Abstract—The choice for image descriptor in a visual navigation system is not straightforward. Descriptors must be distinctive enough to allow for correct localization while still offering low matching complexity and short descriptor size for real-time applications. MPEG Compact Descriptor for Visual Search is a low complexity image descriptor that offers several levels of compromises between descriptor distinctiveness and size. In this work we describe how these trade-offs can be used for efficient loop-detection in a typical indoor environment.

I. INTRODUCTION

Robots that navigate through unknown environments, such as autonomous vacuum cleaners, all face a common challenge: to create a representation of their environment while simultaneously trying to locate themselves. This problem is known in literature as Simultaneous Localization and Mapping (SLAM) and its formulation has been thoroughly reviewed in [1] and [2]. Approaches to SLAM usually involve two alternate phases. During Motion Prediction the robot uses internal parameters to estimate local displacements, while during Measurement Update the robot interacts with the environment to improve both its pose estimation as well as its estimate of its environment map.

In vision-based SLAM, motion prediction can be obtained by extracting and matching visual features from a sequence of images in a process called feature-based visual odometry [3]. These same features can also be used as landmarks for loop-detection during the measurement update phase in a complete Visual SLAM (VSLAM) system [4].

The purpose of Loop-detection is to identify landmarks on the map that have already been seen by the robot during early stages of navigation. During this process, if a particular landmark is allegedly found in two different places it means that the estimated trajectory described by the robot has probably drifted at some point. The relative displacement between these two appearances can then be used by a global optimizer to improve estimations of both the landmarks’ positions as well as robot’s pose.

Early approaches to loop-detection using visual features include the work in [5], where authors used the SIFT [6] for its distinctive power and thus capability of correctly find a loop. SIFT’s distinctiveness, however, comes at a high price in terms of compute complexity leading to substantial battery consumption. Moreover, the amount of SIFT features generated by a single image also makes it prohibitively expensive in terms of bandwidth requirements where remote processing is needed such as collaborative mapping scenarios. To speed-up loop detection, authors in [7] have used an intermediate level of representation known as bags of visual words [8].

Finally as the robot navigates throughout its environment the number of observed landmarks increases and so does the number of descriptors it stores for loop-detection. This means that loop-detection algorithms are bound to become expensive in terms of both memory and computational complexity [2] as the map grows. This forces system designers to either choose to use less complex descriptors, risking wrong data association, or to overestimate memory demands during hardware requirement analysis.

The problem of finding a perfect balance between descriptor distinctiveness and descriptor size is not exclusive to the VSLAM domain. When dealing with large databases, Content-Based Image Retrieval (CBIR) systems face this very same issue. Very recently, the Moving Picture Experts Group (MPEG) has defined a new industry standard for CBIR known as Compact Descriptors for Visual Search (MPEG CDVS) [9]. The standard specifies various modes of compression that offer trade-offs between descriptor distinctiveness and size and also suggests specific metrics to quantify similarity between images.

In this work we propose to employ MPEG CDVS for robot navigation, showing that the characteristics that make CDVS a good compact descriptor are also ideal for this new application. More specifically, we show that MPEG CDVS can be used as a fast, reliable and storage-efficient loop detector in a typical indoor VSLAM application.

Our first contribution comes in Section III where we describe a probabilistic approach to loop detection using the standard’s suggested similarity metric. We then compare the performance of CDVS compression modes in terms of matching speed, feature extraction and storage requirements with the well-known SIFT descriptor for five different types of indoor floors and show that CDVS has superior performance in all cases in Section IV. Finally, in Section V we apply our proposed method to a real robotic application and show that our VSLAM approach gives better results than state-of-the-art laser-based SLAM.
II. THE MPEG CDVS STANDARD

A. Descriptor Generation

A CDVS descriptor is made of two parts: one global descriptor associated to the entire image and a set of local descriptors associated to specific points in the image known as interest points. The entire descriptor extraction process can be summarized as follows:

1) Interest point detection;
2) Feature selection and descriptor extraction: according to the level of compression used only a limited number of interest points will account for the final descriptor;
3) Local descriptor extraction;
4) Local descriptor compression;
5) Coordinate coding;
6) Global descriptor generation: It is an aggregation of local descriptors to generate a fixed, small size description of the entire image;

The final result of CDVS extraction is a compressed file whose size is upper-bounded by 512B, 1kB, 2kB, 4kB, 8kB, and 16kB for extraction modes ranging from 1 to 6 respectively.

