Pedestrian Detection using Stereo-vision and Graph Kernels

F. Suard, V. Guigue, A. Rakotomamonjy and A. Benshrair
PSI CNRS FRE 2645
INSA de Rouen
FRANCE
Email: frederic.suard@insa-rouen.fr

Abstract— **This paper presents a method for pedestrian detection with stereovision and graph comparison. Images are segmented thanks to the NCut method applied on a single image, and the disparity is computed from a pair of images. This segmentation enables us to keep only shapes of potential obstacles, by eliminating the background. The comparison between two graphs is accomplished with a inner product for graph, and then the recognition stage is performed learning is done among several pedestrian and non-pedestrian graphs with SVM method. The results that are depicted are preliminary results but they show that this approach is very promising since it clearly demonstrates that our graph representation is able to deal with the variability of pedestrian pose.**

I. INTRODUCTION

From many years now, a lot of works have been devoted to pedestrian detection. Although, the literature have shown very promising results based on different approaches, the problem is not yet fully solved. At the present time, the main research direction for solving this problem is to combine artificial vision and machine learning technique. Within this context, several methods have been tested. For instance, Papageorgiou et al. [10] have used a wavelet based-template combined to Support Vector Machines, whereas Zhao et al. [16] used stereo-vision for segmenting pedestrian from background, then a neural network architecture performs the recognition stage. Other related works dealt with shape-based methods [1], [2], [6], probabilistic template [8] and motion cues [14]. Similarely to artificial vision technique, many machine learning algorithms have been tested for addressing the detection and recognition stage : neural networks [16], [15], Adaboost [14], rigde regression [11] and Support Vector Machines (SVM) [10].

As stated by Shashua et al. [11], one of the reason why detecting pedestrian is challenging is the variability of pedestrian image : Pedestrians can appear under different poses, different textures and different position of the body articulation. This paper addresses this problem of pose variability by presenting a pedestrian representation based on the articulation structure of the body part. The overall idea is to separate the pedestrian to the background by means of stereovision and them representing all resulting objects with a skeleton. A graph is associated to such a skeleton. The advantage of using a skeletal graph to model the pedestrian shape is that the information on the shape and the structure of the body is embedded in the graph. Then the recognition stage aims at discriminating graphs generated by a pedestrian from objects.

Since, our objective is to discriminate some structured objects, we need a learning algorithm that is able to handle such non-vectorial description. Recently, kernel methods have been a very active research area and have proposed some efficient algorithms like SVMs or Kernel Principal Components Analysis. However, the most preminent advances have been achieved through the development of inner product for structured objects that can be used for SVMs. In this work, we will implement graph kernels in order to be able to use SVMs and to discriminate graph representations of pedestrian and non pedestrian objects.

This paper is organized as follows. Section II describes the methodology that has been used for obtaining graph representation of objects. Section III gives a short presentation about graph kernels, SVMs and how we have used them in the context of pedestrian detection. Results are depicted in Section IV, followed by summary and conclusions in Section V.

II. METHOD DESCRIPTION

The main idea of this method is to describe a pedestrian as a graph. As we can see in 3(a) , a pedestrian could be easily cut into several parts :

- a head
- one or two arms
- one body
- one or two legs

These parts could be represented separately as an edge, and all parts are linked with nodes. Using graph has some advantages :
first, a graph is less influenced by noise. Even if edges are very disturbed, the structure of the shape is not modified. Another interesting point is that this method could solve the problem of occlusion. If the pedestrian is partially occluded, it is even possible to detect it, thanks to the rest of the graph. The occlusion could be solved thanks to the use of stereovision. The stereovision enables us to distinguish pedestrians and obstacles based on the road from the background. Another interest of this method is that two graphs can be easily compared, since the comparison is applied with less than 20 features. And finally, a pedestrian could have different positions and sizes, but can be still recognized. Indeed, methods for

pedestrian recognition are based on rigorous constraints, such as geometric symmetry, front sied view, no neighbor. But with graph we can recognize any pedestrian since we have some characteristic elements.

