Correspondence and Clustering Techniques of a Stereo Vision System for the Detection of Obstacles around Aircraft in Aerodrome Areas

J. Gauci*, D. Zammit-Mangion*
***Cranfield University**

Keywords: *correspondence, clustering, stereo vision, obstacle detection*

Abstract

Ramp and taxiway regions are the most congested areas of an aerodrome. A lot of activity goes on in this environment and aircraft are surrounded by several types of obstacles, including other aircraft and vehicles. This paper discusses the correspondence and clustering techniques of a stereo vision system that can be installed onboard an aircraft to detect and track generic obstacles around the aircraft during ground manoeuvres.

1 Introduction

The most congested area at an airport is the ramp. This is a very dynamic environment, with several aircraft taxiing in and out of the stands and parked aircraft being refueled, loaded/unloaded and boarded simultaneously. Aircraft are situated very close to each other, making it demanding to manoeuvre an aircraft in such confined spaces. Taxiways are also very busy, with multiple aircraft moving between the runway and the ramp and queuing to enter the runway. Many different types of obstacles are found on ramps and taxiways, including aircraft, vehicles and fixed structures.

Although several precautions are taken to prevent ground collisions between aircraft and obstacles, accidents still occur [1]. This suggests that current methods and systems only provide a partial solution to the problem. Of particular interest are collisions between two aircraft when taxiing. Reports of some of these collisions can be found at [2-6]. The collisions investigated in these reports involve large commercial passenger aircraft. In each of these accidents, the wing of a taxiing aircraft has come into contact with the wing or tail of a stationary aircraft. Most of the collisions between two aircraft occur in fine weather and good visibility. In most cases, the pilots of the taxiing aircraft are aware of the other aircraft but misjudge the separation between the two aircraft. Judging distances from the cockpit of a large aircraft is not a trivial task. In most cases, pilots either have a restricted view of the wingtips, an impaired view, or no view at all. Distance judgement is complicated by the fact that most commercial transport aircraft have swept wings and these are subject to an effect known as *swept wing growth* or *wing creep* [7].

A novel system is proposed in [8] and can be installed on an aircraft to provide further protection against incidents and accidents (particularly wingtip collisions) on ramps and taxiways. The solution proposed is an onboard non-collaborative system. This has the advantage of being completely independent of airport infrastructure and of other aircraft and obstacles. The platform assumed is a large transport aircraft since such an aircraft is expected to benefit most from this system. From this point onwards, this platform will be referred to as the *ownship*. The main functional requirements of the proposed system are (a) to detect and track obstacles around an aircraft during ground manoeuvres and (b) to alert the flight crew in the event of loss of separation or a potential collision.

To meet the first requirement, the system needs to detect generic obstacles. However, since the biggest threats are, by far, vehicles and other aircraft, more care needs to be taken to detect these types of obstacles. Also, for the system to have maximum effectiveness, it needs to focus on the most vulnerable areas of an aircraft. Accordingly, it therefore needs to focus mainly on obstacle detection around the wingtips. For this purpose, a rectangular protection zone is defined around each wingtip as shown in Fig. 1. The size of this zone was determined by taking into consideration a number of parameters, such as typical wingtip clearances, taxi speeds and pilot reaction times.

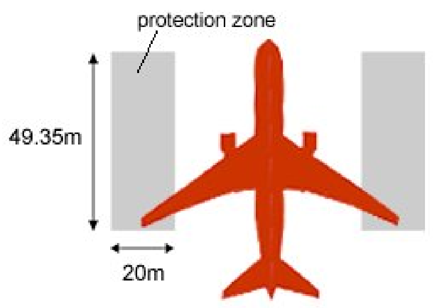


Fig. 1 Definition of a protection zone around the ownship's wingtips

The protection zone is the smallest region around a wingtip that has to be clear of obstacles such that, in the event of a conflict (where an obstacle is detected as being on a collision course with the wingtip), the pilots will have enough time and space to react and bring the ownship to a stop without colliding with the obstacle. This means that the system needs to focus on the detection and tracking of obstacles in the region beyond the boundaries of the protection zone. This is necessary in order to be able to detect potential collisions – and, hence, to issue timely warnings – before an obstacle penetrates the protection zone. If an obstacle enters the protection zone, it may be too late to avoid a collision.

The obstacle detection and tracking system has been implemented using stereo vision techniques, with a pair of stereo cameras located on either wingtip. The functional block diagram of the system is provided in Fig. 2. This paper focuses on two computational blocks of the stereo vision process: correspondence and clustering. These are typical steps of a stereo vision-based obstacle detection system. However, their implementation depends on the specific requirements of the application.

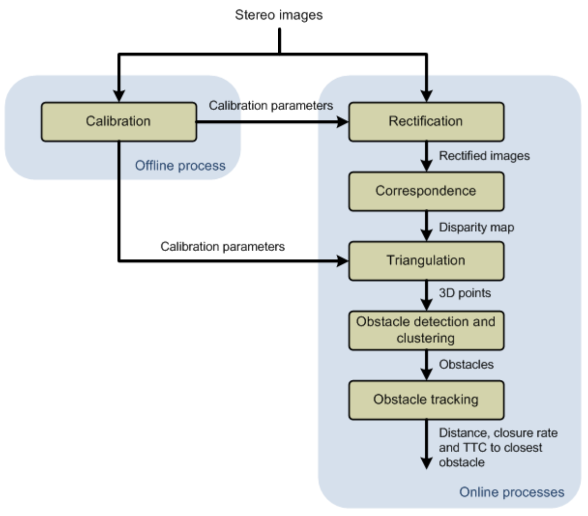


Fig. 2 Functional block diagram of the stereo vision system

2 Correspondence

As the name implies, correspondence is the process of finding corresponding pairs of pixels in the stereo images. This process is reduced to a 1D problem by warping the images during rectification (refer to Fig. 2). Rectification ensures that corresponding pixels have the same row coordinate. Correspondence methods can be broadly classified into two categories: those that produce a dense disparity map and those that produce a sparse disparity map.

