Abstract—Pedestrians are the most vulnerable urban traffic participants. In order to better protect them in pre-crash scenarios, it is necessary to detect them. Unfortunately, pedestrian detection is very difficult in highly cluttered urban environments, using cameras mounted on a moving vehicle. We present a novel approach to walking pedestrian detection, using a dense stereo vision system. We use multiple features combined into a Bayesian framework to yield a high rate of pedestrian detection. The feature set includes simple features such as width, height, lateral and longitudinal speed. It also includes complex motion features, such as the variance of the motion field caused by the pedestrians’ legs and arms moving during walking and the periodicity of the pedestrians’ walking pattern.

I. INTRODUCTION

Pedestrians are the most vulnerable traffic participants [1]. Therefore, the development of driving assistance system for urban environments must detect potential collision courses with pedestrians and warn the driver or take active measures. Such active measures include preventing the vehicle from starting, emergency breaking or active safety devices activation.

Unfortunately, pedestrian detection is not an easy task, especially in cluttered urban environments [2] and using moving cameras. The shape of pedestrians is highly variable, affected by pose, clothing, handbags, backpacks, walking speed etc. It is also hard to segment out pedestrians from background, because of the moving cameras, the fact that pedestrians may move in groups, parts of their body may be invisible due to occlusions and other parts may be spuriously added.

A. Related Work

The high interest in human recognition systems, both for automotive and surveillance applications, can be seen from the large number of papers published in this domain.


There are a number of works, e.g. [5], [6], [7] which describe motion cues in pedestrian detection. A typical approach for motion based pedestrian detection is to determine if the motion associated with the presumed pedestrian is periodic. However, in cluttered environments, it is usually very hard to distinguish object motion from background motion. Also, due to high car speed, close range and low framerate, it is possible that objects will move many pixels from one frame to the next, thus making local methods for motion detection infeasible. We use a complex approach based on 2-D and 3-D information, in order to correctly segment foreground from background features. We track detected objects, and we compensate their global motion, in order to ease the extraction of local relative motion, associated with the arms and legs of a walking pedestrian.

B. Contributions

This paper presents a novel approach for pedestrian detection, using multiple features. 3-D motion cues, form factors and speeds are combined into a powerful Bayesian classifier. The strength of this approach is that it is able to detect walking pedestrians even if not properly segmented or partially occluded. Also, because the method is rather simple, it does not require the high computational resources that most other approaches need. There is also no need to have special infrared or color cameras, because our method uses only normal, gray-scale cameras.
C. Proposed Architecture

We describe a method for detecting walking pedestrians using multiple features. Our pedestrian detector is integrated into the driving assistance system for urban environments we are currently developing.

To deal with the difficult problem of pedestrian detection we use a hierarchical approach, starting with low level image features and gradually focusing to pedestrian detection. The list of steps performed in order to detect pedestrians is (see Fig. 1):

a) Dense Stereo: We use a stereo camera configuration and a hardware dense stereo reconstruction system, “TYZX” [8] which provides us with the basic features: the gray-scale left and right images and a depth image, associated with the left gray-scale image.

b) Points Classification: The points for which the hardware “TYZX” is able to recover the correct depth are further classified into road points, object points and other points, based on various features. Only object points, i.e. points that could possibly belong to interesting objects in the scene (vehicles, pedestrians, road-side objects) are considered in the following steps.

c) Grouping: Object points are grouped into meaningful cuboid objects (boxes), based on their 3-D coordinates distribution. The result of this step is list of coarse objects, represented by their boxes and characterized by their associated coordinates and dimensions.

d) Simple form features extraction: The height of grouped objects and their base radius are simple, if not powerful features, that can be used for pedestrian detection, especially when rejecting too large or too small objects. These features are taken directly from the output of the grouping module.

e) Tracking: Coarse objects are tracked across multiple frames using a Kalman filter-based multi-tracker. This step ensures stable detection and easier optical field computation (because of the inherent motion compensation resulting from the tracking). All tracked objects are considered as possible pedestrian candidates.

f) Object speed extraction: From the tracker’s output, and knowing the ego vehicle speed, object speeds can be computed. Currently, we use 2 types of speeds for our classifier: one that is parallel to the ego vehicle’s axis, the longitudinal speed, and one that is perpendicular to the first.