B. Descriptor Matching and Score

When comparing two images MPEG CDVS suggests the use of two different types of matching scores: global score and a local score.

The global score is given as a weighted correlation between global descriptors from the two images.

The local score on the other hand results from the sum of local scores of each descriptor in those images, i.e a one-to-one comparison is made between local descriptors from both images. Finally, the standard also suggests the use of a geometric consistency analysis, known as Distrat [10], to eliminate false matches between descriptors based on their geometry disposition.

In this work, in order to detect a loop as defined in Section III-A, we consider only features that have passed the geometric consistency test. Moreover, we consider the values given by local score as our means to indirectly measure the probability of loop detection for it gives more reliable results.

III. PROPOSED MOTION MODEL AND LOOP DETECTION

A robot carrying a calibrated camera navigates through an indoor environment while taking a picture $I_k$ of the floor below at each time step $k$. The robot’s starting position and heading define both origin and $x$-axis of a global coordinate frame. This coordinate system then becomes uniquely defined as we choose the $z$-axis to point upwards.

We assume the environment’s floor to be a planar surface so that, for each time step $k > 0$, the robot’s pose is given by $x_k = [x_k, y_k, \theta_k]^T$, where $x_k$ and $y_k$ indicate the robot’s coordinates and $\theta_k$ is the robot’s heading.

Final motion between time steps $k-1$ and $k$ can be modeled as a rotation followed by translation, so that at $t = k$ pose can be recursively obtained as

$$[x_k, y_k]^T = R(\Delta \theta_{k-1,k})[x_{k-1}, y_{k-1}]^T + T_{k-1,k}$$

$$\theta_k = \theta_{k-1} + \Delta \theta_{k-1,k}$$

where $\Delta \theta_{k-1,k}$ is the rotation angle estimated between time steps $k-1$ and $k$, $R(\Delta \theta_{k-1,k})$ is the rotation matrix for that same angle, and $T_{k-1,k}$ is the translation vector.

A. Loop Definition

The use of a downward-facing camera allows for a natural definition of loop based on the intersection of imaged regions. For images $I_a$ and $I_b$ taken along the robot’s path, we define loop as a function of the overlap ratio between the floor areas observed by these two images. So given the area of intersection $\{I_a \cap I_b\}$, and the respective area of union $\{I_a \cup I_b\}$, a loop can be defined as

$$J = \frac{\text{area}(I_a \cap I_b)}{\text{area}(I_a \cup I_b)}$$

where $r$ is the threshold that defines the minimum overlap ratio for which two intersecting images can be considered a loop. In this work we set this threshold to $r = 0.33$, which roughly amounts for an area intersection of 50% when $I_a$ and $I_b$ have the same areas.

B. Loop Probability

Loop detection as defined in (4) requires the knowledge of how much intersection there is between the two images. In order to indirectly measure the probability of having a particular area ratio we use the local score given between two images so that

$$P(\text{loop} = 1 | \text{score} = s) = P(J \geq r | \text{score} = s)$$

$$P(J \geq r | \text{score} = s) = \frac{P(J \geq r, \text{score} = s)}{P(\text{score} = s)}$$

The conditional probability in (5) can be experimentally estimated through (6) by combining the knowledge of the camera’s parameters with a source of relative motion estimation. This process will be described in depth during the next section.

IV. TRAINING OF PROPOSED MODEL

Besides being distinctive, a descriptor also needs to be economical in terms of storage and extraction and matching times in order for it to be considered as a feasible option for loop detection.