Correspondence methods that produce a dense disparity map use intensity information and correlation techniques to compute the disparity of every pixel in the image. These methods are further divided into local and global optimisation methods. Local methods use local information in order to find corresponding points. These methods are also known as *window methods*. On the other hand, global correspondence methods use information from a larger region of the image in order to find the disparity at each pixel. A comprehensive review and evaluation of dense correspondence methods can be found in [9].

Correspondence methods that produce a sparse disparity map are also known as *feature-based correspondence methods*. This is because they only compute the disparity of particular image features such as edges or corners. The same correspondence methods that are used to obtain a dense disparity map can also be used to process only certain image features.

2.1 Outline of Algorithm

In this application, the aim is not to reconstruct the whole scene but to detect obstacles within the scene. For this reason it is only necessary to find the disparity of subsets of the image that are likely to correspond to obstacles. Therefore, a feature-based correspondence method using intensity information has been adopted. The image features that are processed are edge pixels. There are several reasons for processing edges. Firstly, edges contain the most important structural information about an image. They normally occur in textured image regions and are more likely to provide a reliable match. Most importantly, edges are a key feature of obstacles in the context of this application. Moreover, since edge pixels account for a small percentage of the whole image, computation time can be significantly reduced by using this approach. Edges are detected using the Canny edge detection method [10]. This edge detector removes noisy edges while preserving continuous edge contours.

The output of edge detection is a binary edge map with pixels either classified as edges (1) or non-edges (0). For each edge pixel, the disparity is found using a local, window-based method. This matches edge pixels in the left (reference) image to corresponding edge pixels in the right image as follows (refer to Fig. 3):

1. A square window of pixels is selected around a pixel $p_l(x_1,y_1)$ in the left image.
2. A similar window is selected around a candidate edge pixel, with the same row coordinate, in the right image.
3. The matching cost between the left and right image windows is computed.
4. Steps (2) and (3) are repeated for every edge pixel that is within the disparity search region.
5. The right edge pixel $p_r(x_2,y_1)$ corresponding to the minimum matching cost (global minimum) is identified.
6. The disparity d of $p_l(x_1,y_1)$ is given by:

$$d=x_1-x_2 \qquad (1)$$

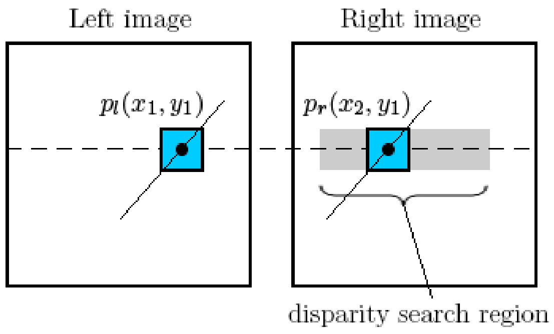


Fig. 3 Window-based correspondence

To compare intensity regions in the left and right images, the Sum of Absolute Differences (SAD) matching cost is used. Since the stereo vision system is to be used in an outdoor environment, it is important to compensate for photometric distortion. This is done by normalising the intensity windows. The matching cost between left and right image regions is therefore given by:

$$C(m)=\sum_{(x,y)\in W}\left|\left(\frac{I_l(x,y)-\mu_l}{\sigma_l}\right)-\left(\frac{I_r(x+m,y)-\mu_r}{\sigma_r}\right)\right| \qquad (2)$$

where:
$d_{min} \leq m \leq d_{max}$ is the disparity search range,
$C(m)$ is the matching cost at disparity m,
W represents the left and right image regions,
$I_l(x,y)$ and $I_r(x+m,y)$ are the intensities of pixels within the left and right windows respectively,
μ_l and μ_r are the average intensities of the left and right image regions respectively,

σ_l and σ_r are the standard deviations of the intensities of the left and right image regions respectively.

2.2 Selection of Window Size and Number of Windows

One important issue that can affect the quality of the disparity map is the size of the window used during the correspondence process. A small window is prone to noise but is computationally quick and all the pixels within the window are likely to be at the same depth (i.e. the disparity will be constant within the window). On the other hand, a larger window has a better Signal-to-Noise Ratio (SNR) but increases the processing time. Also, as the window size is increased, it is more likely that the disparity changes within the window. This means that a larger window has a higher probability of containing occluded pixels which lead to increased differences between the left and right image regions used for matching.

To choose a suitable window size, the correspondence algorithm was tested on 8 stereo images, the ground truth disparity maps of which were known. These are standard test stereo images and are available at [11]. Only the edge pixels were processed in each image. Figure 4 shows how the processing time[1] and the percentage of correct disparities change when increasing the window size from 3x3 pixels to 15x15 pixels. It can be observed that the greatest increase in the percentage of correct disparities occurs when the window is enlarged from 3x3 pixels to 5x5 pixels. Little improvement is observed for windows larger than 7x7 pixels. The processing time increases non-linearly as the window size is increased.

Since the biggest improvement in the accuracy of the disparity map is obtained when increasing the window size from 3x3 pixels to 5x5 pixels, a window size of 5x5 pixels would provide the ideal compromise between disparity map quality and computation time. However, in order to be on the safe side, a window size of 7x7 pixels was chosen for this work.

[1] The percentage increase in processing time is measured with respect to the processing time when using a window size of 3x3 pixels.