g) Depth Masks: Object masks are computed for all tracked objects. Only points for which their 3D coordinates lie inside the tracked cuboid are considered. This step is important as it eliminates spurious background points and deals with partial occlusions.

h) Optical Flow: A pyramidal, corner-based optical flow detection algorithm is used to compute optical flow in all corner points belonging to tracked objects. Only optical flow vectors starting and ending on non-masked points are considered.

i) 3D Velocity: The true 3D velocity of the considered points is computed, using the 2D optical flow, stereo depth and frame time-stamps.

j) PCA: Principal component analysis is used to find the principal direction of the 3D velocity field variation for each individual object. Variance is smoothed across frames, to increase its stability. We call the magnitude of this principal component a “motion signature”. This motion signature is much smaller for non-pedestrians as compared to pedestrians, and is thus a powerful feature for pedestrian detection.

k) Motion History: Using tracking information, we record the motion signature across multiple frames, to determine its history.

l) Motion Spectrum: We compute the spectrum of the motion signature variation in time. Pedestri-
ans display a typical periodic motion signature, while other types of objects display only impulsive noise. We found that the cutoff frequency for the motion spectrum is much smaller for pedestrians as opposed to non-pedestrians. This makes the motion spectrum a powerful feature for pedestrian classification.

m) Bayesian Classification: A naive Bayesian classifier is used to combine the extracted features (height, base radius, lateral and longitudinal speed, motion signature and the motion spectrum cutoff frequency). The apriori pedestrian probability is also an input of the Bayesian classifier. If the probability output of this classifier is more than 0.5 pedestrian, the object is classified as a pedestrian.

D. Paper Structure

The rest of the paper presents the computation of each feature, as well as the Bayesian classifier. The next section presents a number of simple features we use for classification. Section III describes the computation of the motion signature. Section IV describes the computation of the motion periodicity spectrum, and its associated feature, the cutoff frequency. Section V describes the way we combine our features into a naive Bayesian classifier. In section VI we show the experimental results obtained by using our walking pedestrian detector. The section VII concludes our work and shows possible future improvements.

II. SIMPLE FEATURES

A number of easy to compute features are used in our classifier. These features use only the information given by the grouping and tracking modules.

The objects are represented by their bounding 3-D box. We extract the simple features directly from this box. The simple features are: the object’s height, base radius, longitudinal speed (in the direction of the ego vehicle) and lateral speed (the direction perpendicular to the direction of the ego vehicle). Although these simple features are not powerful enough by themselves to detect pedestrians, adding them improves the final detection. Figure 2 shows some plots of these features’ relevance to classification. The objects having a box height of 1.5–2m a box radius of about 0.5m positively influence the overall pedestrian probability, while other values have a negative influence. Also, the probability of pedestrians drops greatly with the increase of both lateral and longitudinal speed. Some small lateral speed seems to increase the probability of pedestrians, because most pedestrians in the training set are crossing the street in front of the ego vehicle. By testing our system with and without using these simple features, we came to the conclusion that, while the influence of these simple features on the overall detection is not very large, they prove useful in some occasions. As these simple features are already computed, using them does not increase the complexity of our algorithm.

III. MOTION SIGNATURE

This section describes the computation of the “motion signature” feature. This feature captures the usual pedestrian walking pattern. Pedestrians are articulated objects, and their different body parts move in different direction during walking. The motion signature feature captures this behavior, by using the variance of the 3D velocity field associated with different points located on the pedestrian’s body. The next subsections will describe
in detail the steps required in order to compute the motion signature.

A. Depth Masking and 2D Optical Flow Extraction

This section describes the 2D optical flow extraction, as a preliminary step to the true 3D velocity field computation. The approach used for this step is the pyramidal Lucas-Kanade like optical flow extraction presented in [9]. We tried various other approaches and concluded that this is the most suited to the considered environment (moving camera, moving objects, urban area).

The following subsections present the difficulties of optical extraction introduced by the urban environment alongside with our solutions for them.

