REAL-TIME DETECTION OF ARBITRARY OBJECTS IN ALTERNATING INDUSTRIAL ENVIRONMENTS

Dirk Balthasar, Thomas Erdmann, Johannes Pellenz, Volker Rehrmann, Jörg Zeppen, Lutz Priese

Image Recognition Laboratory
University of Koblenz-Landau
Rheinau 1, D 56075 Koblenz, Germany
lb@uni-koblenz.de

ABSTRACT

Consider the following problem: How to detect objects of undetermined and unequal shape, size and color in front of an alternating but repeating background in real-time? In addition, the objects may slightly overlap each other but have to be considered as separate objects. Also, the repeating background may change in time because of variations of the illumination or staining of the environment. This problem arises from industrial applications and requires a very robust and fast solution. Some hundreds of objects have to be detected and analyzed for overlapping within a second. The presented solution has already been used successfully in several automatic sorting techniques for recyclable materials, where the objects to be sorted are placed on a very fast conveyor belt and a perfect distribution of the objects cannot be guaranteed. The solution is based on the following principles:

- dynamic update of a reference image of the background to adapt to variations of the environment.
- intelligent difference image generation using reference background and actual image, to distinguish between objects and changes within the background.
- new fast cutting technique on chain codes to split overlapping objects.

1. INTRODUCTION

Detecting objects on a transportation unit is a standard problem for industrial applications. If all objects are of the same shape or of a shape out of a few types this problem has been frequently solved by model-based recognition methods. However, there are many examples in practical applications where these objects may largely vary. These cases are usually not handled by computer vision methods. For example, in recycling plants paper, bottles, plastics and other materials have to be sorted. These objects occur in a wide range of shape, size and color. They are generally transported on conveyor belts. To inspect the objects a stationary color CCD camera mounted above the conveyor belt is used to acquire images of the scene. Fig. 1 shows the simplified diagram of an automated visual sorting system.

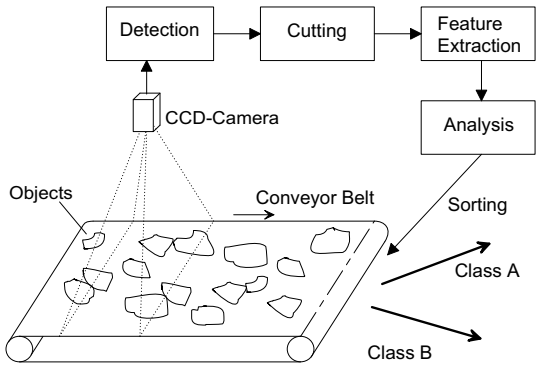

Fig. 1. Diagram of an automated visual sorting system

To apply computer vision methods the objects must cover the belt in a single layer. On the other hand, a reasonable throughput has to be achieved. By using high-speed transportation belts both can be reached, reasonable throughput and avoiding that objects are covered by others. Nevertheless, overlapping of objects on the transportation unit cannot be prevented in practical applications. For the sorting process it is necessary to treat these overlapping objects as several single objects to perform further analysis. Thus, two or more adjoined objects must be detected as different objects. Standard techniques – using morphological operations [2], [8] – are not suitable for this situation as explained in section 3. We present a rather general method that we use successfully in several applications.

Another problem is to decide whether or not a pixel belongs to an object or to the background. Several approaches are known to separate the foreground from the background [6]. These methods work reliably in well defined environments. For example, if the background has a high contrast against the objects simple thresholding can be used. At a visual sorting system – such as the one shown in Fig. 1 – objects of undetermined shape, size and color are transported

on a monochrome conveyor belt. The objects may soil the belt and dirt or abrasion lead to stains of unpredictable shape and color. These stains should be regarded as a part of the background and not as an object to be classified. Because the stains could have similar shape, size and color in comparison with the objects, region- or edge-based approaches for the segmentation will fail.

Our approach is based on background subtraction – a difference image technique – which is a common way to detect moving objects in front of a static background [5], [7]. We adapt this approach to detect objects on a moving background by using the repetitive occurrence of the same conveyor belt area with each revolution of the belt. We further use dynamic background adaptation to cope with illumination changes and stains on the belt, as shown in the next section.

