coordinates}]=merge_frame(subframe_1,...,subframe_n,{'border'})
% [frame,{border_coordinates}]=merge_frame(subframe.no1...n,n,{'border'})
%
% Merges n subframes vertically in one frame and generates
% optionally the border coordinates around each subframe.
%
% stefan.gachter@ieee.org

border=0;
if strcmp(varargin{nargin},'border')
  border=1;
end

if ~isstruct(varargin{1})
  n=nargin;
  if border
    n=nargin-1;
  end
  frame=[];
  subframe_size=[];
  for i=1:n
    eval(['subframe_tmp=varargin{' num2str(i) '};']);
    subframe_size=[subframe_size;size(subframe_tmp)];
    frame=[frame;subframe_tmp];
  end
end

if isstruct(varargin{1})
  n=varargin{2};
  subframe=varargin{1};
  frame=[];
  subframe_size=[];
```

```
  for i=1:n
    eval(['subframe_tmp=subframe.no' num2str(i) ';']);
    subframe_size=[subframe_size;size(subframe_tmp)];
    frame=[frame;subframe_tmp];
  end
end

if border
  frame_size=size(frame);
  border_coordinates=[];

  left=1;
  bottom=1;
  width=subframe_size(n,2);
  height=subframe_size(n,1);
  border_coordinates=[border_coordinates;[left bottom width height]];

  for i=n-1:-1:1
    left=1;
    bottom=bottom+height-1;
    width=subframe_size(i,2);
    height=subframe_size(i,1)+1;
    border_coordinates=[border_coordinates;[left bottom width height]];
  end
  varargout{1}=border_coordinates;
end
```

Reset Background Model

```
function mean_bgd_img_j= ...
         reset_mean_bgd_image(mean_bgd_img_i,frame_j,frame_i,digits,varargin);

% mean_bgd_img_j= ...
%  reset_mean_bgd_image(mean_bgd_img_i,frame_j,frame_i,digits, ...
%                   {mask_exclude_bgd_region});
%
% Reset mean background region.
%
% stefan.gachter@ieee.org

struct_element= ...
              [0 1 0;
               1 1 1;
               0 1 0];

frame_j(find(frame_j<10^(-digits)))=1;
ratio=frame_i./frame_j;

if nargin==4;
  mean_bgd_img_j=mean_bgd_img_i.*ratio;
  return
```

```
end

if nargin==5;
  mask=varargin{1};
  mask=dilate(mask,struct_element);
  smoothed_ratio=roifill(ratio,mask);
  mean_bgd_img_j=mean_bgd_img_i.*smoothed_ratio;
  return
end
```

Least Median of Squares for Scalar Values

```
function lms=scalar_lms(scalar_set);

% function lms=scalar_lms(scalar_set)
%
% Computes the Least Median of Squares for a set of scalars.
%
% see Rousseeuw, P., Leroy, A., Robust Regression and Outlier Detection,
%    Wiley, 1987
%
% stefan.gachter@ieee.org

scalar_vect=scalar_set(:);
sort_scalar_vect=sort(scalar_vect);

n=length(scalar_vect);
h=fix(n/2)+1;
g=ceil(n/2);
diff=sort_scalar_vect(h:n)-sort_scalar_vect(1:g);

[dummy,min_diff_index]=min(diff);
lms=0.5*(sort_scalar_vect(min_diff_index)+sort_scalar_vect(min_diff_index+h-1));
```

Split Frame

```
function [varargout]=split_frame(frame,varargin);

% [subframe_1,...,subframe_n]=split_frame(frame)
% [subframe_1,...,subframe_n]=split_frame(frame,[height1,...,heightn])
% [subframe.no1...n]=split_frame(frame,n)
% [subframe.no1...n]=split_frame(frame,[height1,...,heightn])
%
% Split the frame vertically in n subframes.
%
% stefan.gachter@ieee.org


if nargout>1
  if nargin==1
```

```
    n=nargout;
    frame_size=size(frame);
    step=fix(frame_size(1)/n);
    for i=1:n-1
      eval(['subframe_' num2str(i) ...
           '=frame((' num2str(i) '-1)*step+1:' num2str(i) '*step,:);']);
      eval(['varargout{' num2str(i) '}=subframe_' num2str(i) ';']);
    end
    eval(['subframe_' num2str(n) ...
         '=frame((' num2str(n) '-1)*step+1:frame_size(1),:);']);
    eval(['varargout{' num2str(n) '}=subframe_' num2str(n) ';']);
    return
  else
    height=varargin{1};
    n=length(height);
    frame_size=size(frame);
    ind_begin=1;
    ind_end=0;
    for i=1:n
      ind_begin=ind_end+1;
      ind_end=ind_begin+height(i)-1;
      eval(['subframe_' num2str(i) '=frame(ind_begin:ind_end,:);']);
      eval(['varargout{' num2str(i) '}=subframe_' num2str(i) ';']);
    end
    return
  end
end

if (nargout==1)&(nargin==1)
  n=nargout;
  varargout{n}=frame;
  return
end

if (nargout==1)&(nargin==2)
  if length(varargin{1})==1
    n=varargin{1};
    frame_size=size(frame);
    step=fix(frame_size(1)/n);
    for i=1:n-1
      eval(['subframe.no' num2str(i) ...
           '=frame((' num2str(i) '-1)*step+1:' num2str(i) '*step,:);']);
    end
    eval(['subframe.no' num2str(n) ...
         '=frame((' num2str(n) '-1)*step+1:frame_size(1),:);']);
    eval(['varargout{1}=subframe;']);
    return
  else
    height=varargin{1};
    n=length(height);
    frame_size=size(frame);
    ind_begin=1;
```