B. Frame Rate

The frame rate is relatively low as compared to the velocities of the objects in the scene. The average frame rate in the sequences we used was 15 fps, which means that an object sufficiently close by could move many pixels from one frame to the next. For example, a pedestrian situated at 3 meters in front of the camera moving with $1.5\,m/s$ on a direction parallel to the camera, would generate an image motion of

$$\Delta X = \frac{f V_x}{Z} \Delta t$$

which equals 27 pixels in our $f = 800\,\text{pixels}$ camera. In order to cope with the relatively low frame rate we use the output from the tracker to estimate the global motion of the object. We extract the object’s image from the previous frame based on its previous location and the object’s image in the current frame based on the location predicted by the tracker. The size of the extracted image is given by the minimum rectangle that encloses the projected 3D cuboid associated with the each object. In order to simplify the optical flow computation we equalize the sizes of the previous and current object image, considering the largest size.

C. Occlusions

Objects passing in front of each other cause occlusions. The problem of occlusions must be addressed, because if we ignored it, a moving object passing in front of a stationary one will cause spurious optical flow components associated with the object in the background. In order to eliminate from objects’ images the pixels that are not associated with true object parts we only consider a “slice” of the image. We compute the minimum and maximum object depth values of the cuboid associated with the considered object, as expressed camera’s coordinate system. We then filter out the points for which the depth estimate computed by the TYZX system lies outside the minimum and maximum depth interval. Having dense stereo information is crucial for this step, as a sparse set of points would not capture the true extend of the object’s shape. Even with a dense set of stereo-reconstructed points, the masked image sometimes contains “holes”, especially if the pedestrian’s clothing texture is uniform. This does not pose big problems, because in areas with uniform texture we would not be able to extract the optical flow anyway. Another problem is that some parts of the pedestrian’s feet are linked with the road. This too does not seem to influence the result of the optical flow computation.

D. Optical flow variability

We tried various methods for computing the optical flow, based on brightness constancy constraints such as those described in [10], [11], [9] and block matching.
We also tried methods based both brightness and depth as described in [12]. Unfortunately when the ego vehicle is non-stationary, the radial optical flow field components generated by the motion of the ego vehicle varies greatly from the center of the image to its edges. Also, because of imperfect tracking, global object motion cannot be totally eliminated. The moving parts of the human body also generate a large optical flow variance when imaged from close range. However, is imperative to compute a correct 2D optical flow field, as our detection scheme relies solely on motion cues.

The methods described in [10] and [11] do not yield good results because they are unable to estimate sufficiently large motion vectors, caused by low frame rate and large motions (ego vehicle and other objects). The method described in [12] doesn’t seem to increase the precision of the optical flow, because the range data used to form the linear depth constancy equation is too smooth (lacking corners or edges) to be useful.

Consequently, only two methods for optical flow computation are useful for our environment: block matching and the pyramidal approach described in [9]. Block matching gives good results, but is prohibitively computational expensive. It is also unable to generate a sufficient number of optical flow vectors, because it uses a fixed size, large window. Therefore, we used a the pyramidal approach described in [9]. This approach has the advantage that it works across a large range of displacements. It also computes the optical flow only where it can be recovered exactly, at image corner points. The number of corner points is relatively small and tracking them is not very computationally intensive. The fact that we track the global motion of each object further increases the working range of this optical flow extraction method, as it only needs to detect local motion displacements. Because we perform corner detection only on the masked object images, and only corner points are tracked, optical flow computation does not have a very high computational expense.

To summarize, the steps we perform for optical flow extraction are (see Fig. 3):

1) Tracked object’s image extraction
2) Depth mask computation, which eliminates wrong points from the object’s image
3) Pyramidal optical flow computation:
   a) Corner detection (based on eigenvalues)
   b) Gaussian pyramid generation
   c) Pyramidal optical flow computation
4) Elimination of optical flow vectors who’s ends do not fall into points with correct depth

The optical flow vectors computed in this step are not directly usable as features, because their length varies with depth. A small motion at close range will produce a large optical flow vector, while a large motion at distant range would produce only a small displacement. Also, not all components of the true 3D motion are relevant for our detector. The next section discusses the recovery of the true 3D motions.