A. *Image Segmentation*

The first step consists in extracting the requested features from the images. In our case, the features are all elements which could be considered as obstacles. Different kinds of segmentations are possible. Since we will process graphs, we have to deal with region segmentation. We use the method called 'NCut' which segments an image into a defined number of regions. The particularity of this method is to treat image as a graph $G = (V, E)$, with V vertices of graph and E edges of graph. So the segmentation could be defined as a graph partitioning problem. The image is originally made of pixels, which become nodes in the graph. If two pixels are similar, they are linked with an edge in the graph. The weight *w(i,j)* applied on edges is function of the similarity between nodes *i* and *j*. When the totality of image is transformed into a graph we could partition it. The partitioning criterion is called *normalized cut*. It measures the dissimilarity between different groups. For example, the degree between two sets A and B is called the *cut* : $cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$
The optimal partioning is obtained when *cut* is minimal. In order to avoid unnatural partioning, the criteria has to be normalized : $Ncut(A, B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$ where $assoc(A, V) = \sum_{\in A, t \in V} w(u, t)$

For more information, the reader can refer to [12].

One interest of this method is that we can chose the number of regions, and we can test different configurations for our method. If their are too many regions, it is possible that all obstacles will be splitted into two or three parts, and if they are too few regions, obstacles can be disturbed by their neighbors. We decided to conserve more regions, and made a compromise about : 30 regions.
Then we use the stereovision to merge some regions and eliminate the background. The result is shown on 3(b). The principle of stereovision is simple, when a point in left image is correlated with a point in right image, we can obtained its disparity by computed the gap between their respective position (refer to [9]). To make the correlation easier and efficient, we capture video sequences with a particular camera configuration : the cameras are placed in the same horizontal line, so the vertical disparity is null.

There are different possibilities for obtaining the disparity map. The natural method is to compute disparity for each pixel. But with a basic correlation algorithm, the computation time is too expensive, and the results are not accurate, since a lot of false disparities are found. We can also compute disparity for particular points like edges. In our case, we compute the disparity for each point resulting of the difference between the two images. If the result is positive, it means that the point does not belong to the background, and the pixel is significant. The next step consists in joining edges. This point is obtained by searching a pixel with the same disparity and presenting some similar and symmetric characteristics. Results are shown in 3(c)

Now we have to merge information from Ncut partitioning and disparity. For each region $R_i,\, i \in 1..N$, N number of regions , resulting from Ncut, we consider the disparity $D(R_i)$ for each pixel (of this region). If this region has a low disparity $mean(D(R_i)) < min\_threshold$, this region is considered as belonging to the background. Then, pixels in resulting image $I_{DispNcut}$ corresponding to the region are set to 0, $I_{DispNcut}(R_i) = 0$. Else, we compute the average disparity $mean(D(R_i))$ and all pixels are set to this value , $I_{DispNcut}(R_i) = mean(D(R_i))$.

Now, as we can see in 3(d), shapes are splitted into several parts. But some neighbor shapes have a similar disparity and we have to merge them. To do this, we look at each regions R_i and merge it with each neighbor region if their disparities are close :
$|D(R_i) - D(R_j)| < disp\_threshold, i \in 1..N, j \in 1..N$ where N number of regions and thus, $I_{DispNcut}(R_i, R_j) = mean(D(R_i), D(R_j))$

Finally we extract each shape, that is to say each non-null region of $I_{DispNcut}$, in a bounding box binarized it and compute its skeleton and graph.

B. *Graph designing*

In this part, we will describe our method used to transform a black and white image into a graph. This part is very important for continuation and is one particularity of our method.
The aim is to keep the main information contained in a shape, that is to say geometric properties or topologic properties. This last property is particularly interesting in our case. If a shape is made of a set of sub-parts, the skeleton will preserve the connectivity and the shape arrangement.

