Multiscale beamlet transform application to airfield runway detection

Samir Sahli and Yunlong Sheng

Image Science group, Center for Optics, Photonics and Lasers, Laval University,
Quebec, G1K 7P4, Canada, samir.sahli.1@ulaval.ca, sheng@phy.ulaval.ca

ABSTRACT

The context-driven target recognition requires the object-of-interest (OOI) to be first detected. We use the multiscale beamlet transform to detect airport runways as the OOI for detecting the aircraft. The up-to-down strategy in the beamlet graph structure is used for the connectivity and directional continuation of the edges, which are first detected in a coarse scale and are then refined at several finer scales.

Keywords: Context-driven target recognition, beamlet transform, multiscale analysis

1. INTRODUCTION

The wavelet transform and the optical wavelet transform are well known in optical pattern recognition community. The beamlet transform is a new multiscale transform proposed by Donoho *et al.*[1] along with the curvelet transform and wedgelet transforms [2–4]. These transforms are beyond the wavelet transform in the sense that they retain not only the location and scale, but also directional information, which is essential in the image and is not retained in the wavelet analysis. Human vision is inherently a multiscale phenomenon and is sensitive to orientation and elongation of one stimulus [5]. Edges are dominant features in the human vision. Based on these facts, the beamlet transform kernel functions (beamlets) are line segments with all possible orientations in the dyadic squares. The beamlet transform detects image edges by the matched filter with a line segment as template. This approach is more robust to noise than the local derivative-based edge detectors. Moreover, the beamlet transform performs multiscale edge detection, edge linking and edge feature analysis in a uniform tree framework. The beamlet pyramid provides a multiscale sparse and optimal representation for the smooth edges in the image. Although the computation of the beamlet transform at all the multiple scales with finest resolution could be more expensive, the beamlet operations for connectedness and directional continuation of the edges are performed much faster and more robustly than pixel-by-pixels edge tracking process in conventional derivative-based edge detectors[6].

The beamlet transform has been used in different applications: with a normalized histogram of orientations, Sampat *et al.*[7] categorized shapes of round masses in mammographic; with a pre-processing using the steerable filters, Berlemont *et al.*[8] detected DNA filaments in noisy images obtained by fluorescent microscopy. The beamlet pyramid is used in most of those applications [9]. In this paper, we apply, for the first time to our best knowledge, the multiscale beamlet up-to-down process in the beamlet graph data structure to detection of airfield runways. From the *a priori* knowledge on the estimated runway length and width, we apply the beamlet transform at a chosen scale level and then refine and complete of runways detection using multi-scale coarse-to-fine beamlet analysis. The runways can serve as the object-of-interest for the context-driven detection of aircraft, which would be nearby. This context-driven detection method is useful only when detecting the object-of-interest is easier than detecting the targets themselves. The beamlet transform permits searching for regions-of-interest, which contain the runways, by scanning the aerial image without segmentation. In addition, the beamlet transform offers location and orientation information of the object-of-interest. Experimental results will be presented.

2. BEAMLET TRANSFORM

2.1 Beamlet and wedgelet transforms

The beamlet transform is performed in the dyadically partitioned squares of an image. An image of size $[0, 1]^2$ is partitioned into four squares of side $[0, 1/2]^2$, each of them is partitioned into four squares of side $[0, 1/4]^2$. The dyadic

Optical Pattern Recognition XIX, edited by David P. Casasent, Tien-Hsin Chao,
Proc. of SPIE Vol. 6977, 69770G, (2008) · 0277-786X/08/$18 · doi: 10.1117/12.777578

partitioning continues recursively, such that for an image of each side of $n = 2^J$ pixels, at a scale level j with $0 \le j \le J-1$ we have 2^{2j} dyadic squares with the sides of $n2^{-j}$ pixels. After the dyadic partition a quantum resolution δ is used to divide the boundaries of a dyadic square into vertices. Then, the beamlets are defined as line segments connecting any pairs of two vertices on the boundaries of the dyadic square, except the vertices that are on the same side of the dyadic squares. The beamlet transform of a continuous image $f(x_1, x_2)$ is the collection of the line integrals along all the beamlets $b \in \beta_{n,\delta}$:

$$T_f(b) = \int_b f(x(l))\, dl \tag{1}$$

where $x(l)$ traces out b along a unit path, $\beta_{n,\delta}$ is the set of beamlets belonging to all dyadic squares at all scales. For a digitized image the digital beamlet transform integral (1) should be computed with help of interpolation functions[10]. The number of vertices in one dyadic square is equal to $M = 4n2^{-j}/\delta$ at the scale j. The number of the beamlets in all the dyadic squares at all the scale levels of an image of *nxn* pixels is estimated in the order of $(n/\delta)^2 \log_2(n)$. The computation cost of the beamlet transform can be reduced by taking dyadic values of the resolutions $\delta > 1$. The beamlet coefficients are normalized as

$$B_s(b) = T(b)/\sqrt{L(b)} \tag{2}$$

where *L(b)* is the length of the beamlet *b*. The beamlet transform of an image results in a collection of the beamlet components, which are characterized at each scale level $0 \le j \le J-1$ by a series of beamlet vectors consisting of the beamlet coefficients and the *In* and *Out* of the beamlets, denoting the locations of their vertices on the dyadic square boundaries, from which the orientation and the length of the beamlet *b* can be easily determined.

Beamlet transform can be applied to either gray-scale images or binary edge maps. In the multi-scale framework, any edges can be approximated in the dyadic squares up to some fine scale by line filaments. For detecting the edges in the dyadic square, the beamlet line integral is performed as a matched filter. It is well known that the matched filter is optimal against random noise, so that the beamlet approach is more robust against noise than the local-derivative-based edge detectors in detecting the line filament embedded in random background noise[11].

In the case of gray-scale image, the wedgelet transform should be used instead of the beamlet transform for detecting edges. In the dyadic squares up to some fine scale, any regions in a gray-scale image can be approximated as piecewise constant with curved boundaries between the pieces of the image. The wedgelet transform searches for an optimal division in the dyadic squares of the image by wedge-shaped piecewise constant regions, such that the approximation error between the image and the wedge-shaped regions is minimal[12]. Then, the wedgelet coefficient is defined as the difference in the gray-scale levels between the two piecewise constant regions, which are separated by the line segment *b* which is equivalent to a beamlet.

2.2 Beamlet data structures

The beamlet transform vectors, which offer scale, location and orientation information of the image, are organized into the hierarchical multi-scale pyramid or into the beamlet graph structures for further data exploration.

2.2.1 Beamlet Pyramid

The beamlet pyramid is a tree structure. First, the recursive dyadic partition (*RDP*) of an image generates the dyadic squares. After a complete recursive partition, one depicts all the dyadic squares at each scale level into one plane to form a quad-tree pyramid. In the highest plane at the coarsest scale there is one square of image size *nxn*. In the lowest plane at the finest scale level there are $2^{2(J-1)}$ squares with side dimension $n2^{-(J-1)} = 2$ pixels.

There are lines connecting the centers of the squares at one scale j to that at the lower scale $j+1$ constituting a quad-tree, as shown in Fig.1(a). Then, one thresholds the beamlet coefficients in each dyadic square based on the maximum beamlet statistic and looks for the maximum beamlet coefficient over all the beamlets as

$$C_s = \max_{b \in S} \{B_s(b)\} \tag{3}$$

where *S* is a collection of all the beamlets in the dyadic square *s* in such a way that one dyadic square is "decorated" by only one beamlet with the maximum coefficient *Cs*. In the beamlet pyramid the image is represented by a sparse