E. 3D Motion Field

In this section we discuss the computation of the true 3D motion of objects in the scene, based on the 2D optical flow and the range image. As explained in the previous section, we only compute 2D optical flow vectors which start and end in points for which the hardware TYZX system is able to supply the range value. Let \( p_1(x_1, y_1) \) denote the start of the optical flow vector \( \vec{v} \) in the previous frame and \( p_2(x_2, y_2) \) denote the end of the optical flow vector (in the current frame). Also let \( z_1 \) and \( z_2 \) be the depth values supplied by the TYZX system at points \( p_1 \) and \( p_2 \) respectively. Then, the 3D velocity vector (expressed in the left camera’s coordinate system) associated with the 2D optical flow vector \( \vec{v} \) is:

\[
\vec{V} = \begin{pmatrix} \frac{(x_2 - x_1)z_1}{f_x} \\ \frac{(y_2 - y_1)z_1}{f_y} \\ z_2 - z_1 \end{pmatrix},
\]

where \( f_x \) and \( f_y \) are the focal lengths of the left camera, expressed in horizontal and vertical pixel units respectively.

Because the camera’s position is arbitrary, we would like to express the 3D motion vector into the more suitable world coordinate system. We have:

\[
\vec{V}_w = R^T (\vec{V} - \vec{T}),
\]

where \( \vec{T} \) and \( R \) are the translation vector and the rotation matrix from the world to the left camera’s coordinate system, as determined by the calibration process.

After performing these transformations we end up with a set of \( \vec{V}_w \) vectors for each tracked object. The first step is the elimination of objects who lack a sufficiently high number of motion vectors. Our experiments determined that objects for which there are less than 5 motion vectors are very unlikely to be walking pedestrians. These objects are manly poles ore other stationary objects. Therefore, in the next steps, we will only consider objects for which more than 5 motion vectors have been computed.
The 3D motion vectors cannot, by themselves, serve as a discriminating feature between pedestrians and other objects. Because of imperfect tracking, objects tend to still have a global motion, even after the global displacement predicted by tracking is eliminated. We solve this problem by subtracting the average motion
\[ \mu_V = \frac{\sum_{i=1}^{n} V^W_i}{n} \]  
from each motion vector. Another problem is caused by objects such as vertical poles. Because they lack horizontal edges (high frequency components along the vertical direction), such objects may present a spurious vertical motion component. We tried various approaches to eliminate such spurious motion vectors. The approaches we tried include:
1) Considering the ratio of horizontal to vertical motion components.
2) Considering the average angle between the horizontal plane and the motion vector.
3) Considering only the horizontal components of the motion vectors.

Although all the above approaches yield a better result (higher discriminating power) than considering only the modules of the motion vectors, they are all rather sensitive to noise. A much better and stable approach, principal component analysis, is described in the next section.

**F. Principal Components Analysis**

While experimenting with the different features extracted from the 3D motion field, presented in the previous section, we reached the conclusion that both the magnitudes and the orientation of the motion field are important features for our walking pedestrian detector. We would like to find the main direction along which most motion takes place. Also, we are not interested in the motion itself, but rather in its variability. For example, while walking, a foot moves forward while the other moves backward (relative to the global body motion), and also the arms tend to have the same motion pattern (if not carrying large bags). A measure of the motion variance can be obtained by performing a principal component analysis. The covariance of the motion vectors is:
\[ C = \frac{1}{n} \sum_{i=1}^{n} (V^w_i - \mu_V)(V^w_i - \mu_V)^T \]  
The 3 by 3 matrix \( C \) represents a covariance matrix. Its eigenvalues are real and positive. Let \( \lambda_{max} \) be the largest eigenvalue of matrix \( C \). The eigenvector \( \vec{V}_{\max} \) associated with \( \lambda_{max} \) represents the direction of the principal variance of the vector field \( \vec{V} \). The standard deviation along the direction \( \vec{V}_{\max} \) is \( \sqrt{\lambda_{max}} \). Pedestrians move mainly in the horizontal \( xOz \) plane. Therefore, we eliminate the vertical (\( y \)) motion components, and consider only the projection of \( \vec{V}_{\max} \) on the \( xOz \) plane. We compute the new variance as:
\[ \lambda_{xz} = \frac{\lambda_{max} ||V_{xz}||}{||\vec{V}_{max}||} \]  
As the experimental results will show, the value of the new standard deviation \( \sigma = \sqrt{\lambda_{xz}} \) is a good feature for discriminating walking pedestrians from other objects.