1) Finding the skeleton: A skeleton can be obtain from a binary image. A skeleton can be defined as a line representation of an object, that is to say it :

- is one-pixel large
- is around the middle of the object
- preserves the geometry and topology object

Given the definition of Lantuejoul [4], a skeleton subset of a black and white image $S_k(A)$ is defined as : $S_k(A) = E(A, kB) - [E(A, kB)\, o\, B]\; k = 0, 1, ...K$ where B is a structuring element, and K is the largest value of k before the set $S_k(A)$ becomes empty. The skeleton is then the union of the skeleton subsets : $S(A) = \cup_{k=0}^{K} S_k(A)$

2) Skeleton to graph: Once the skeleton is obtained, we can build the graph. A graph G is made up of vertices and edges $G = (V, E)$. A vertex is a junction between different edges. We can differentiate two types of vertices : nodes (V_n), which are a junction of many edges and vertices which are edge ending (V_s).
To build the graph we look at each pixel of skeleton the type

of the pixel. If the pixel as only one neighbor, this is an edge ending. If the neighoring corresponds to a defined mask, for example $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$, then this pixel is defined as a node.

Once all of the skeleton pixels have been tested, we can build the graph. Algorithm 1 shows the algorithm of this part.

First, we join each edge ending to a node by searching a way containing a sequence of skeleton pixels. The first step is to search each the existing edge for each edge ending, since only one node is linked with an edge ending. When all end vertices are linked, we can search all links existing between nodes vertices. This method is different, because in the precedent search, the stop condition was attempted when a node was found, but now, if another vertex is reached, it does not mean that all possible ways were tried. So the stop condition is different, and there exists at least one way to link a node vertex to a pixel of the skeleton, it is possible to reach another node vertex. The course of the skeleton is facilitate, because the skeleton is one pixel thin, so when the skeleton does not fill this condition, it means that a vertex is reached and the way is finished. An edge between the two nodes is established and other ways are continued until they reach an edge.
Comparing two graphs can be summarized as comparing labels placed on vertices and edges. We decided to put simple labels, that is to say the position for the vertices and the length for the edges. The positions are not normalized, in order to keep a scale. Indeed, position is function of the subimage (3(e)), and the graph size respects the proportion of original shape. We saw earlier that a graph comparison can make up with size, and this point seems to be in contradiction. But in fact, some results have shown that the scalar product between a small and a big pedestrian shape is really a scalar product between two shapes with the same size.

S is the skeleton of the shape : a black and white image with the original shape size. $F = S(p)$ means that F is a local window of S, which is 3 pixels high and 3 pixels wide, centered on pixel p. W is a set of ways linking different vertices.

This algorithm seems to be very complex, in terms of time computation, but in fact, since there are not a lot of vertices (between 10 and 30), the computation is fastly achieved.

3) Comments about the segmentation and graph construction: figure 3 presents some results coming from the segmentation. Figure 3(a) is one of the stereo images, this is the original image. Then, we apply the Ncut method and request 30 regions. The result is shown in figure 3(b). As we can see, the segmentation has damaged the majority of the shapes in the image. We also compute the disparity, figure 3(c). This last point does not seem to be significant, but we volontary apply constraints to the disparity computation in order to have the most valuable results. Then we mix the disparity with the

Algorithm 1 Algorithm for skeleton to graph transform

```
for $s \in V_s$ do
  $NodeReached = FALSE$
  $point = V_s(s)$
  while $NodeReached == FALSE$ do
    $F = S(point - 1 : point + 1)$
    if $\exists n \in F | n \in V_n$ then
      $NodeReached = TRUE$
      $E(s, n) = 1$
      $L_s(n) = point$
    else
      $point = p | F(p) == 1$
    end if
  end while
end for
for $n \in V_n$ do
  $W\{1\} = n$
  $point = V_n(n)$
  while $\exists W$ do
    for $i \in W$ do
      $point = W\{i\}$
      $F = S(point)$
      if $\exists n \in F | n \in V_n$ then
        $E(n, i) = 1$
        $L_s(n) = point$
        $clear W\{i\}$
      else
        $newpoint = p | F(p) == 1$
        $add(W\{i\}, newpoint)$
      end if
    end for
  end while
end for
```

Ncut, figure 3(d), in order to suppress the background and show off the 3D shapes. This figure is a brut result, with no filtering or region regrouping. Then for each remaining region, the disparity of which is non null, we compute their skeleton and graph. We take for example a shape of pedestrian (left subimage of 3(e)) and a shape of a column situed in front of the scene (right subimage). Finally in figure 3(f) graphs shows these previous shapes.