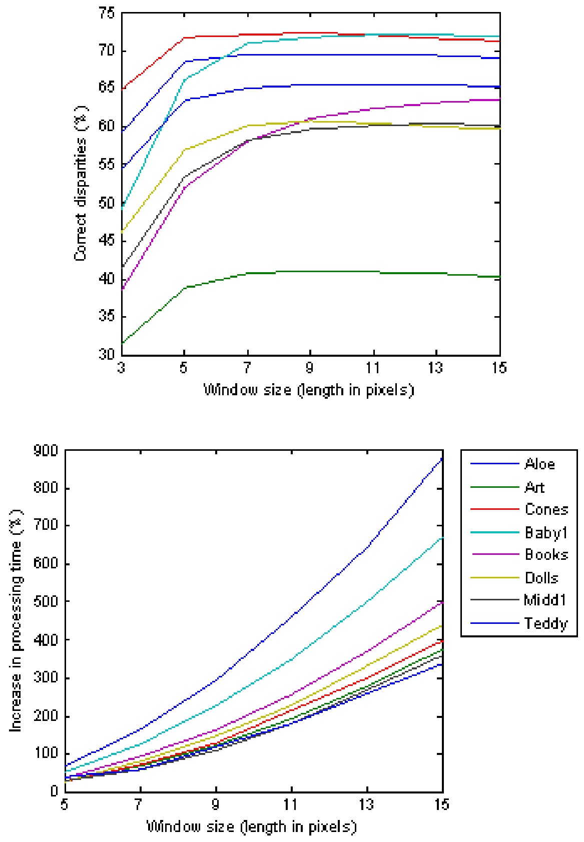


Fig. 4 Effect of correlation window size on percentage of correct disparities (top) and percentage increase in processing time (bottom)

Apart from window size, another important consideration is the number of correlation windows used. If only a single window is used to match left and right image regions, a bad match is likely to occur whenever the regions contain depth discontinuities or occluded pixels, since these tend to increase the difference between the two regions. This problem can be reduced by using multiple windows, where each window is situated at a slightly different position with respect to the pixel of interest [12].

To demonstrate the benefit of using a multi-window scheme, the correspondence algorithm was tested on 100 noisy synthetic images. Nine 7x7 correlation windows (shown in Fig. 5) were used during the correspondence process. When all of the images were processed, the total number of times that each type of window produced the best match (minimum matching cost) was expressed as a percentage of the total number of processed edge pixels. Another test was carried out to check the percentage increase in processing time when varying the number of correlation windows from 1 to 9. The results are shown in Fig. 6.

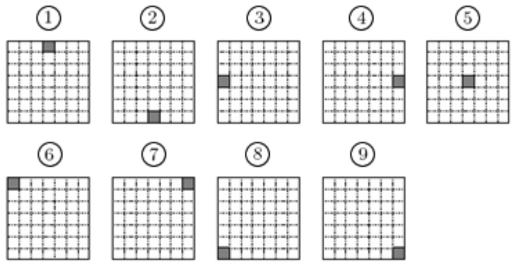


Fig. 5 Correlation windows (The grey pixel represents the pixel of interest)

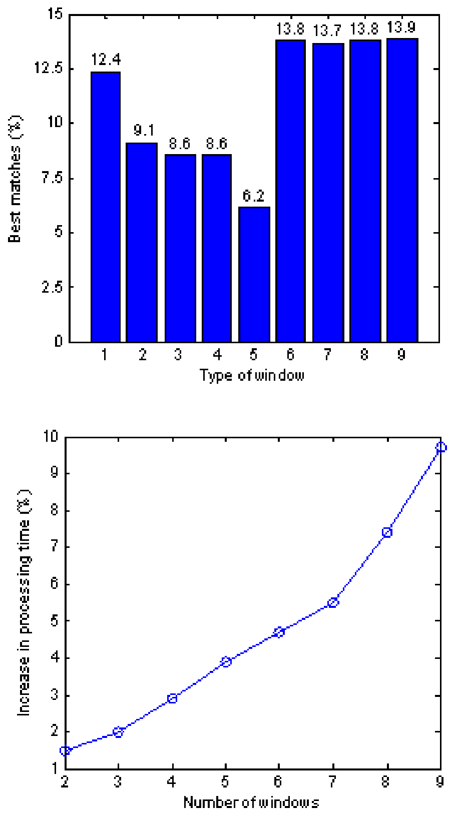


Fig. 6 Percentage of best matches provided by each type of window (top) and percentage increase in processing time with number of windows (bottom)

In the top figure, the numbers on the x axis represent the different types of windows shown in Fig. 5. As expected, the window that produces the least best matches is the central window (window type 5). Since most of the edge pixels occur at object boundaries, the central window has a greater probability of containing depth discontinuities. Therefore, the disparity is not constant within the window, leading to bad matches. The best matches are produced by window types 6-9, where the pixel of interest lies at the corner of the correlation window. These four windows account for over half of the best matches. Moreover, there is only a 2.9% increase in processing time when using four windows as opposed to a single window. Therefore, as a compromise between matching accuracy and processing time, a 4-window scheme (consisting of window types 6-9) was adopted in this work.

2.3 Detection of Incorrect Disparities

After estimating the disparity of a pixel, a number of tests are carried out to ensure that this disparity is reliable and accurate. The first test exploits the *uniqueness* constraint which states that a left image pixel should match uniquely with a right image pixel. Assume that, when finding the disparity of a left image pixel p_{l1}, the best match is provided by a right image pixel p_r, with matching cost C_1 (Fig. 7). Now assume that, when determining the disparity of another left image pixel p_{l2}, the best match is again provided by the right image pixel p_r, with matching cost C_2. This violates the uniqueness constraint and, therefore, one of the matches must be incorrect. In this case, the matching costs C_1 and C_2 are compared and the better match is accepted while the other is rejected.