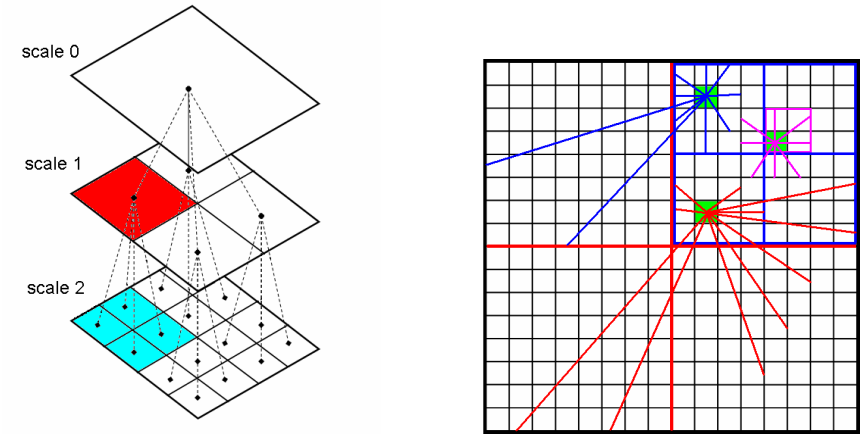

Fig. 1. (a):Beamlet pyramid with complete dyadic partition. Scale 0 is the coarsest and 2 is the finest. (b): Beamlet graph: Vertices belonging to different dyadic squares and scales are linked by the beamlets with different colors.

sequence of beamlets. For this purpose, a bottom-up pruning operation in the quad-tree structure is performed with the following criterion

$$C_{s_1}^2 + C_{s_2}^2 + C_{s_3}^2 + C_{s_4}^2 - 4\lambda \le C_s^2 \tag{4}$$

where C_s^2 is the "energy" of the square *s* which is decorated by a single beamlet, as defined in Eq. (3) and C_{si}^2 with *i = 1, 2, 3, 4* are the energies of the four dyadic squares, which are the children of the parent square *s*, λ is a penalty parameter. If inequality (4) is satisfied, i.e. the energy of the square *s* is more important than the sum of the energies of its four children squares, then the four children squares at a lower level are replaced by their parent square at a higher level. Otherwise, the parent square is replaced by its four children squares. In Eq. (4) a high value of λ favors the parent rather than the children. The bottom-up process prunes the hierarchical beamlet pyramid to a sparse representation of the image by an incomplete recursive dyadic partition. The whole beamlet pyramid structure can be further optimized to maximize the energy of the pyramid, which is an optimal sparse representation of the smooth curves embedded in an image[1].

The image can be reconstructed from the beamlet pyramid. In the reconstruction, a curve in the image is approximated by several beamlets coming from squares of different scales. To avoid overlapping, the pruning process is inhibitory. If one beamlet is chosen for the reconstruction, then other beamlets belonging to its descents can no longer be exploited.

2.2.2 Beamlet Graph

The beamlet graph is another data structure, in which all dyadic squares in all scale levels and all $(n+1)^2$ vertices corresponding to the *nxn* pixels in the image are presented in the same graph. The beamlets coefficients are computed according to Eq. (1) in the dyadic squares at all scale levels. In the beamlet graph one still thresholds the beamlets, but with no restriction for retaining only one maximum beamlet coefficient per one dyadic square, as in the case of beamlet pyramid, so that a dyadic square can be decorated by more than one beamlets that survived the thresholding.

The surviving beamlets are plotted in the beamlet graph connecting their respective vertices. Unlike the beamlet pyramid, in the beamlet graph, one vertex in the image could be connected to other vertices located beyond its nearest-neighbor vertices. Hence, relationships between any vertices at any scales are possible. Beamlet line segments colored in Fig.1(b) illustrate the possible relationships. Beamlets with different colors are used to show relationships through different scales.

Image analysis through the beamlet graph structure is more effective than the pixel-by-pixel edge tracking and linking processes using the conventional edge extraction algorithms. In fact, the beamlet line chaining process by connectedness and directional continuation of the beamlets needs to be performed only at the ends of the beamlet line segments, but not at every point on the beamlet. Moreover, the curves of any shape can be approximated by paths in the beamlet graph with the beamlets of multiple scales. This is an important advantage compared with the conventional edge extraction in a single scale, as will be shown in our application examples in the next section.

2.2.3 Cost computation

The *RDP* results in a collection of 2^{2j} dyadic squares number at the scale *j*. The number of vertices in one dyadic square is order to $M=4n2^{-j}/\delta$ at the scale *j*. Consequently at all the scale levels of an image of *nxn* pixels, the number of beamlets is estimated in the order of $(n/\delta)^2 log_2(n)$. All possible beamlets need to be evaluated for either finding the maximum beamlet statistic in the pyramid beamlet or keeping the beamlets whose values are higher than the threshold in the beamlet graph. In the both cases, two tasks would be performed:

- Estimate coordinates pixels along unit paths of each possible beamlets.
- Evaluate the linear integral shown in Eq.(1).