III. Kernel methods

This section describes the recognition stage based on SVMs and graph kernels.

A. SVM Classifier

The Support Vector Machines classifier is a binary classifier algorithm that looks for a optimal hyperplane as a decision function in a high-dimensional space [3], [13], [5]. Thus, consider one has a training data set $\{\mathbf{x}_k, y_k\} \in \mathcal{X} \times \{-1, 1\}$ where $\mathbf{x}_k$ are the training example and y_k the class label. At first, the method consists in mapping $\mathbf{x}_k$ in a high dimensional space owing to a function Φ. Then, it looks for a decision function of the form : $f(\mathbf{x}) = \mathbf{w}\cdot\Phi(\mathbf{x}) + b$ and $f(\mathbf{x})$ is optimal in the sense that it maximizes the distance between the nearest point $\Phi(\mathbf{x}_i)$ and the hyperplane. The class label of $\mathbf{x}$ is then obtained by considering the sign of $f(\mathbf{x})$. This optimization problem can be turned, in the case of L_2 soft-margin SVM classifier (misclassified examples are quadratically penalized), in this following one :

$$\min_{\mathbf{w},\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{k=1}^{m}\xi_k \tag{1}$$

under the constraint $\forall k, \quad y_k f(\mathbf{x}_k) \geq 1 - \xi_k$. The solution of this problem is obtained using the Lagrangian theory and it is possible to show that the vector $\mathbf{w}$ is of the form :

$$\mathbf{w} = \sum_{k=1}^{m} \alpha_k^* y_k \Phi(\mathbf{x}_k) \tag{2}$$

where α_i^* is the solution of the following quadratic optimization problem :

$$\max_{\alpha} W(\alpha) = \sum_{k=1}^{m}\alpha_k - \frac{1}{2}\sum_{k,\ell}^{m}\alpha_k\alpha_\ell y_k y_\ell K(\mathbf{x}_k,\mathbf{x}_\ell) \tag{3}$$

subject to $\sum_{k=1}^{m} y_k\alpha_k = 0$ and $\forall k, 0 \leq \alpha_k \leq C$, where $K(\mathbf{x}_k,\mathbf{x}_\ell) = \langle\mathbf{x}_k,\mathbf{x}_\ell\rangle$. According to equation (2) and (3), the solution of the SVM problem depends only on the Gram matrix K. Hence, in our case, the graph classification with SVMs only needs a graph kernel.

B. graph kernels

We use the inner product between graphical representations based on Kashima et al. paper's [7]. The idea is to compare two label sequences generated by two *synchronized* random walks on the two graphs. The operation is repeated until there is convergence of the result.

Given :

1) h_i^1 and h_j^2 the nodes i and j of graphs G^1 and G^2,
2) $p(h_i|h_{i-1})$ the transition probability from node h_{i-1} to node h_i,
3) $p_q(h_\ell)$ the probability to stop at node ℓ,
4) $K_n(h_k^1, h_k^2)$ the inner product between two nodes of the graphs G^1 and G^2,
5) et $K_a(a^1_{h^1_{k-1}h^1_k}, a^2_{h^2_{k-1}h^2_k})$ the inner product between 2 arcs a^1 and a^2 of G^1 and G^2.

The comparison of all paths, of all length, from all nodes in the two graphs, weighted by the path probability leads to :

$$\begin{gathered} K(G^1,G^2) = \\ \sum_{\ell=1}^{\infty}\sum_{h^1}\sum_{h^2} p(h_1^1)\prod_{i=2}^{\ell} p(h_i^1|h_{i-1}^1)p_q(h_\ell^1)\times \\ p(h_1^2)\prod_{j=2}^{\ell} p(h_j^2|h_{j-1}^2)p_q(h_\ell^2)\times \\ K_n(h_1^1,h_1^2)\prod_{k=2}^{\ell} K_a(a^1_{h^1_{k-1}h^1_k}, a^2_{h^2_{k-1}h^2_k})K_n(h_k^1,h_k^2) \end{gathered} \tag{4}$$

The detailed computation of $K(G^1,G^2)$ is in the paper of Kashima et al. [7] .