In section 3 we describe a new technique to separate the overlapping objects. This technique is based on chain codes. In section 4 and 5 an overview of all needed steps of the system is given. The design of the system enables a high scalability by using pipelining and parallel processing. Real-time performance can be obtained on standard PCs.

2. OBJECT DETECTION

To distinguish between static background and moving objects with a stationary camera the difference image method has been used for years [9]. The objects are detected by subtracting the pixel values of different frames and using thresholding. Let $f(x, y, t) = (r(x, y, t), g(x, y, t), b(x, y, t))$ denote the RGB-value at the image position (x, y) of the image $F(t)$ at the time t.

$$f_d(x, y, t_1, t_2) = f(x, y, t_1) - f(x, y, t_2) \qquad (1)$$

is a pixel of the difference image $F_d(t_1, t_2)$ between t_1 and t_2. If $F(t_1)$ captures a scene with no objects in front of the background it can be used as a reference image F_{ref} of the background. The idea of the background subtraction method is to subtract the current image from this reference image.

2.1. Difference Image Generation

The difference image method is usually used in applications with a fixed background and moving objects. It can also be applied to detect objects which are placed on a moving but repeating background such as a conveyor belt. After each revolution of the belt the same background – with other objects placed on it – appears again. The current image has to be compared with the reference image of the identical belt position. For this reason, images of the entire belt have to be kept in memory. To ensure that the current image matches the stored reference image perfectly, the CCD camera takes images always at exactly the same belt positions. This is achieved by using an asynchronous reset camera and triggering it by marks that are attached on the belt. The distance between the marks is chosen in such a way that the images cover the whole surface of the belt. The initial set of reference images is built by taking images of the empty belt.

Integrating a thresholding operation in the difference image method results in the distinction between fore- and background for each pixel in the image. The thresholding is necessary since the camera noise in the images always produces some response in the difference images – even if the scene didn't change at all.

Two simple methods can be used for thresholding:

1. For each pixel the three differences of the RGB values are added and compared with one single threshold to determine if the pixel is a foreground or a background pixel. The result is stored in a binary image B consisting of pixels

$$b(x, y) = \begin{cases} 1 : \text{if} & |r_{ref} - r_{cur}| + \\ & |g_{ref} - g_{cur}| + \\ & |b_{ref} - b_{cur}| > \Omega \\ 0 : \text{else} & \end{cases} \qquad (2)$$

2. Three individual thresholds $(\Omega_r, \Omega_g, \Omega_b)$ for each color are used, because the noise distribution for each color channel of a RGB camera can differ [4].

$$b(x, y) = \begin{cases} 1 : \text{if} & |r_{ref} - r_{cur}| > \Omega_r \vee \\ & |g_{ref} - g_{cur}| > \Omega_g \vee \\ & |b_{ref} - b_{cur}| > \Omega_b \\ 0 : \text{else} & \end{cases} \qquad (3)$$

In our approach method 2 is used since it also detects objects that differ in only one color component from the background. The noise of the camera is determined by analyzing a static scene for a number of frames. By defining an error ratio r which determines how many pixels are accepted to be classified falsely as foreground pixels the thresholds can be calculated.

2.2. Dynamic Background Adaptation

The simple difference image approach cannot handle the problem of variation in the illumination or of new stains appearing in the background. Therefore a dynamic background update may be used as described in [1]. However, this approach fails if the same belt position is covered by a new object at each revolution. This happens frequently in practical environments where objects are transported on a belt. Therefore, we extended the algorithm described in [1] to handle these problems without losing precision in the results of the background adaptation.

The idea is that for each pixel in the reference image F_{ref} two additional pieces of information are stored:

- The color that was seen at this pixel position in the recent images taken from this belt position is contained in the update image F_u.
- The number of times this color has successively been seen before at this place is saved in a counter array C.

If the same color was observed for a number of times, this color becomes the new reference color at this position.