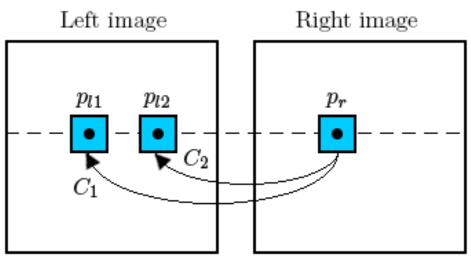


Fig. 7 Violation of the uniqueness constraint

If the uniqueness constraint is satisfied, a second test is carried out. This is called the *sharpness* test and it compares the disparity corresponding to the global minimum with the disparity associated with three pseudo-minima:

$$\Delta d = \sum_{i=1}^{3} \left| d_i - d_{\min} \right| \tag{3}$$

where:
d_{min} is the disparity corresponding to the global minimum,
d_i is the disparity corresponding to the pseudo-minima.

A large value of Δd implies that the pseudo-minima occur far from the position of the global minimum. In this case the match is considered to be ambiguous unless the matching cost of the global minimum is much smaller than that of the pseudo-minima. On the other hand, a small value of Δd implies that the pseudo-minima are close to the global minimum and the match is considered to be reliable even if the score of the global minimum is not much smaller than that of the pseudo-minima.

If Δd is larger than a certain threshold (due to an ambiguous match), a third test is carried out. This is called the *distinctiveness* test and it compares the error value of the global minimum with that of the pseudo-minima:

$$\Delta C = \sum_{i=1}^{3} \left| C_i - C_{\min} \right| \tag{4}$$

where:
C_{min} is the matching cost corresponding to the global minimum,
C_i is the matching cost corresponding to the pseudo-minima.

If ΔC is greater than a certain threshold, the disparity is considered to be valid; otherwise, it is rejected. The sharpness and distinctiveness tests were adopted from [13].

Figure 8 shows two correlation profiles obtained using the correspondence algorithm described. The top figure shows a profile where the location of the global minimum is clear and distinct. This profile satisfies the sharpness test and, therefore, the pixel disparity associated with it is considered to be valid. On the other hand, the bottom figure shows an ambiguous correlation profile. This is the result of repetitive texture. The location of the global minimum cannot be accurately determined from this profile and, therefore, it is successfully rejected by the sharpness and distinctiveness tests.

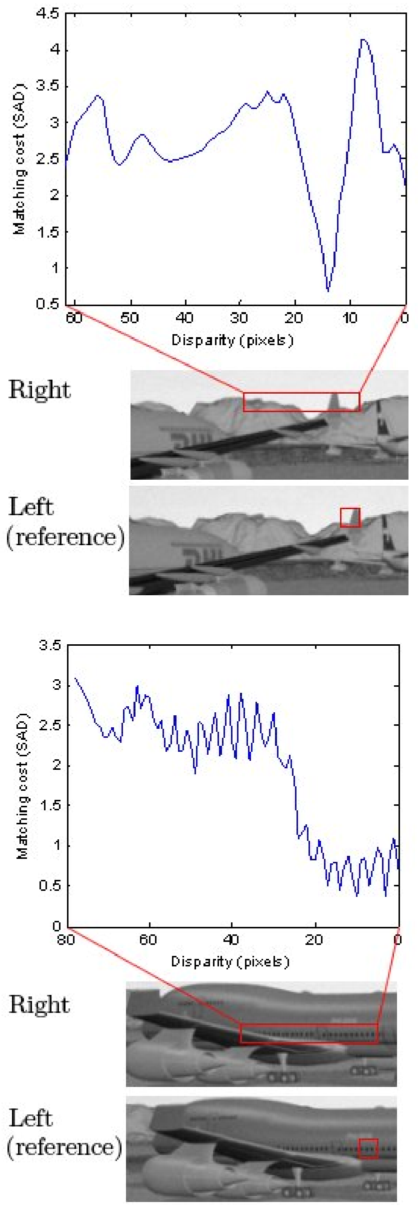


Fig. 8 Correlation profiles: reliable profile (top) and ambiguous profile (bottom) detected by the correspondence algorithm

2.4 Disparity Refinement

It is assumed that disparity varies smoothly over a very small region of pixels. Therefore, for every valid disparity, a second degree polynomial is fitted to the global minimum and its two closest neighbors to calculate the disparity with sub-pixel precision using Equation (5):

$$d_{subpixel} = d_{min} + \frac{C_{d_{min}-1} - C_{d_{min}+1}}{2(C_{d_{min}-1} - 2C_{d_{\min}} + C_{d_{\min}+1})} \qquad (5)$$

2.5 Reduction of Computation Time

One of the factors that affect the processing time of correspondence is the disparity search range. This depends on the stereo setup and on the size of the region over which obstacles need to be detected. In this application, the main region of interest is the protection zone. The disparity of features varies significantly within the protection area (from 208 pixels at 4m to 17 pixels at 50m). If each edge pixel is processed by using the full disparity search range, the computation time complexity will be high. One way of handling such a large range of disparities and reducing the processing time is by using multiresolution techniques [14,15].

The most common approach consists of sub-sampling the original stereo images in order to create an image pyramid, with very low resolution images at the top and increasingly higher resolution images towards the bottom of the pyramid. The coarse images contain approximate information about the scene. Consequently, correspondence is first carried out on the lowest resolution images using the full disparity search range. This provides rough estimates of disparity. Then, correspondence is carried out at the next level of the pyramid. This time, however, the disparity search range for each pixel (referred to as a *child* pixel) is restricted by the disparity of its *parent* pixel in the previous level. For example, assume that the disparity of a parent pixel is d_p and that the dimensions of images at each level of the pyramid are double those of images at the previous level.[2] Then, the disparity search range for a child pixel in the higher resolution image is reduced to:

$$2d_p - \tau \le d_{search}(pixels) \le 2d_p + \tau \qquad (6)$$

where τ is a user-defined tolerance value. This process is repeated for the remaining levels of the pyramid, with a smaller value of τ being used as image resolution increases. Hence, the disparity search range becomes narrower down the pyramid and a more precise estimate of disparity is obtained.