```
    ind_end=0;
    for i=1:n
      ind_begin=ind_end+1;
      ind_end=ind_begin+height(i)-1;
      if ind_end>frame_size(1)
        ind_end=frame_size(1);
      end
      eval(['subframe.no' num2str(i) '=frame(ind_begin:ind_end,:);']);
    end
    eval(['varargout{1}=subframe;']);
    return
  end
end
```

Weighted Accumulation

```
function acc_frame_i= ...
    weighted_accumulation(acc_frame_j,frame_j,acc_length,varargin);

% acc_frame_i=weighted_accumulation(acc_frame_j,frame_j,acc_length,region)
%
% Computes the weighted accumulation of a frame stream restricted to the
% defined region. The length of accumulation is the number of past values
% to be considered.
%
% stefan.gachter@ieee.org

mask=ones(size(frame_j));
if nargin==4
  mask=varargin{1};
end

a=exp(-1/acc_length);

region_acc_frame_i=((1-a)*frame_j+a*acc_frame_j).*mask;
non_region_acc_frame_i=acc_frame_j.*~mask;

acc_frame_i=region_acc_frame_i+non_region_acc_frame_i;
```

A.2. File Index

A.2.1. MatLab Files

The file *name* is listed in the first column, the file *path* in the second, and the reference of the corresponding *section* in the third.

acc_still_bgd_region	/home.stud/qqgechte/omni/matlab/	1,2
bmp_to_gif	/home.stud/qqgechte/matlab/tools/image/	2
circle	/home.stud/qqgechte/matlab/tools/graphics/	1,2
create_init_images	/home.stud/qqgechte/omni/matlab/simulation/	1,2
create_noise_frames	/home.stud/qqgechte/omni/matlab/simulation/	1
create_stream	/home.stud/qqgechte/omni/matlab/simulation/	1,2
define_center	/home.stud/qqgechte/omni/matlab/simulation/	1,2
difference_image	/home.stud/qqgechte/omni/matlab/	1,2
display_frame	/home.stud/qqgechte/matlab/tools/image/	1,2
display_frames	/home.stud/qqgechte/omni/matlab/simulation/	1,2
display_frame_in_figure	/home.stud/qqgechte/matlab/tools/image/	2
display_histogram	/home.stud/qqgechte/matlab/tools/graphics/	2
fix_format_int_str	/home.stud/qqgechte/matlab/tools/general/	1,2
fixed_point_conv	/home.stud/qqgechte/omni/matlab/tools/general/	2
insert_frame	/home.stud/qqgechte/matlab/tools/image/	1,2
insert_number	/home.stud/qqgechte/matlab/tools/image/	1,2
isdouble	/home.stud/qqgechte/matlab/tools/general/	1,2
isuint8	/home.stud/qqgechte/matlab/tools/general/	1,2
label_mov_objects	/home.stud/qqgechte/omni/matlab/	2
main_simulation	/home.stud/qqgechte/omni/matlab/simulation/	2
merge_frame	/home.stud/qqgechte/matlab/tools/image/	1,2
n2s	/home.stud/qqgechte/matlab/tools/general/	2
name_to_rgb	/home.stud/qqgechte/matlab/tools/image/	1,2
plot_hist	/home.stud/qqgechte/matlab/tools/graphics/	1
quantify_noise	/home.stud/qqgechte/omni/matlab/simulation/	1
record_frames	/home.stud/qqgechte/omni/matlab/simulation/	1,2
reformat_image	/home.stud/qqgechte/omni/matlab/simulation/	1,2
reset_mean_bgd_image	/home.stud/qqgechte/omni/matlab/	2
scalar_lms	/home.stud/qqgechte/matlab/tools/statistic/	1,2
size_filter	/home.stud/qqgechte/matlab/tools/image/	1
split_frame	/home.stud/qqgechte/matlab/tools/image/	1,2
svsim	/home.stud/qqgechte/omni/matlab/simulation/	1,2
weighted_accumulation	/home.stud/qqgechte/omni/matlab/	2

A.2.2. Miscellaneous

The file *name* is listed in the first column, the file *path* in the second, and the reference of the corresponding *section* in the third.

whirlgif	`/home.stud/qqgechte/matlab/tools/image/whirlgif/`	2