The exact algorithm is given as follows: First, the counter values $c(x, y)$ are initialized to zero for all (x, y). The binarization $b(x, y)$ of the current pixel $f(x, y)$ is determined by using the difference image method described above. The update of the reference image pixels $f_{ref}(x, y)$ is calculated by the following steps:

- If $b(x, y) = 0$ (the pixel was identified as background), then update the reference image by using the weighted average

$$f_{newref}(x, y) = (1-\omega) \cdot f_{ref}(x, y) + \omega \cdot f(x, y) \quad (4)$$

  with $\omega \in [0 \ldots 1]$. The value of ω determines the weighting of the current pixel compared to the previous ones. The counter value $c(x, y)$ is set to zero. The update of the reference image enables the dynamic adaptation to slightly changes of the background by variation of the lighting conditions or by abrasion of the belt.

- If $b(x, y) = 1$ (the pixel was identified as foreground), there are two cases to distinguish between:
  - If $c(x, y) = 0$, then increment $c(x, y)$ and set $f_u(x, y)$ to $f(x, y)$. This is a new foreground pixel which occurs the first time.
  - If $c(x, y) > 0$, then $f(x, y)$ have to be compared with $f_u(x, y)$.

    If the differences – calculated similar to Eq. 3 – are more than the difference thresholds, then another foreground than in the last frame is detected. Thus, the counter value $c(x, y)$ is set to one and $f_u(x, y)$ to $f(x, y)$.

    If the differences are less than the difference thresholds, then the same foreground pixel occurs as in the last frame. The counter value $c(x, y)$ is incremented and the pixel of the update image has to be adjusted by the weighted average

$$f_u(x, y) = (1-\omega) \cdot f_u(x, y) + \omega \cdot f(x, y) \quad (5)$$

    with $\omega \in [0 \ldots 1]$. If $c(x, y)$ is greater than a threshold Φ, $f_{ref}(x, y)$ is set to $f_u(x, y)$ and $c(x, y)$ is set to zero. In this case the same foreground pixel has occurred Φ-times unchanged. Some event must have happened, that changed the color of the conveyor belt at this position and thus the value of the reference pixel. Perhaps the light conditions changed significantly or an object sticks on the belt.

This method can handle both slight and rapid changes of background. A threshold Φ of 3 is used successfully in several applications. This means, if similar values appear successively three times at the same position on the belt and they are different to the current reference image, the reference image is updated.

2.3. Segmentation

From the binary image which represents the foreground and the background pixels the connected regions have to be extracted. This is done by scanning thru the image and creating a suitable object representation of the found regions for further processing. Fig. 2 shows two consecutive binary

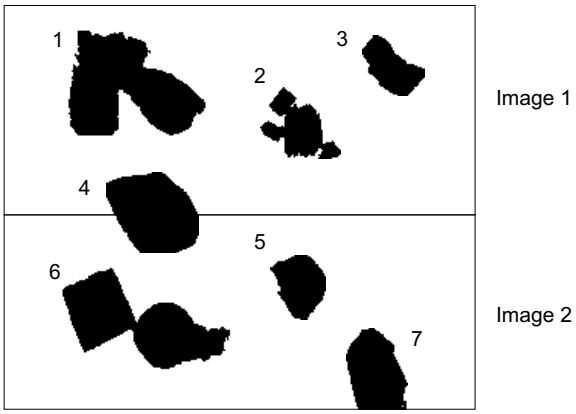

Fig. 2. Two consecutive binary images

images. Four connected regions can be detected in the first image. The regions 1 and 2 are composed of several physical objects and have to be split up. Region 4 consists of one single object but is located in image 1 and image 2. If image 1 is processed first, the two pieces of the object have to be merged at the borders of the images. The object representation should have the following properties to meet our requirements:

- The calculation of the object representation should be fast and reliable.
- It should be a very compact representation, so that the further operation can be computed efficiently.

For these reasons the chain code data structure is chosen which represents the boundary of a binary image in an efficient way. Details on chain codes can be found, e.g., in [3] and [9]. The chain code enables to compute many shape features which are needed for the further operations – the cutting and the merging of regions – in a faster way than in the matrix representation of the binary image.

3. CUTTING OF OVERLAPPING OBJECTS

The objects transported on the belt are distributed by mechanical techniques. A method has to be found to detect overlapping objects on the belt and to represent and treat them as separate objects by the computer vision system. For the further processing, each part of a connected region has to be considered as a single object because these objects might belong to different classes. Thus, each physical object has to be treated on its own. This can not be done by segmentation using the color or texture information, since the objects have undetermined colors which can even vary within an object. The only way to detect groups of overlapping objects is to analyze the shape of the image.