This strategy speeds up the overall processing time. However, the main disadvantage of this approach is that disparity errors tend to propagate down the pyramid. If the disparity of a parent pixel is incorrect and the true disparity of its child pixels falls outside the restricted disparity search range as a result, the computed disparity of the child pixels will also be incorrect. The correspondence algorithm will not be able to recover from such an error in the remaining levels of the pyramid. One way of trying to prevent disparity errors from propagating through the pyramid would be to use a larger tolerance value and hence widen the disparity search range. However, this would increase the processing time and would defeat the whole purpose of the multiresolution approach. For this reason, a slightly different approach is proposed here.

First, a low resolution version of the stereo images is obtained by sub-sampling the original images. The dimensions of the low resolution images are four times smaller than those of the original images.[3] Correspondence is carried out on the coarse images using the full disparity search range given by:

$$0 \le d_{search}(pixels) \le 52 \qquad (7)$$

Due to their very small size, the coarse images are processed very quickly. Then, the maximum and minimum disparities of the images (d_{max} and

[2] In this case, each parent pixel would have four child pixels.

[3] The maximum disparity of the original images is considered to be 208 pixels. This is equivalent to a disparity of 208/4 = 52 pixels in the coarse images.

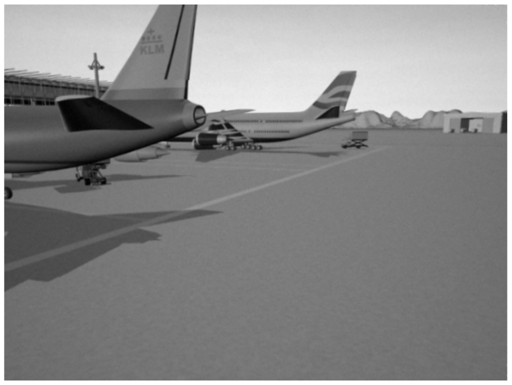
d_{min}) are determined. d_{max} and d_{min} are scaled and the original images are then processed using the following disparity search range:

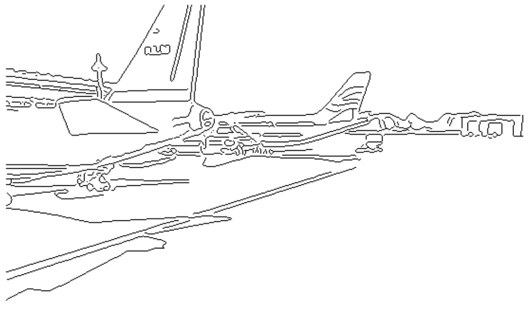
$$4d_{\min} \leq d_{search}(pixels) \leq 4d_{\max} \qquad (8)$$

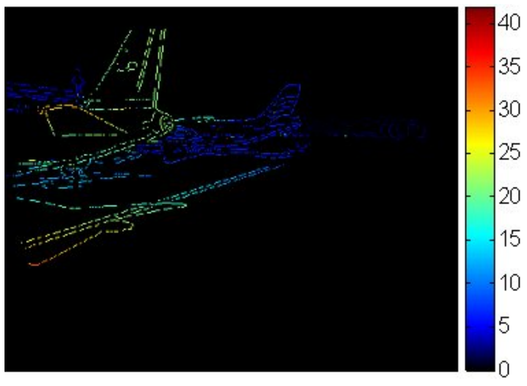
With this method, individual disparity errors that occur when processing the low resolution images are very unlikely to propagate to the original images. This is because the disparity search range for a child pixel in the original images is not directly linked to the disparity of its parent pixel. Therefore, as long as d_{max} and d_{min} are correct, the correspondence algorithm can recover from any disparity errors that occur when processing the coarse images. The main assumption being made in this implementation is that obstacles will only occupy part of the detection area during any single frame. Hence, the range of disparities will vary between frames and will rarely reach the boundaries of the full disparity search range. Therefore, computation time is significantly reduced by using this modified multiresolution approach.

2.6 Correspondence Results

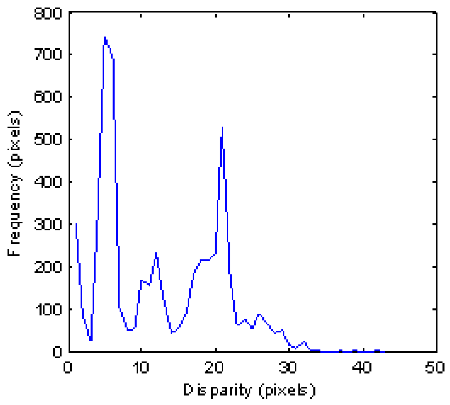
Figure 9 shows the results obtained when carrying out correspondence on a pair of noisy images featuring a typical aerodrome scene. As expected, objects that are closer to the cameras have a higher disparity. Also, the disparity varies smoothly over individual objects in the scene. The peaks in the disparity histogram give an indication of the number and size of objects in the scene. By looking at the edge map and the disparity map, it can be observed that the disparity of most edge pixels has been computed successfully. Some isolated pixels with incorrect disparities can be identified. These are pixels whose disparity differs significantly from that of neighboring pixels. Steps to remove these 'noisy' pixels are discussed next.