One notices that the Eq.(1) will be computed for all the beamlets. Consequently the computation of the beamlet transform at all multiple scales with finest resolution could be expensive. Yang *et al.* [7] greatly reduces the complexity by computing the coordinates of the pixels on the beamlets using the symmetry and parallelism properties of lines segments in the dyadic square. Theirs algorithm reduces about 60% time the computation time compared to traditional interpolation method.

Another issue is prospected by Berlemont *et al.*[8] to speed up the evaluation of beamlet coefficients. An approximation of beamlet coefficients based on the two-scale recursion technique[13] which compute all possible line integrals in at most O*(nlogn loglogn)* operations, where *n* is the number of data point. The computation of a complete *RDP* with an initial *1024×1024* image takes approximately 1s on a dual processors based computer.

The problem of computational cost is tackled by us in the beamlet graph structure, where we compute the beamlet transform only at the coarsest scale, then at other fine scales we need to compute just beamlets which belong to regions where a edge refinement is required.

3. EXPERIMENTS

We apply the beamlet transform to detecting airport runways in aerial surveillance imagery. The airport runways represent a unique feature in the aerial images of the ground, and can serve as objects-of-interest for attracting attention for detecting the aircraft, which must be nearby. The context-driven object detection strategy presumes that the objects-of-interest can be reliably detected in possible complex configurations. One would not expect to spend the similar or more energy to detect the objects-of-interest as that to detect the object itself. Ideally, the runways should be detected without expensive segmentation of the entire large aerial image scene.

3.1 Multi-scale up-to-down beamlet analysis

In the aerial surveillance, the large image scenes are examined bloc by bloc. Suppose a bloc of size *256 x256* pixels in an airport image is under investigation. We first applied the Canny edge detector to the image and obtained the edge map as shown in Fig.2(a). In our application the conventional derivative-based edge detectors can readily detect edges in the images, so that we do not need to use the wedgelet transform for detecting edges. The challenge now is to detect the runways among many irrelevant noise edges in the edge map. One of the discriminant features of the runways is their long length. However, the long edges of the runways are broken by taxi ways and other roads, which intersect the runways. Other discontinuities on the edges result from the thresholding in the Canny edge detector against high amplitude of local gradients due to image noise. Moreover, the runways themselves may contain a number of parallel broken edges, including not only the runway edges but also sign lines and landing tire marks, as shown in the green box of Fig. 2(a). These discontinuities are such a constraint that detecting runways edges via the sole criterion of edge length can fail. Conventional edge tracking and linking process or the Hough transform may be usually used to tackle the problem.

We present an alternative approach to the problem by application of the multi-scale up-to-down beamlet analysis. The basic idea in this application is to first use the beamlet transform at a coarse scale to detect the "*seed*" or promising edges of the runways. Thresholding the beamlet coefficients would be less sensitive to edge discontinuities, than thresholding the edge lengths for selecting the seed beamlets, because by the definition a beamlet connecting one vertex to another crosses continuously an entire dyadic square. The selected seed beamlets at the coarse scale are then refined by the beamlet transforms through multiple scales down to the finest scale in order to determine the true edges of the runways.

To extract the runways, we computed the beamlet transform of Fig. 2(a) at the scale levels of j=*1, 2, 3 and 4*, corresponding to the dyadic square with the sides of *128, 64, 32 and 16* pixels, respectively. At the coarsest scale *j=1*, the dyadic square size of *128 x128* is compatible with the airfield size and the runway length, which can be estimated a

priori according to the fly altitude and the camera parameters. For the binary edge map the beamlet coefficient *T(b)*, as defined in Eq. (1), corresponds to the number of edge pixels on the beamlet b. The beamlet coefficients are normalized by square root of the beamlet length $B_s = T(b)/L(b)$, instead of the length *L(b)*. This normalization favors the long edges for having larger B_s, so we can detect the seed beamlets by thresholding the normalized beamlet coefficients B_s. We chose the quantum resolution $\delta = 4$ *and* 2, for the scale $j=1$ *and* 2, which correspond to dyadic squares with the sides of *128 and 64* pixels respectively. This choice reduces the computation time and lets the number of vertices to be a constant value of 128 for both scales $j=1$ *and* 2. For other finer scales $j > 2$, we let the resolution $\delta=1$.