3.1. Existing approaches

A common approach for the cutting of connected regions in binary images is based on morphological operations. First an erosion is executed several times on the binary image. Slightly overlapping objects lead to regions which are shown in Fig. 3. After three iterations of the erosion with a 3×3 cross mask as the structure element some of the regions are divided into separate regions of smaller size than the original objects. Therefore, dilation grows the regions back toward the original size of the objects. To prevent merging of the just separated regions some logic has to be used. In addition, an AND operation between the dilated image and the original image has to be performed to restore the original region boundaries. [8] describes in detail which steps

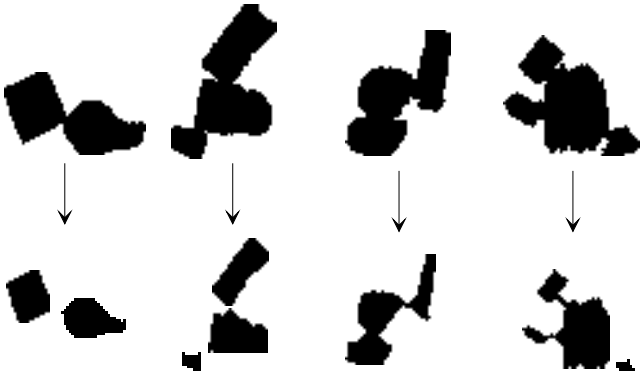

Fig. 3. Binary images of objects and the result after three cycles of erosion

are necessary to get the original object shape. It is obvious that the simple approach of erosion and dilation is not suitable if the original size and shape of the split objects is needed for further analysis.

Fig. 3 also shows that not all regions are cut by the mentioned three cycles of the erosion. To decompose all regions from each other more iterations would be necessary. This is only possible if all objects habe almost the same size. By increasing the number of erosion cycles for the objects shown in Fig. 3 some small objects will disappear and cannot be recovered. An approach to deal with this problem is given by the watershed segmentation technique which is also based on morphological operations [8]. Unfortunately, this approach is not suitable for us because of our real-time requirements on standard PCs. Other approaches (e.g. [2]) need strict preconditions regarding the shape and the size of the objects. Since objects may have unpredictable shape and size in industrial applications these preconditions cannot be guaranteed in many practical applications.

3.2. Cutting by boundary information

Our approach uses only boundary information which is coded in the chain code representation. Fig. 4 shows the binary images of some overlapping objects. Regions representing a group of overlapping objects are characterized by constrictions. The goal of the algorithm is to find the line at which a region should be cut. Let us call two points on the boundary of a shape where a cut should take place a pair of cut-points. We now characterize a pair of cut-points as follows:

- The direct line between a pair of cut points is rather short,
- but the (shortest) distance between this pair of cut-points following the boundary of the shape is rather large.

By using both distances – the length of the boundary line and the direct line – a ratio can be calculated:

$$r = \frac{\text{length of the boundary line}}{\text{length of the direct line}}$$

This ratio is used to decide whether the direct line between two arbitrary points on the boundary of a region could be a constriction. By examining this ratio for all possible pairs of cut-points the best candidates for a pair of cut-points are found. For these pairs the ratio is the maximum.

The algorithm has some favorable characteristics:

- All parameters can be calculated by the using the chain code representation without using the matrix representation of the binary image.
- The calculation can be accelerated without losing precision.
- The cutting can be adjusted by different parameters:
  - By defining a threshold r_{min} for the ratio r that determines how significant the constriction must be.
  - By defining a maximal length for the distance between the cut-points.
  - By defining a curvature value to determine how strong the curvature has to be at the boundary points.

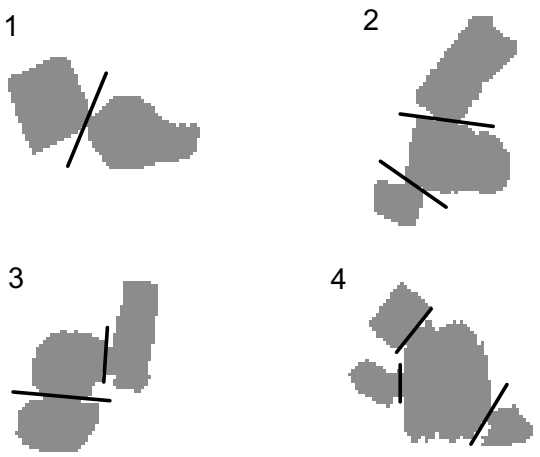

Fig. 4. Binary images of overlapping objects with the desired cutting line

To decide whether the algorithm has to be applied at all on a certain region, the compactness of the region is analyzed. Due to their high compactness the objects of Fig. 5 are not subjects of this cutting algorithm at all.