(a) Left intensity image


(b) Left edge map



(c) Edge disparity map



(d) Disparity histogram

Fig. 9 Correspondence results

3 Clustering

After correspondence is carried out, corresponding pairs of pixels are mapped onto 3D coordinates by means of triangulation. Assuming that the ground is flat in the vicinity of the ownship, obstacle points are then detected by measuring the height of each point above the ground. If the height exceeds a certain threshold, then the point is assumed to belong to an obstacle. However, due to errors in the previous stages of stereo vision, certain edge points are mapped onto incorrect 3D positions. As a result, height thresholding is not sufficient to filter out ground features and incorrect 3D points. Therefore, after detecting potential obstacle points, the next step is to group these points into individual obstacles. By the end of this process, the remaining 'noisy' points are removed while the true obstacle points are retained.

Individual obstacles are obtained from the set of potential obstacle points by means of clustering techniques. Clustering is the process of organising data sets (such as a set of 3D points) into groups (called *clusters*) based on a number of neighborhood and similarity criteria. Ideally, clustering should maximise the distance between clusters (also known as the *inter-cluster distance*) and minimise the distance between points in the same cluster (also known as the *intra-cluster distance*). There are two main classifications of clustering techniques: hierarchical clustering and non-hierarchical (partitional) clustering.

Hierarchical clustering is further divided into agglomerative and divisive clustering. Agglomerative clustering follows a bottom-up approach. Each point is initially treated as a cluster. Then, the two closest clusters are merged into a single cluster. This merging process is repeated by measuring the distance between clusters and merging the two closest clusters. As the clusters become larger, more distant clusters are linked together and the dissimilarity between elements of the same cluster increases. Eventually, all the points are grouped into a single cluster.

Divisive clustering follows a top-down approach. The points are initially treated as one cluster. This is repeatedly divided into smaller clusters until some stopping condition is met. In the limit, the number of clusters is equal to the total number of points.

The second category of clustering techniques is partitional clustering. These techniques divide the data set into clusters, typically by trying to minimise some criterion or error function.

3.1 Outline of Algorithm

The stereo vision system needs to be able to detect a wide range of obstacles, particularly aircraft extremities and vehicles. The number and size of obstacles in each frame is unknown. The shape of the obstacles, and the fact that only the edge pixels are processed, suggests that the clusters will tend to be elongated. Taking these points into consideration, it was decided to implement a new agglomerative, hierarchical clustering technique. As will be shown, this algorithm uses both spatial and non-spatial attributes to cluster obstacle points.

Initially, the obstacle points are treated as individual clusters. The clustering algorithm then proceeds as follows:

1. The first point is selected from the 3D point cloud and is labelled as *Cluster n*. This point is defined as the *root*. *n* is initialised to 1.
2. A search is carried out for points that are within a specific distance from the *root*. These points are defined as *children* of the *root*.
3. A search is carried out for points that are within a specific distance from each *child*. This step is repeated for any subsequent *children* until no more *children* are found. All *children* are assigned the same label (*Cluster n*) as the *root*.
4. All points labelled as *Cluster n* are removed from the 3D point cloud and *n* is incremented by 1.

5. Steps (1)-(4) are repeated until the 3D point cloud is empty and all the points are labelled.
6. Any clusters that do not satisfy certain criteria are removed.

The clustering process is illustrated through a simple 2D example in Fig. 10. It can be observed that *Cluster 1* is formed after 4 iterations of Step (3) whereas *Cluster 2* is formed after 2 iterations.

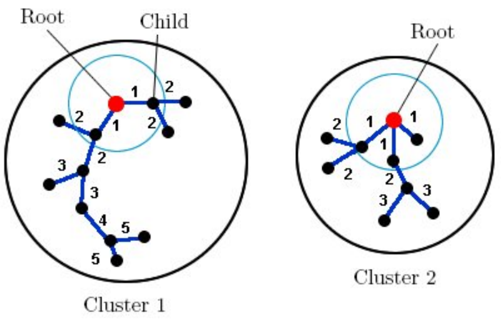


Fig. 10 Clustering

3.2 Grouping and Filtering Criteria

3.2.1 Grouping Criteria

During Steps (2) and (3) of the clustering algorithm, the distance between a point and the root/child is defined in terms of three grouping criteria (which are weighted such that the individual weights add up to 1):

- C_{3d} – The 3D Euclidean distance between the two points. This is the most important measure and is given the greatest weighting ($w_1 = 0.5$).
- C_{2d} – The 2D Euclidean distance between the pixels corresponding to the two points. This is an important measure which is made under the assumption that points that are close in 3D space are also close in the image plane. It is not as important as the first criterion because it is possible that neighboring points in the image plane are far apart in 3D space. This can happen, for instance, at object boundaries. This criterion is given a weighting of $w_2 = 0.4$.
- C_{int} – The absolute difference in intensity between the pixels corresponding to the two points. It is expected that, if the pixels belong to the same object and are situated close to each other, the intensity difference between them will be very small. However, it is also possible that completely unrelated pixels have the same intensity. For this reason, this measure is given the smallest weighting ($w_3 = 0.1$).

Rather than imposing ‘hard’ thresholds for each criterion (i.e. a criterion is either met or not), it was decided to use ‘soft’ thresholding where C_{3d}, C_{2d} and C_{int} are mapped onto score values between 0 and 1 as shown in Fig. 11. Since the three criteria have different units, the mapping also serves to normalise the data. Then, each score is multiplied by its corresponding weighting value and the overall distance D between the two points is calculated as follows:

$$D = w_1 S_{3d} + w_2 S_{2d} + w_3 S_{\text{int}} \quad (9)$$

where:
S_{3d}, S_{2d} and S_{int} are the score values associated with each criterion,
$w_1 = 0.5$, $w_2 = 0.4$ and $w_3 = 0.1$ are the weighting values associated with each criterion,
$w_1 + w_2 + w_3 = 1$.