In this section we analyze the distinctiveness of all CDVS’ compression modes for the five different types of floorings seen in Fig. 1. We also compare their memory and processing time requirements with a popular implementation of the SIFT descriptor found in [11].
A. Distinctiveness of CDVS local score

Our analysis starts by driving the robot forward for 10 meter using a PointGrey Grasshopper 3 camera rigidly mounted on a Turtlebot 2 in a setup defined in Section III.

For each floor type we obtain a sequence of images and extract their respective CDVS descriptors. We then match each image with all the previous ones in that same sequence using CDVS local score to measure similarity. We repeat this process for all modes of compression to evaluate its effect on feature distinctiveness.

Distinctiveness in this context means to have high local score for pair of images with much overlapping regions and very low local score otherwise. Since images were taken in a sequence during robotic motion, images that are close in the sequence are also spatially next to each other, and thus should have high local score.

A visual representation of these matches using compression mode 6 is given in Fig. 2 where pixel intensities in position \((i,j)\) represent the local score between current image \(i\) and a previously visited image \(j\). Since we only match current images with previous ones, each matrix representing the matches is triangular.

To allow for a fair visual comparison, the matrices values have been normalized. Yellow pixels mean high local score while dark blue pixels indicate a low score. The presence of small, bright triangles seen at the lower end of each matrix indicates when the robot had stopped.

Ideally, these matching matrices should display increasingly intensity of pixel values (yellow) in regions near each diagonal and very low values (dark blue) everywhere else. The natural randomness intrinsically associated to the production of most of flooring types enables them to have a relatively thick principal diagonals and to display very low matching scores where no overlap occurs.

The one noticeable exception occurs for the printed wood floor. This particular artificial type of flooring is made of a printed repetitive patterns. The effect of such patterns appears as bright spots on its matching matrix and can be particularly harmful for loop-detection since it leads to erroneously detected loops.

We can observe the evolution of these spots and the diagonal thickness in Fig. 3 as we vary the compression mode.

Fig. 1: Different types of floorings commonly found in indoor environments.

Fig. 2: Visual representation of Local Score for different floor types.

Fig. 3: Visual representation of Local Score for the Printed Wood floor using different compression modes.

It is clear that the diagonal thickness decreases as the compression level increases, i.e. for lower modes of compression. This phenomenon happens to all flooring types and it is due to the fact that CDVS will use fewer keypoints with shorter local descriptors to represent each image. This makes it difficult to correctly match images that are even just slightly displaced with respect to one another. Therefore; as expected, lower modes of compression can be considered to offer less distinctive local descriptors.

On the other hand and for the same reason, bright spots on the wooden pattern become even more visible as the level of compression increases, which makes this particular kind of flooring the worst case scenario and also our study case to test CDVS for loop detection.

B. Effects of Feature Selection

Besides being able to correctly distinguish between different floor patches, CDVS must also be economical in terms of storage, extraction time and matching time if it is to be considered as an alternative to well-established descriptors such as SIFT [6]. Here we examine these characteristics by analyzing the same five types of flooring.
As seen in Fig. 4, feature selection has the effect of reducing the number of local features generated for each image. Since the final binary size of a CDVS descriptor is limited by its compression mode, the maximum number of local descriptors produced by each mode is upper-bounded and does not significantly depend on the particular type of flooring.

In terms of memory efficiency, feature selection has a clear effect on reducing storage requirements. For example, an image taken from a Mosaic floor would normally require over 300kB of memory if SIFT descriptor were to be used considering implementations such as [11], while CDVS would require at most 16kB at its least compressed mode.

Another positive effect of feature selection is the reduction of extraction time as reported in Table I. Since feature selection is made based on keypoints' characteristics, only features from a few selected keypoints will be processed. Moreover, having a limited number of descriptors per image will also limit the time spent for comparing two images as reported in Table II.

Finally we observe that both extraction and matching times are at least an order of magnitude lower than SIFT and that these values show little variation within a given compression mode.

Having upper-bounded memory requirements and extraction and matching times that are relatively invariant to the different types of floorings are essential qualities for systems that may work on different environments. For example, system requirements for automatic vacuum cleaner should not depend on consumer’s specific type of floor.