The complexity of this computation is $\mathcal{O}\left((|G^1||G^2|)^2\right)$, with $|G^i|$ the number of nodes in graph G^i. For this reason the number of nodes in each graph have to be the smallest as possible.

The graph kernel suggests that a kernel between nodes and a kernel between edges have to be defined. In our case, we have chosen to use a classical gaussian kernel since nodes and edges are labeled with vectorial values (position of the nodes and length of edges) :

$$K_n(x,y) = K_a(x,y) = \exp\left(-\frac{||x-y||^2}{\sigma}\right) \tag{5}$$

where σ is the bandwidth of the gaussian kernel.

Note that the graph kernel depends only on the probability transition between nodes and kernels between nodes and kernel between edges. This means that the label information of edges and nodes can be richer than it is at the present time. In fact, since we only need an inner product values, labels can be a non-vectorial data which admits a kernel.

IV. RESULTS

This section is devoted to some preliminary results that we obtain using our algorithm.

First of all, let us note that our benchmark images are composed from stereovision acquisition that has been captured indoor and under some good lightning conditions.

After the image processing procedure, we have a set of pedestrian and non-pedestrian structures that we have labeled manually. The number of pedestrian structure is 100 whereas we have about 1600 non-pedestrian images. Although, our two classes are not symmetric, the training set has been built so that each class is equally represented.

Since for our learning system, two parameters have to be chosen, namely C and σ, we have decided to analyze the performance of our algorithm for a large range of values of these two hyper-parameters. Then for each couple of C and σ the average results over 30 random drawn of the training set are then provided. Besides, in order to investigate how robust is our pedestrian representation, we have made the number of training set elements ℓ varying from 1 element per class to 75. For a given class, the number of elements in the test set is chosen to be 100 - ℓ

Results are reported in Figure 1. We have reported there the plot of classification rate and the area under the ROC curve with regards to the number of examples per class in the training set. the bottom figure shows an example of ROC curve for a number of example per class of 10.

These results are very preliminary results that have been obtained of pedestrian images that have not been captured in real situations. Besides, we only have few test elements. However, the results are still interesting. For a very low number of training examples, our algorithm is still able to achieve a good classification rate (about 0.7). As the number of example increases AUC and recognition rate also increases.

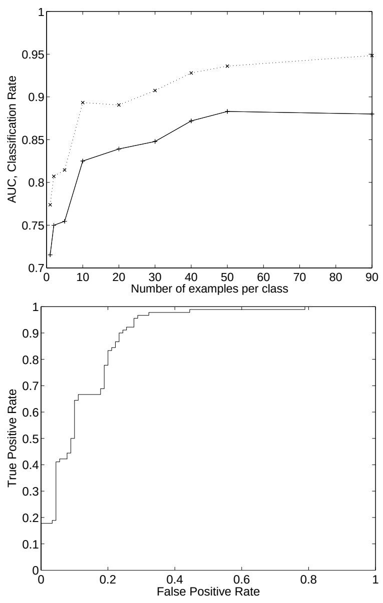

Fig. 1. These 2 figures shows the performance of the recognition stage. The top figure shows the classification rate (in solid line) and the Area Under the ROC curve (in dotted line) with regards to the number of example per class. the bottom figure gives an example of Area under the ROC curve for a number of example of 10.