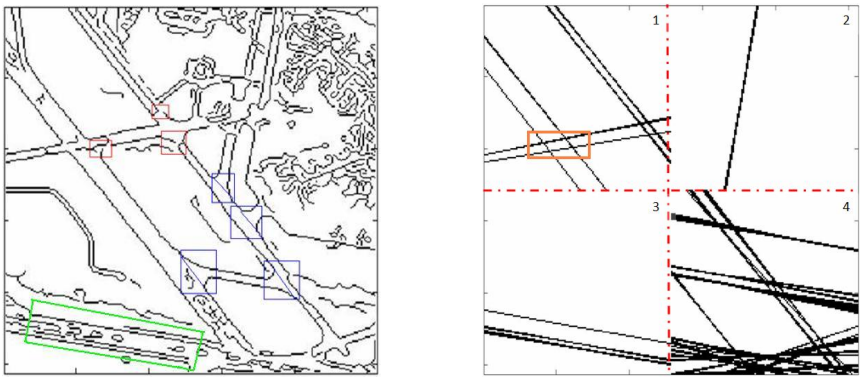

Fig. 2. (a): Edge map of a part of airport. Discontinuities of edges are due to intersections with taxi ways (blue boxes), to edge detector (red boxes) and to the nature of runways (green box); (b): Thresholded seed beamlets in the dyadic squares of *128 x128* pixel size.

The beamlet transform integrals are computed along all possible beamlets in a dyadic square. Thus, a threshold must be applied to eliminate irrelevant beamlets. In the beamlet pyramid, the beamlet decorated recursive dyadic partition (*BD-RDP*) operation eliminates all the beamlets except the one with maximum B_s. As a result, one dyadic square is decorated by at most one beamlet. This process is not appropriate to our application for runways detection because if a section of runway falls into one dyadic square then its long and parallel edges will present several peak values of the beamlet coefficient B_s and the *BD-RDP* would retain only the maximum beamlet and eliminate all the secondary edges of the runways. The runway detection however relies on the detection of at least two parallel edges separated by the runway width, so that the secondary edges, lost in the thresholding, must be recovered from the beamlet transforms at finer scale levels. This recovering process can be expensive. We therefore use the beamlet graph data structure in our experiments. At the coarse scale in the dyadic squares with thesides of *128* pixels we selected the beamlets B_s whose values are higher than 50% of the maximum B_s, which is computed for a possible diagonal edge in the same dyadic square. The thresholding operation is a critical step for extracting the seed edges. We prefer a low threshold value to keep all possible edges associated to the runways and to avoid loss of the runway edges. On the other hand, we have a large margin for keeping short beamlets and broken segments, which can be treated in the refinement process.

The beamlets in the coarsest scale $j=1$ surviving the thresholding are presented in Fig. 2(b). In the dyadic square 1, edges of one runway and two taxi ways are all extracted, and other short edges in Fig. 2(b) are removed by the thresholding. Both the runway and taxi way beamlets continue through the intersections where the physical edges do not exist, as can be seen in the orange box in Fig. 2(b), because by the definition, the beamlets continue across the dyadic square. These error edges can be removed in the beamlet graph at finer scales.

We see in Fig. 2(b) that the seed beamlets in square 1 defining the runway edges are not prolonged in the top-right corner of square 3, or in the bottom-left corner of square 2, due to the competition with other beamlets in the thresholding in the same squares 2 and 3. The edge segments localized in the corners of the squares 2 and 3 are short. Their respective beamlet coefficients B_s do not exceed the threshold. These lost segments of the runway edges cannot be recovered at the same coarse scale level $j = 1$. On the other hand, the thresholded beamlet in square 2 is a false alarm, and there are more false alarms in the square 4 as a consequence of thresholding with the low threshold value of 50%.