As can be observed, D can vary between 0 and 1. If D is greater than a certain threshold *thres1* = 0.7, the two points are considered to belong to the same cluster.

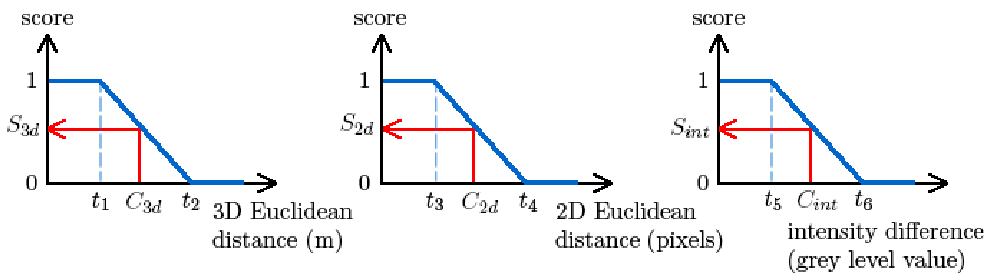


Fig. 11 Mapping of grouping criteria onto score values

3.2.2 Filtering Criteria

In Step (6) of the clustering algorithm, the following weighted criteria are used to filter the clusters:

- $C_{\overline{3d}}$ – The average 3D distance between neighboring points in the cluster. This gives an indication of point density. The greater the density, the more likely it is that the cluster is an obstacle. This criterion is given a weighting of $w_4 = 0.4$.
- $C_{\overline{2d}}$ – The average 2D distance between pixels corresponding to neighboring points in the cluster. Like the first criterion, this gives an indication of point density. However, because of the possibility that neighboring pixels might be far apart in 3D (as mentioned when describing C_{2d}), this criterion is given a smaller weighting ($w_5 = 0.3$).
- C_{pts} – The number of points in the cluster. This gives an idea of obstacle size. Given the different sizes of obstacles that are likely to be present in the scene, the number of points can vary considerably. Furthermore, the number of points does not only depend on obstacle size but also on the distance from the camera. Nevertheless, it can still be assumed that very small clusters are due to noise. This criterion is assigned a weighting of $w_6 = 0.3$.

As in the case of the grouping criteria, soft thresholding is used and $C_{\overline{3d}}$, $C_{\overline{2d}}$ and C_{pts} are mapped onto score values between 0 and 1 as shown in Fig. 12. Then, each score value is multiplied by its corresponding weighting value and the overall score S is calculated as follows:

$$S = w_4 S_{\overline{3d}} + w_5 S_{\overline{2d}} + w_6 S_{pts} \qquad (10)$$

where:
S_{3d}, S_{2d} and S_{pts} are the score values associated with each criterion,
$w_4 = 0.4$, $w_5 = 0.3$ and $w_6 = 0.3$ are the weighting values associated with each criterion,
$w_4 + w_5 + w_6 = 1$.

S can vary between 0 and 1. If S is greater than a certain threshold *thres2* = 0.7, the cluster is assumed to be valid; otherwise, it is removed.

The density of points in 3D space decreases with increasing distance from the cameras. This is mainly due to triangulation uncertainty which affects range resolution and positional accuracy. Hence, this factor needs to be taken into account when normalising C_{3d} and $C_{\overline{3d}}$. For this purpose, thresholds t_1, t_2, t_7 and t_8 increase linearly with distance as follows:

$$\begin{aligned} t_1 &= k_1 z \\ t_2 &= k_2 z \\ t_7 &= k_1 \bar{z} \\ t_8 &= k_2 \bar{z} \end{aligned} \qquad (11)$$

where:
z is the z coordinate of the point (such as the root) whose children need to be determined,
$\bar{z}$ is the average of the z coordinates of all the points within a cluster,
k_1 and k_2 are positive constants.

Suitable values for thresholds t1..t12 used in the clustering algorithm were determined experimentally by varying the threshold values and running the algorithm on different synthetic images.

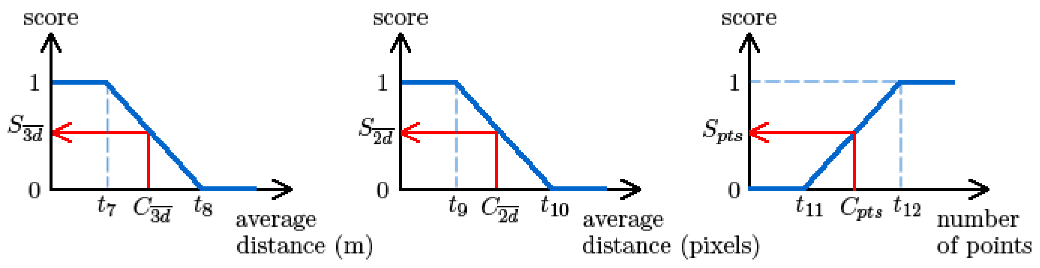


Fig. 12 Mapping of filtering criteria onto score values

3.3 Clustering Results

Figure 13 shows an example of the results of clustering. In this scenario, the ownship is approaching an aircraft which is of a similar size to it. From Fig. 13(c), it is apparent that obstacle detection based on height thresholding is not sufficient to eliminate all of the 'noisy' points. A lot of noisy points are still present and some of them are actually inside the protection zone. Figure 13(d) shows the significant improvement obtained after clustering (the clusters are represented by different colours in Figs. 13(b) and 13(d)). Most of the noisy points, including those that were previously detected in the protection zone, are removed. At the same time, however, the points corresponding to the aircraft and to the hangar behind it, are preserved. The shape and orientation of the aircraft is clearly visible in Fig. 13(d) and it is also observed that, since the hangar is further away from the ownship, the points corresponding to it (represented by the green and red clusters) are more dispersed than the points corresponding to the aircraft. The clustering algorithm is able to detect such distant obstacles by adjusting its thresholds dynamically.