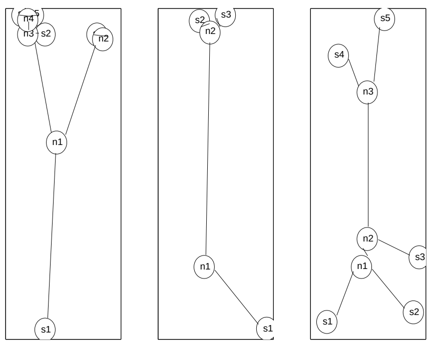

Fig. 2. This figure depicts some graph representations. On the left, we have a pedestrian graph of the learning set, on the middle a graph of a correctly recognized pedestrian, and on the right a correctly recognized non-pedestrian.

However, they do not increase anymore when one uses 50 or 90 training examples per class. All these results suggest that although pedestrians appear in different poses, our graph representation is able to capture some of these variabilities. Figure (2) illustrates such assertion. Graphs of 2 different pedestrians are depicted where the first graph corresponds one among the two the pedestrian training set and the second graph, which is somewhat different, is the graph of a correctly recognized pedestrian in the test set.

V. Conclusion

In this paper, we presented a kernel method based on graph for pedestrian recognition. Taking into account the fact that the results came from a preliminary study, they are encouraging and confort our idea of describing a pedestrian with a graph.

In the future, some improvements are planned. First we will add some information in the graph, particulary in the vertices : gray level, number of pixels, histogram of gradient like in [11]. The aim is to improve the classifier performance without increasing the computation complexity. Besides, we plan to extend our experiments to real-world pedestrian situations and to a larger number of testing set. An other point is the segmentation. The Ncut method gives good results, but takes also too many time. We will try other methods, faster and as reliable. If the method reveals to be satisfiing, we will envisage other applications, in particular multi-class applications, that is to say we will add other elements like cars, bicycle and other obstacles in the learning.

References

[1] M. Bertozzi, A. Broggi, R. Chapuis, F. Chausse, A. Fascioli, and A. Tibaldi. Shape-based pedestrian detection and localization. In *IEEE Intl. Conf. on Intelligent Transportation Systems 2003*, 2003.

[2] M. Bertozzi, A. Broggi, and A. Fascioli. Shape-based pedestrian detection. In *IEEE Intelligent Vehicles Symposium 2000*, 2000.

[3] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

[4] S. Beucher C. Lantuejoul. On the use of the geodesic metric in image analysis. *Journal of miscrocopy*, 121(1):39–49, 1981.

[5] N. Cristianini and J. Shawe-Taylor. *Introduction to Support Vector Machines*. Cambridge Univeristy Press, 2000.

[6] D. Gavrila and J. Geibel. Shape- based pedestrian detection and tracking. In *Proceedings of IEEE Intelligent Vehicles Symposium*, pages 215–220, 2000.

[7] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieh International Conference on Machine Learning*, 2003.

[8] H. Nanda and L. Davis. Probabilistic template based pedestrian detection in infrared videos. In *IEEE Intelligent Vehicles Symposium*, 2002.

[9] R. Debrie P. Miche. Fast and self adaptative image segmentation using extended declivity. *Journal of miscrocopy*, 50(1):3–4, 1995.

[10] C. Papageorgiou and T. Poggio. Trainable pedestrian detection. In *Proceedings of the 1999 International Conference on Image Processing*, pages 35–39, 1999.

[11] A. Shashua, Y. Gdalyahu, and G. Hayon. Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. In *Proceedings of IEEE Intelligent Vehicles Symposium*, 2004.

[12] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[13] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[14] P. Viola, M. Jones, and D. Snow. Pedetrian using patterns of motions and appearance. In *IEEE Int. Conf on Computer Vision*, pages 734–741, 2003.

[15] C. Wohler, J. Aulanf, T. Portner, and U. Francke. A time delay neural network algorithm for real-time pedestrian detection. In *IEEE Int. Conf on Intelligent Vehicle*, 1998.

[16] L. Zhao and C. Thorpe. Stereo and neural network based pedestrian detection. *IEEE Trans. on Intelligent Vehicles Transportation systems*, 1(3):148–154, 2000.