3.2 Refining edges at finer scale levels

The beamlet graph at the coarsest scale permits to extract long edges of large structures as the runways, in spite of discontinuities on the edges. However, some useful edge segments can be lost in the thresholding of beamlet coefficients

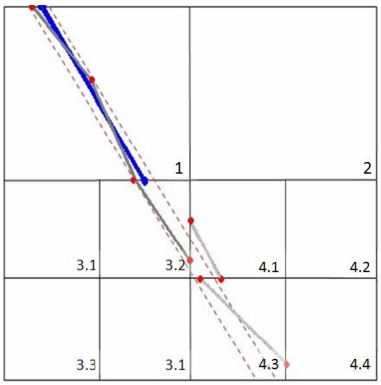

Fig. 3. Refinement of a beamlet with δ-collinear testing. The dotted lines show the error tolerance of 2Δ

to more strong beamlets presented in the same dyadic square. On the other hand, there are false continuations of the edges by the beamlets at some locations, where the physical edges are not present. We refine the detected coarse edges by the beamlet transforms down to the several finer scale levels through the directional continuation and connectedness operations in the beamlet graph, where the beamlets of different scales are brought together.

The basic procedure is illustrated in an example shown in Fig. (3), where a seed beamlet at the coarse scale *j=1* is with blue color in a dyadic square 1. We then focus attention to its neighbourhood including that of its prolongations beyond the borders of its own dyadic square 1 to dyadic squares 3.2, 4.1 and 4.3 in Fig. (3). Then, at the finer scale *j=2*, we selected the beamlets, whose extremity vertices have distances to the coarse seed beamlet with *j=1* or its prolongation shorter than a predetermined error tolerance Δ, which could be defined differently at different scales. If such finer beamlets exist, then the coarse seed beamlet in blue color is replaced by the new finer beamlets in gray color in square 1 as shown in Fig. (3). The gray colored beamlet inside square 3.2 is restored as the prolongation of the coarse blue beamlet. Those fine beamlets inherit the position and orientation information of the seed beamlet and are closer to the real edges than the coarse beamlet. The beamlets of gray color in square 4.1 and 4.3 at *j=2* scale level are rejected because the distances from their vertices to the seed beamlet prolongation are larger than the tolerance Δ.

Next, the beamlets resulting from refinement at the scale *j=2* are considered as new seed beamlets, and are refined by the beamlets at a finer scale *j=3*. The process is applied recursively until the finest scale. This process is referred to as the *δ-collinear testing* for directional continuation and connectedness of the beamlets. It is an up-to-down multiscale refinement operation.

The refinement process has been applied to the seed beamlets in the dyadic squares of *128 x128* pixels as shown in Fig.2(b) and result in Fig.4(a), 4(b) and 4(c) for scales *j=2, 3 and 4* in the dyadic squares with the sides of *64, 32 and 16* pixels, respectively. The green line segments in Fig.4(a) and 4(b) are the runway edges, which were lost when thresholding the coarse beamlet at scale *j=1* with the dyadic square of *128* pixels sides, and are recovered in the beamlet graph at finer scales *j=2 and 3*.

Two types of false continuation errors are marked in the red boxes in Fig.4(b). The first type of false continuation occurs because the beamlets always continue from one vertex to another across the dyadic square. The false continuation shown in orange boxes in Fig.2(b), 4(a), 4(b) and 4(c) is also in this type. In addition, if the true edge does not through the square then the beamlet can be longer than the true edge as shown in the small red box in the bottom-right corner of Fig.4(b).

The second type of false continuation occurs because of the directional continuation of the seed beamlet, as shown in the large red box in the top-right of Fig.4(b). Indeed, the seed beamlets in the dyadic squares with the sides of *64* pixels are prolonged from top-left quarter to top-right quarter of the image. The latter attract many beamlets in the dyadic squares with sides of *32* pixels. Those beamlets come from the image noise but are quite strong to exceed the threshold in their respective dyadic square.