C. Estimating Loop Probability

A camera’s intrinsic and extrinsic parameters define the camera’s pose with respect to the world and also allow us to make real world measurements directly from images. These parameters can also be used to circumscribe observed regions by projecting the camera’s field-of-view onto the imaged floor. Once the projected areas of images \( I_a \) and \( I_b \) are known, it is sufficient to know their relative positions to estimate their area of intersection and thus to be able to evaluate the overlap ratio \( J \).

Relative motion during training was obtained using the robot’s odometry, and although odometry suffers from error accumulation after long trajectories, it does provide dependable relative motion estimations over short range distances. Moreover, images that are relatively distant from each other, will have zero overlapping region and therefore error accumulation will constitute a problem. During training phase relative motion was obtained by using a Kalman filter that combined information from both wheel odometry and a robot’s internal gyroscope during the experiment described at the beginning of this section.

By combining these pieces of information with the local scores of each analyzed matching pair, we can generate for each compression mode a loop detection probability curve as defined in Eq. 6. The resulting curves as seen in Fig. 5 show the probability two images having more than 50% of intersection for each mode given a local score.

Lower compression modes achieve certainty at lower values of local score. This is due to the fact that low compression modes also have fewer descriptors to be used during match.

From these curves we select the minimum values...
of local score $s$ that guarantee loop detection for each compression mode. These hypothesis values are reported in Table III and used to define the loops during the final experiments discussed in Section V.

### D. Visual Odometry for Testing

In order to demonstrate that our approach could be applied to a vision-only navigation system having no other sensors such as gyroscope or wheel encoder, we have decided to also implement VSLAM using visual odometry. Our robot setup follows the one in [12]. However, although we do use a similar approach to obtain odometry, our main concern in this work is the correct detection of loops for VSLAM.

Depending on system requirements, less complex feature descriptors such as Brisk [13] or optical flow [14] could be used to generate odometry, while CDVS would be used just for loop detection. However, since local features from each image will already be available, we choose to use CDVS local descriptor to generate visual odometry as well.

For each pair of consecutive images $I_{k-1}$ and $I_k$ we perform a feature extraction and match of MPEG CDVS descriptors, which results into two sets of $N > 2$ matching coordinate pairs. We combine these pixel coordinates with the camera’s calibration information and produce the sets $P_{k-1}$ and $P_k$ each containing the 3D coordinates for the $N$ matching pairs. By defining $\tilde{P}_{k-1}$ and $\tilde{P}_k$ to be the centroids of $P_{k-1}$ and $P_k$ respectively, we retrieve rotation and translation using Singular Vector Decomposition.

A visual representation of this process is shown in Fig. 6.

![Fig. 6: Matching between images at time $k$ and $k + 1$. Keypoints are indicated as dots. Yellow dots represent non-matching features. Green lines represent correspondences between matches. Red lines are false matches found by Distrat.](image)

Although CDVS already performs geometric consistency validation, we make use of a few RANSAC [15] cycles to remove possible remaining outliers and improve results.

### V. EXPERIMENTAL RESULTS

Partial results from Section IV have led us to try our loop-detection technique on the most challenging flooring for loop-closure, i.e. the flooring most susceptible to false-loop detection.

In this experiment, the robot navigates through indoor office for about 110 meter while taking a total of 7154 images of its printed wood floor and performing loops before finally going back to its original position.

<table>
<thead>
<tr>
<th>local score</th>
<th>mode 1</th>
<th>mode 2</th>
<th>mode 3</th>
<th>mode 4</th>
<th>mode 5</th>
<th>mode 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypothesis</td>
<td>10</td>
<td>14</td>
<td>15</td>
<td>18</td>
<td>23</td>
<td>25</td>
</tr>
<tr>
<td>Experimental</td>
<td>–</td>
<td>20</td>
<td>16</td>
<td>18</td>
<td>24</td>
<td>27</td>
</tr>
</tbody>
</table>

**TABLE III:** Hypothesized and Experimental threshold values for local score loop detection.

We first use the sequence of images to generate the path’s visual odometry as described in Section IV for all except the first compression mode, which was unable to generate enough matching points between consecutive images. For those modes capable of estimating translation and rotation from consecutive images, we report their respective paths in Fig. 7 where we use the room’s blueprint as reference map.