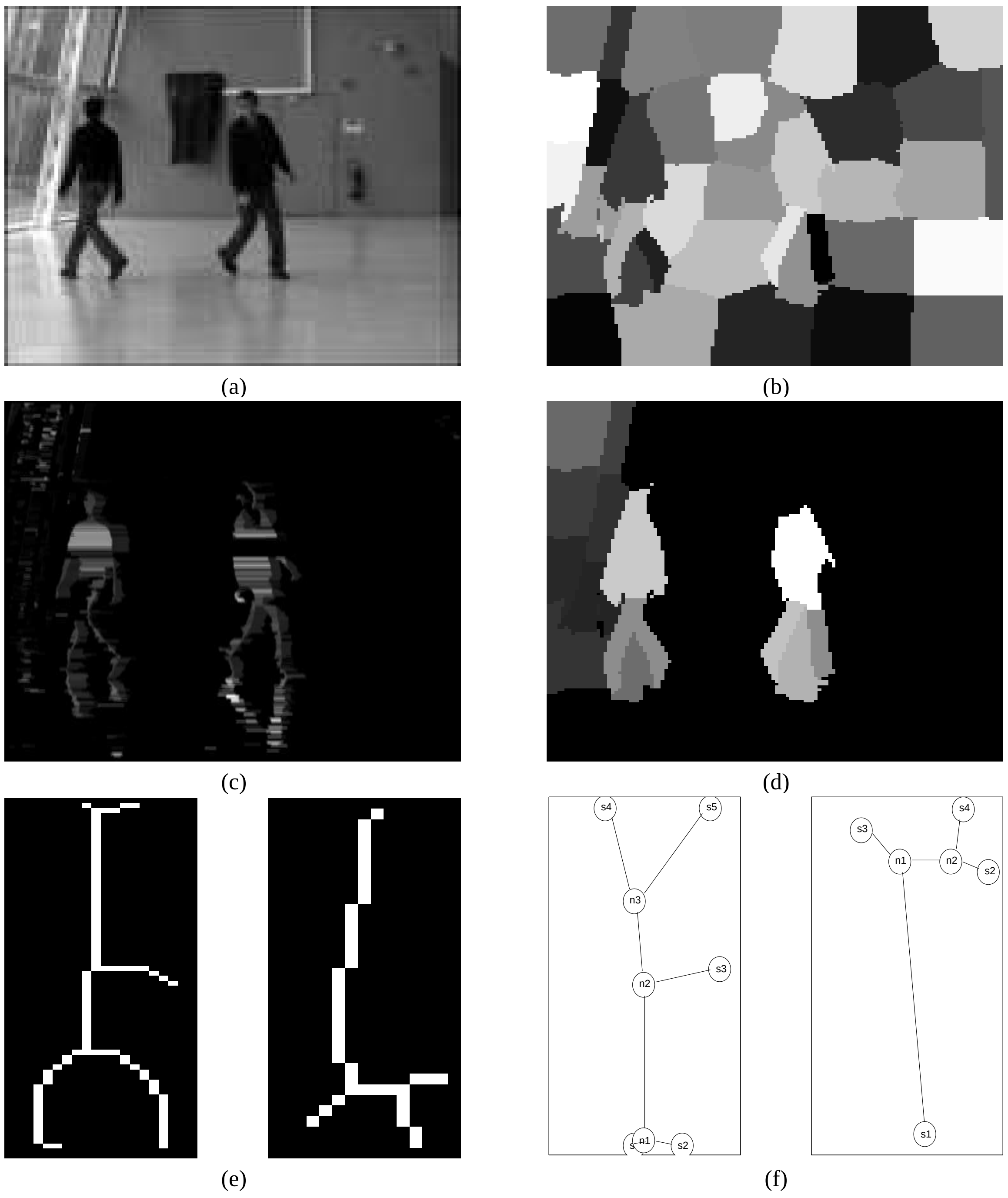

Fig. 3. This figure shows the different steps of the pedestrian extraction from the background and the resulting skeleton and associated graph. (a) Original image. (b) Ncut result. (c) Disparity map. (d) Ncut combined with disparity. (e) 2 examples of skeleton. Left : pedestrian. Right : non pedestrian. (f) Graph of previous shapes. Left : pedestrian, right : non pedestrian.