The first type of false continuation has been corrected by the refinement at a finer scale *j=4* in the dyadic squares with the sides of *16* pixels, as shown in the blue box in the bottom-right corner of Fig.4(c). The first type of false continuation in the intersection of the runway, shown in the orange box in Fig.2(b) were partially corrected also by the refinement at the scale *j=4*. The second type of false continuations was corrected partially in the same process, as shown in the blue box in the top-right corner of Fig.4(c). Both types of false continuations could be further corrected by pursuing the refinement process through more finer scales *j=5, 6 and 7*. However, we stopped the refinement process at

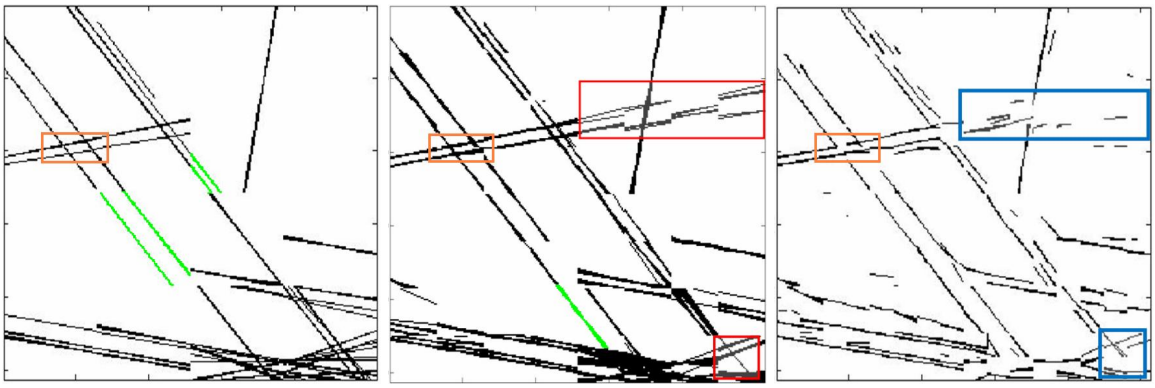

Fig. 4. Refinement of the seed Beamlets (a) at scale j=2 with dyadic square of 64 x64 pixels. Green line segments are recovered runway edges lost at j=1 as shown in Fig.2b; (b) at scale j=3 with dyadic square of 32 x32 pixels. Red boxes enclose false continuation; (c)Beamlets at scale j=4 with dyadic squares of 16 x16 pixels. Blue boxes enclose false continuation partially corrected.

the scale *j=4* and used the criterion of runway width to eliminate the remaining false continuation edges. Compared with the refined edge map Fig.4(c) and the original one in Fig.2(b) we have successfully extracted the straight line edges of the runways and eliminated the noise edges by using the multi-scale beamlet transform and the up-to-down refining processing in the beamlet graph structure.

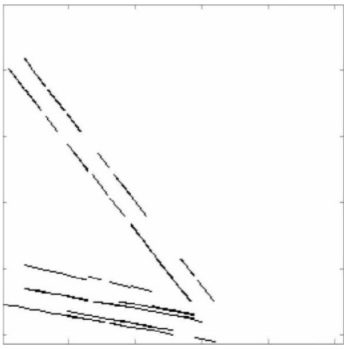

Fig. 5. Detected runways edges from the beamlets shown in Fig.4(c) using the a priori information about the width of runways. using the a priori information about the width of runways.

3.3 Discrimination of runways

In order to discriminate the aircraft runways from other auxiliary ways and roads we used the criterion of the runway width, which can be estimated *a priori*. The discrimination process was applied to the two lists of the beamlet coefficients. One is the list of the seed beamlets at the coarsest scale in the dyadic squares of *128 x128* pixels. Another is the list of the refined beamlets at the finest scale in the dyadic squares of *16 x16* pixels. We looked for the beamlets in the second list, whose two vertices In and Out have both the distances to any seed beamlets in the first list equal to the runway width within an error tolerance. Once such a beamlet was found, its weight is incremented by one. As the seed beamlets belong to different dyadic squares of *128 x128* pixels, an edge of the true runways can be represented by several seed beamlets, so that a beamlet in the dyadic square of *16 x16* pixels could pass the width test for several seed beamlets and have therefore its weight superior to one, while short isolated seed beamlets that are not edges of the true runways are not able to increase the weights of the beamlet in the dyadic square of *16 x16* pixels to more than one, and were eliminated readily by a thresholding on the weights.

In the second step, one repeated the same process, but within the same list of the refined beamlets in the dyadic squares of *16x16* pixels. In this list we tested the beamlets, whose two vertices *In* and *Out* have both the distances to any selected beamlets, which passed the first test, equal to the runway width within an error tolerance.