However, because we did not use time stamps for the frames, \( \lambda_{xz} \) varies a lot from one frame to the next. To cope with this problem, we use a simple smoothing process:
\[ \lambda_{xz}^t = (\alpha - 1)\lambda_{xz} + \alpha \lambda_{xz}^{t-1}. \]  
We tested different values of \( \alpha \). As \( \alpha \) grows, the value of \( \lambda_{xz}^t \) depends more and more on the value of \( \lambda_{xz}^{t-1} \) and less on the value of the measured \( \lambda_{xz} \). We found that a value of \( \alpha = 0.5 \) is suitable, because it smooths out spurious values of the measured \( \lambda_{xz} \) but it is also able to adapt quickly enough to a series of new, correct measurements.

**IV. Motion Signature Periodicity**

Another powerful feature, related to the motion signature, is its periodicity. Although non-pedestrian objects display occasionally some spike noise in their motion signature, this noise is not periodic.
probability that object \( C \) is present is:

\[
P(F_1) \quad \text{probability that the feature } F_1 \text{ appears, regardless of the object’s class. This is actually a normalization constant, and we will show that it is not actually needed for classification, as it can be eliminated. But why is such a complicated formula needed? Why not just statistically determine } P(C_1|F_1) \text{? There are multiple reasons why this approach is not optimal. Firstly, should the probability of class } C_1 \text{ change (as for example the probability of pedestrians encounters in urban as opposed to non-urban scenarios), } P(C_1|F_1) \text{ will also change. However, by using the Bayes’ formula, we need to change only } P(C_1) \text{ (which is much easier). Secondly, should we need multiple features, it is next to impossible to determine the probability of } C_1 \text{ conditioned by the appearance of all possible feature combinations. A simplification of this the Bayes’s formula is to use likelihood instead of probability. The likelihood of a given class is defined as:}
\]

\[
L(C_1|F_1) = \frac{P(C_1|F_1)}{P(\neg C_1|F_1)}
\]

Should \( L(C_1|F_1) \) be greater than 1, the object is classified as belonging to \( C_1 \). If the likelihood is smaller than 1, the object is classified as belonging to \( \neg C_1 \). If there are multiple classes, the object is classified as belonging to the class having the highest likelihood.

Another useful measure is the logarithmic likelihood or log likelihood:

\[
\ln(L(C_1|F_1)) = \ln\left( \frac{P(C_1|F_1)}{P(\neg C_1|F_1)} \right) = \ln(P(C_1|F_1)) - \ln(P(\neg C_1|F_1))
\]

By combining Bayes’ formula and the log likelihood we obtain:

\[
\ln(L(C_1|F_1)) = \ln(P(C_1)P(F_1|C_1)) - \ln(P(C_1)P(F_1|C_1))
\]

The usefulness of log likelihoods is that they can be summed over, without requiring complicated multiplications or divisions that could result in loss of precision.

The next issue we shall discuss here is how to classify using multiple features. The probability of an object belonging to class \( C_1 \) when features \( F_1, F_2, \ldots, F_n \) are known to be present is given by the Bayes’ formula:

\[
P(C_1|F_1, F_2, \ldots, F_n) = \frac{P(C_1)P(F_1, F_2, \ldots, F_n|C_1)}{P(F_1, F_2, \ldots, F_n)}
\]
because it requires far too many parameters. A very useful simplification is to assume that features are independent:

\[ P(F_i|F_j) = \delta_{ij} \]  

(13)

Using the independence assumption, we obtain:

\[ P(C_1|F_1, \ldots, F_n) = \frac{P(C_1)P(F_1|C_1)\ldots P(F_n|C_1)}{P(F_1)\ldots P(F_n)} \]  

(14)

This last formulation of the conditional probability is very useful, as all features are treated independently. The likelihood of class \( C_1 \) is:

\[ L(C_1|F_1, \ldots, F_n) = P(C_1)P(F_1|C_1)\ldots P(F_n|C_1) \]  

\[ \times P(\neg C_1)P(F_1|\neg C_1)\ldots P(F_n|\neg C_1) \]  

(15)

Taking the logarithm here also helps, and yields:

\[ \ln(L(C_1|F_1, \ldots, F_n)) = A + B_1 + \ldots + B_n \]  

(16)

where:

\[ A = \ln \frac{P(C_1)}{P(\neg C_1)} \]  

(17)

and

\[ B_i = \ln \frac{P(F_i|C_1)}{P(F_i|\neg C_1)} \]  

(18)

If the log likelihood is greater than 0, the object is classified as belonging to class \( C_1 \), otherwise it is classified as belonging to \( \neg C_1 \). The expression using log likelihoods is easier to compute, because combining different probabilities involves summation as opposed to multiplication.