Fig. 5. Binary images of single objects

In Table 1 the compactness for the regions 1 to 8 is listed. The compactness is defined as

$$c = \frac{4 \cdot \pi \cdot area}{(perimeter)^2}$$

which can be computed efficiently by using the chain code representation. Regions with a high compactness are not treated by the cutting algorithm. By experiments we have determined that only regions with a compactness lower than 0.6 have to be cut.

Table 1. Compactness of the regions in Fig. 4 and 5

Region	Compactness	Region	Compactness
1	0.366	5	0.672
2	0.267	6	0.739
3	0.310	7	0.799
4	0.232	8	0.706

After such a cutting one or both of the resulting regions may still have a low compactness. Therefore, the cutting algorithm has to be applied recursively on the resulting regions until their compactness is large enough. Fig. 6 shows an example of a recursive cutting of regions of overlapping objects.

Fig. 6. Recursive cutting of regions

Unfortunately, the algorithm proposed in the last section runs in time O(n^2), where n is the length of the chain code. However, some optimizations can be applied on the algorithm to reduce the execution time dramatically without loosing precision in the results:

- *Selecting curve points*: By analyzing the shape of the object groups depicted in Fig. 4 it is obvious that the start- and the endpoints of the constriction are always located in a left curve (following the chain code clockwise). The strength of the curve can easily be computed for each point by comparing its (X, Y)-position with the (X, Y)-position of the preceding and subsequent chain code coordinates. For the following steps of the algorithm, only these "left curve points" are considered which are only a small fraction of all points. Fig. 7 depicts the selected points (in black) on a typical group of two objects.
- *Limiting the length of the constriction*: If the maximum length of the constriction is limited to c pixels, a various number of points on the chain code may often be skipped. Suppose two points A and B are tested whether they form a pair of cut-points. Now, if their pixel-distance is $d > c$ than they are not admissible for a distinction of length $\leq c$. But also the next $d - c$ points B' following B on the chain code cannot form an admissible pair of cut-points together with A and may be skipped from consideration. This optimization yields enormous savings in computation time, especially for large regions.

4. MERGING OF REGIONS

In section 2.3 it has been shown that a part of an object may be at the bottom of an image and another part at the top of the next image. The images are processed immediately after they are captured by the camera to meet real-time requirements. Thus, regions at the borders of an image have to be merged, if they belong to one object that is only cut

Fig. 7. Selected candidates for the start- and endpoint of the constriction

by the image border. Further, as objects may change their relative position towards the background, the same object at time $t+1$ may be at a different position than at time t. A simple merging of those moving parts may thus lead to effects as shown in Fig. 8. It shows a border between two successive images with three objects which are cut by the image border.

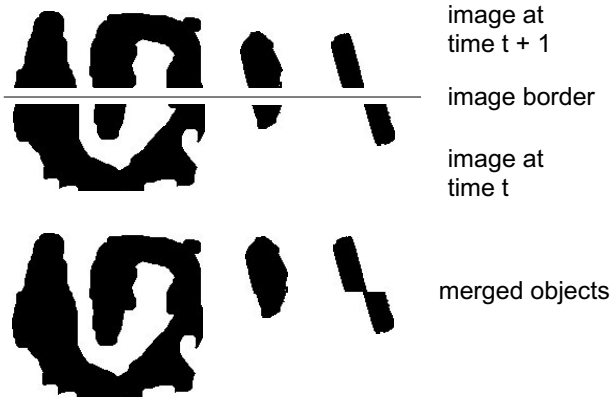

Fig. 8. Images of border objects which are merged together

In section 3 we have presented a cutting algorithm that operates solely on the chain code representation. In real-time applications one should avoid several representations of the same object, e. g. by chain code and matrix representation. However, some rather obvious considerations lead also to a fast merging algorithm based on the chain code representation of objects. A list of points on the chain code is stored that describes positions where the chain code touches the border - the so called border-point-list. But the representation of the right most object in Fig. 8 is rather unsatisfactory, as the change in position from time t to time $t+1$ should be corrected. Therefore, it is sufficient to compare objects at the bottom border of image t with objects at the top border of image $t+1$. If two objects relatively close in x-axis have a cut-line – defined by the image border – of equal length, they should be merged as shown in Fig. 9. This is easily done using solely the chain codes and the border-point-list.