4 Conclusion

This paper has presented the correspondence and clustering techniques of a stereo vision system to detect and track generic obstacles around an aircraft on ramps and taxiways. These techniques have been shown to be effective by testing them with noisy, synthetic, stereo images that are representative of the aerodrome environment.

Future work will focus on additional testing – using synthetic images generated under various illumination, visibility, and image noise conditions, as well as real images captured at an actual airport.

References

[1] University of Malta (Malta). Internal Safety Study. Unpublished. 2006.

[2] AAIB – Extract from Bulletin No. 7/1996: EW/C1995/11/04. Technical report, Air Accident Investigation Board (AAIB), 1996.

[3] AAIB – Extract from Bulletin No. 9/2005: EW/C2004/07/03. Technical report, Air Accident Investigation Board (AAIB), 2005.

[4] AAIB – Extract from Bulletin No. 12/2005: EW/C2004/11/01. Technical report, Air Accident Investigation Board (AAIB), 2005.

[5] ATSB Transport Safety Investigation Report, Aviation Occurrence Report 200600524. Technical report, Australian Transport Safety Bureau (Australia), 2006.

[6] Ground collision between two Airbus A320s, Denver, August 3, 2005. Technical report, National Transportation Safety Board (NTSB), 2005.

[7] Wing Tip Clearance Hazard. http://www.skybrary.aero/index.php/Wing_Tip_Clearance_Hazard. (last accessed January, 2009).

[8] Gauci J. *Obstacle Detection in Aerodrome Areas Through the Use of Computer Vision*. PhD thesis, Cranfield University, UK, 2010.

[9] Scharstein D. and Szeliski R. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, Vol. 47, No. 1, pp 7-42, 2002.

[10] Trucco E. and Verri A. Introductory Techniques for 3-D Computer Vision. Prentice Hall, 1998.

[11] Middlebury Stereo Datasets. http://vision.middlebury.edu/stereo/data/. (last accessed May, 2009).

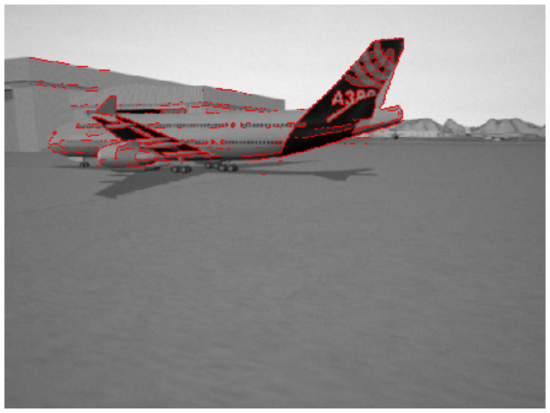
[12] Fusiello A., Roberto V., and Trucco E. Efficient Stereo with Multiple Windowing. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR97)*, pp 858-863, 1997.

[13] Di Stefano L., Marchionni M., and Mattoccia S. A PC-based Real-Time Stereo Vision System. *Machine Graphics and Vision International Journal*, Vol. 13, No. 3, pp 197-220, 2004.

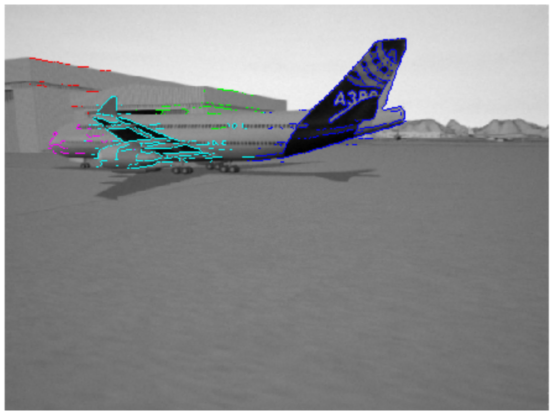
[14] Iocchi L. and Konolige K. A Multiresolution Stereo Vision System for Mobile Robots. *In AIIA (Italian AI Association) Workshop on New Trends in Robotics*, 1998.

[15] Magarey J. and Dick A. Multiresolution Stereo Image Matching Using Complex Wavelets. *In 14th International Conference on Pattern Recognition (ICPR)*, Vol. 1, pp 4-7, 1998.

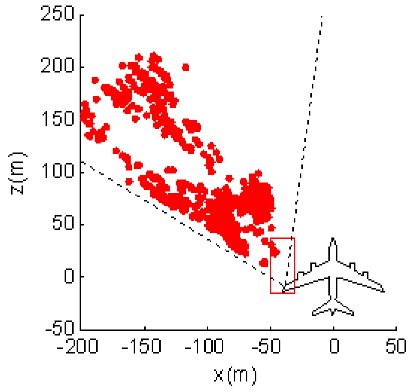
Copyright Statement



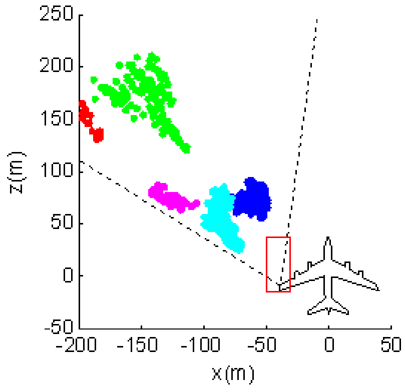
(a) Obstacle points superimposed on intensity image before clustering



(b) Obstacle points superimposed on intensity image after clustering



(c) Plan view of obstacle points before clustering



(d) Plan view of obstacle points after clustering

Fig. 13 Clustering results