4. DISCUSSION

Figure 5 shows the result of the processing of discrimination using the runway width. One observes in Fig.5 two groups of parallel edges, which correspond to the two runways in Fig.2(a). All the edges belonging to other structures, such as terminal buildings and tax ways, are eliminated. The runway in the green box in Fig.2(a) is detected in the ambiguous environment with presence of many noise edges from sign lines and landing tire marks. Only one parasite edge on the grass land nearby this runway happens to be long, parallel to the runway and at a distance to the up edge of the runway close to the runway width, and therefore survives the whole process.

One note that in Fig.5 the edges are not presented by edge pixels as that in conventional edge detectors, but by the *In* and *Out* vertices of the extracted beamlets, so that the interruptions of the runway edges are not caused by the weakness of the method. Some gaps in edges correspond to the intersection of the runways with auxiliary ways, and some gaps in edges could be recovered when continuing the edge refinement process to finer scales with dyadic square of *8x8*, *4x4* and *2x2* pixels.

5. CONCLUSION

We have presented a comprehensive introduction to the beamlet transform. We have analyzed two data structures: beamlet pyramid and beamlet graph in views of the applications. We detected successfully the airport runways using the beamlet transform and without using segmentation. In this application, we chose a coarse scale in which the dyadic square size is comparable to the size of the airfield in order to detect the runway edges by thresholding the beamlet coefficients. Then, the beamlets at three recursive finer scales were used to refine the detected seed coarse beamlet and produce a good approximation to the runways edges. The runways were finally discriminated using the *a priori* scale information on the runway width.

References

[1] D.L. Donoho and X. Huo, "Beamlets and multiscale image analysis", Lecture Notes in Computational Science and Engineering: Multiscale and Multiresolution Methods (2001).

[2] E.J. Candès and D.L. Donoho. "Curvelets - a surprisingly effective *non*adaptive representation for objects with edges", Curves and Surfaces, 105-120, Vanderbilt University Press (2000).

[3] E.J. Candès and D.L. Donoho, "New tight frames of curvelets and optimal representations of objects with piecewise-c2 singularities", Comm. on Pure and Appl. Math **57**, 219-266, (2004).

[4] J.K. Romberg, M. Wakin and R. Baraniuk, "Multiscale wedgelet image analysis: fast decompositions and modeling", International Conference on Image Processing **3**, 585-588, (2002)

[5] D. Marr, *Vision: a computational investigation into the human representation and processing of visual information*, W.H. Freeman, San Francisco, (1982)

[6] X. Huo, "Minimax correlation between a line segment and a beamlet", Statistics &Probability Letters, **72(1)1**, 71-81 (2005)

[7] M.P. Sampat, A.C. Bovik and M.K. Markey, "Classification of mammographic lesions into BI-RADS shape categories using the beamlet transform", Proc. SPIE: Medical Imaging. **5747**, 16-25 (2005)

[8] S. Berlemont, A. Bensimon and J.-C. Olivio-Marin, "Detection of Curvilinear Objects in Noisy Image using Feature-Adapted Beamlet Transform", International Conference on Acoustics, Speech and Signal Processing, **1**, 1225-1228 (2007)

[9] M. Yang, Y. Peng, X. Zhou, "Multiscale linear feature extraction based on beamlet transform", Intelligent Computing in Signal Processing and Pattern Recognition. ICIC. **345**, 353-363 (2006)

[10] J.E Bresenham, "Algorithm for computer control of a digital plotter", IBM Systems Journal **4(1)**, 25-33 (1965)

[11] D.L. Donoho and X. Huo, "Beamlet pyramids: A new form of multiresolution analysis, suited for extracting lines, curves, and objects from very noisy image data", Proc. SPIE: Wavelet Applications in Signal and Image Processing VIII, **4119**, 434-444 (2000)

[12] F. Friedrich, L. Demaret, H. Fuhr and K. Wicker, "Efficient Moment Computation over Polygonal Domains with an Application to Rapid Wedgelet Approximation", SIAM Journal on Scientific Computing, **29(2)**, 842-863 (2007)

[13] A. Brandt and J. Dym, "Fast calculation of multiple line integrals ", SIAM J. Sci. Comput., **20(4)**, 1417–1429 (1999)