In order to use a Bayesian approach towards pedestrian detection, we had to develop a framework for learning the conditional probability distributions of the features involved. The framework was designed to be sufficiently general to accommodate any feature desired for pedestrian classification, and to be easy to use. A large set of objects, containing pedestrians and other types of objects present in urban traffic scenarios was manually labeled, and their associated features logged to a file. Then, a Python script automatically computes the probabilities and generates the C source code for a naive Bayesian classifier. The fact that the script directly generates the classifier makes it very useful and allows fast experimenting with different training sets and features.

### Table I

**Experimental Results**

<table>
<thead>
<tr>
<th>Category</th>
<th>Number</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects</td>
<td>600</td>
<td>100%</td>
</tr>
<tr>
<td>Pedestrians</td>
<td>243</td>
<td>40.5%</td>
</tr>
<tr>
<td>Other Objects</td>
<td>357</td>
<td>59.5%</td>
</tr>
<tr>
<td>False Positives</td>
<td>49</td>
<td>8.1%</td>
</tr>
<tr>
<td>False Negatives</td>
<td>39</td>
<td>6.5%</td>
</tr>
<tr>
<td>Correct Detection</td>
<td>512</td>
<td>85.3%</td>
</tr>
</tbody>
</table>

Fig. 6. Images from our test sequences. The red colored boxes represent tracked objects classified as pedestrians while green colored objects are tracked objects classified as non-pedestrians.

**VI. Experimental Results**

In this section we present the results we obtained by using our walking pedestrians classifier. For testing purposes we used a set of 600 test object images. This set of images was independent from the images we used for determining the threshold for our classifier. The results are summarized in Table I.

Fig. 6 shows one frame for each of the 4 independent sequences we used for our testing purposes. In two of the sequences, a) and c), the ego vehicle was in motion, while in b) and d) it was stopped. Fig. 7 shows a number of images correctly identified as pedestrians. As one can see in fig. 7 even bicyclists are sometimes detected as pedestrians, because of the motion of the legs (we considered this detection to be useful, because bicyclists are also vulnerable traffic participants). Fig. 8 shows a
number of correctly detected non-pedestrians. Most of incorrect detection occurs when the object is entering or exiting the scene, because the object is only partially visible.

Although we did not, as yet, take any steps toward optimizing our system for speed, we found that our detector has no problems running in real time (15 fps). The motion computation and classification adds only a small overhead to the whole driving assistance solution we are currently developing.

VII. CONCLUSIONS AND FUTURE WORK

We managed to develop a walking pedestrian detector, based on multiple features. These features include the object’s height, base radius, lateral and longitudinal speed, motion signature and motion signature periodicity. Our method has a high rate of correct detection and is less sensitive to incorrect grouping and partial occlusions than methods based solely on pedestrian shape. Because our classification is based on a 3D motion field, we can detect pedestrians walking in various directions and at various distances. Our system achieves a high rate of correct detection, above 85%. Because the optical flow computation (for the motion signature) is restricted to only a few hundred points, our detector is fast and runs in real-time.

There are a number of possible improvements for our detector:

1) Considering the occluded area of each object and normalizing the magnitude of optical flow variation with the size of non-occluded object area.

2) Optimizations of the corner detection, Gaussian pyramids generation and corner tracking, using SIMD instructions, in order to make the detector as fast as possible.

3) Perhaps the most obvious improvement will be using other features, like shape and texture. However, we need to find a way to improve the object grouping and tracking, to make these other feature usable.

4) Also, using a Bayesian network instead of a naive Bayesian classifier could help, by capturing the dependencies between individual features.
REFERENCES