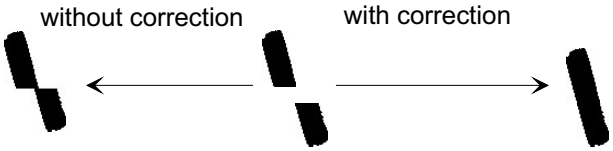

Fig. 9. The correction of the displacement caused by a moving object

5. PROCESS PIPELINE

The detection of the objects in the approach described in this paper is divided in to several stages that form a pipeline, which is depicted in Fig. 10. This clear logical structure allows an implementation on multi-processor-systems.

Image acquisition: The first step is the image acquisition. The applied sensor is a asynchronous RGB camera to grab the current scene. Thereby, the camera is triggered by marks on the belt to capture the images at every revolution of the conveyor belt at exactly the same belt position. This stage provides an RGB image to the next stage.

Binary image generation: The second stage performs all the tasks described in the first subsections of section 2. The binary image is calculated by using the difference image method. In addition, the background adaptation by updating the reference image is performed. All these tasks can be computed during one pass. Because only pixel-based operations are performed, these tasks can be easily distributed on different threads to enable parallel processing. At the end of this stage the binary image is available.

Segmentation: At the segmentation stage the chain code representation for the following steps is computed. The next stages process only on the chain code data and do not need the matrix representation of the binary image. For example, three regions are extracted after the processing of image I in Fig. 10. Now the regions 1, 2 and 3 are available in chain code representation. After the processing of image II – which takes place later – the regions 4 and 5 are available as well.

Cutting of connected regions: Now the chain code representations of the regions reach the first cutting stage. Region 1 in Fig. 10 has a higher compactness than the threshold and, therefore, is not handled by the cutting algorithm and is directly transmitted to the object generation. Region 2 has a constriction which is detected by the cutting algorithm and is cut into the regions 2a and 2b. The ratio for cutting is too low for the third region 3, thus, the region remains unchanged. The region 2a does not touch the upper

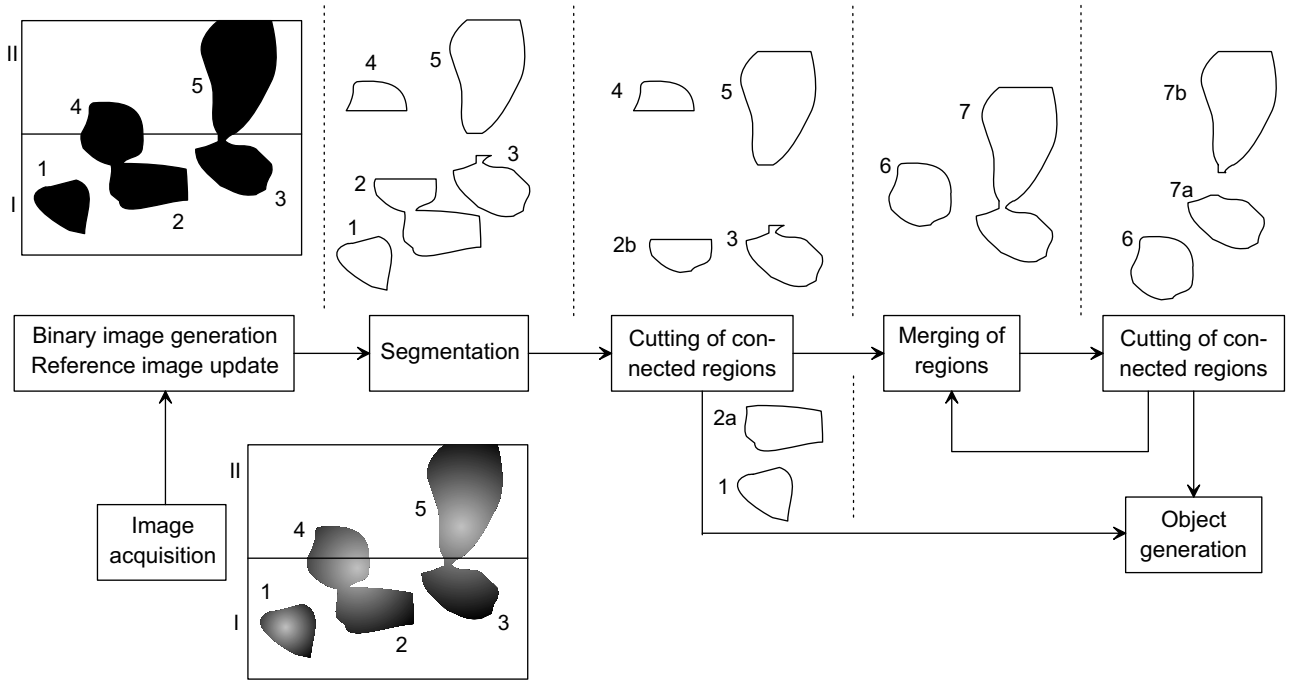

Fig. 10. All stages of the process pipeline

border of the image I and is transmitted to the object generation. Regions 2b and 3 touch the upper image border and have to be processed by the merging stage after the next image has arrived.

Merging of regions: The transmitted regions have to wait until the next image (image II) is captured. In our example regions 2b, 4 and 3, 5 are merged into 6 and 7. After merging they are transmitted to the second cutting stage.

Cutting of connected regions: This stage is necessary to handle regions such as 7. Due to the merging process, the region has now a new constriction. Regions 6 and 7a are directly transmitted to the object generation. Region 7b touches the image border at the top and is going to be merged with a region in the next image.

Object generation: This is the last step of the object detection system. At this stage the color information stored during the binary image generation is attached to the object. The result of this stage is a stream of objects, each represented by its chain code and its color information for further feature extraction tasks.

6. RESULTS AND CONCLUSION

We have presented a very fast and robust object detection system to detect arbitrary objects placed on a moving but repeating backgrounds. The main advantages are:

- Robust and reliable object detection by adapting the difference image method for moving but repeating backgrounds.
- Adaptable reference images which are able to handle both slight variations of the illumination or abrasion and sudden variations by sticking objects or light changes.
- A new very fast cutting technique to split up overlapping objects based only on shape information of the objects.
- The processing of the objects in real-time on standard PCs.

On a Pentium III processor with 550 MHz we are able to localize and cut up to 500 objects per second even if 50 % of the objects are overlapping each other.

7. REFERENCES

[1] Simon A. Brock-Gunn, Geoff R. Dowling and Tim J. Ellis, Tracking using Colour Information, In: *Proceeding of the 3rd International Conference on Automation, Robotics and Computer*, Singapur, November 1994.

[2] B. Chanda, Application of binary mathematical morphology to separate overplapping objects, In: *Pattern Recognition Letters*, 13, 1992, pp. 639-645.

[3] Herbert Freeman, Computer processing of line-drawing images. *Computing Survey*, 6(1), March 1974.

[4] John M. Gauch, Noise removal and contrast enhancement, In: S. J. Sangwine and R. E. N. Horne (eds.), *The Colour Image Processing Handbook*, Chapman & Hall, London, 1998, pp. 149-162.

[5] Thanarat Horprasert, David Harwood and Larry S. Davis, A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection, In: *Frame-Rate99*, 1999.

[6] Bernd Jähne, Image Processing for Scientific Applications, CRC Press, Boca Raton, 1997.

[7] Christof Ridder, Olaf Munkelt and Harald Kirchner, Adaptive Background Estimation and Foreground Detection using Kalman-Filtering, In: *Proceedings of International Conference on Recent Advances in Mechatronics*, Istanbul, August 1995, pp 193-199.

[8] John C. Russ, The Image Processing Handbook, Springer-Verlag, Heidelberg, 1998.

[9] Robert J. Schalkoff, Digital Image Processing and Computer Vison, John Wiley & Sons, Inc., New York, 1989.