![Fig. 7: Path comparison using visual odometry.](image)

We then perform loop detection as described in Section IV where for each image pair whose local score was above the hypothesized value in Table III a loop was declared.

For each compression mode, we have represented data from visual odometry and loop constraints as a path graph so that the robot’s trajectory could be optimized using the LAGO graph optimization software [16], whose purpose is to find a coherent sequence of poses that better describe all loop and odometry constraints, and thus perform VSLAM.

During these experiments, we have observed that the proposed local scores thresholds loop-detection found earlier were slightly too permissive and still allowed for small amount of false-positive loops to be detected. This fact has led us to empirically increase these thresholds until reasonable results were obtained. We report these new values as the *Experimental* entries in Table III, which differ very little from the hypothesized ones and thus proving that the method is still valid. The resulting trajectories for each compression mode using the experimental thresholds can be seen in Fig. 8.

A visual inspection between the two figures reveals the improvements obtained for all compression modes when loops are correctly detected. Except for compression mode 2, all improved trajectories pass through the hallway, enter and exit the northwest room and respect the physical constraints present in the map. However, in order to have a more quantitative measure of such improvements we report in Table III the pose difference between starting and ending poses in the trajectory, which ideally should be none.

To highlight the gains in terms of both storage savings and matching times with respect to SIFT, we have compared the
Table IV: Relative pose errors between staring and final position.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Δx (m)</th>
<th>Δy (m)</th>
<th>Δθ (rad)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode 2</td>
<td>17.35</td>
<td>-6.58</td>
<td>-0.86</td>
</tr>
<tr>
<td>Mode 3</td>
<td>-4.36</td>
<td>1.27</td>
<td>0.03</td>
</tr>
<tr>
<td>Mode 4</td>
<td>0.22</td>
<td>0.19</td>
<td>-0.13</td>
</tr>
<tr>
<td>Mode 5</td>
<td>1.01</td>
<td>0.09</td>
<td>-0.17</td>
</tr>
<tr>
<td>Mode 6</td>
<td>2.10</td>
<td>0.00</td>
<td>-0.23</td>
</tr>
</tbody>
</table>

Table V: Storage requirement for all 7154 images and total matching time between last sequence image and all previous ones.

<table>
<thead>
<tr>
<th>property</th>
<th>mode 2</th>
<th>mode 3</th>
<th>mode 4</th>
<th>mode 5</th>
<th>mode 6</th>
<th>SIFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage (MB)</td>
<td>7.67</td>
<td>14.63</td>
<td>28.59</td>
<td>56.55</td>
<td>112.43</td>
<td>1213.84</td>
</tr>
<tr>
<td>Time (s)</td>
<td>4.23</td>
<td>6.62</td>
<td>9.62</td>
<td>27.27</td>
<td>58.32</td>
<td>1264.20</td>
</tr>
</tbody>
</table>

Fig. 9: Map and path generated using a laser scanner with Gmapping algorithm.

At first sight, results from laser scanner can be considered incorrect and unreliable. This occurs because laser scanner was unable to create a precise map of environment and thus was unable to reproduce its path correctly on the real world map. This becomes evident in Fig. 9 where the path generated by the laser seems to be coherent to its self-generated "bent" map. Our method clearly does not suffer from the same issue.

VI. Conclusion

In this work we have proposed the use of MPEG CDVS in a SLAM framework for loop-detection in an indoor environment.

We have shown experimentally that CDVS’ feature selection serves not only to reduce the final descriptor size but also to significantly speed up feature extraction and matching. In our practical experiment CDVS’s least compressed mode was shown to be over 20 times faster than SIFT during matching time and to require 10 times less storage space and still able to provide for correct loop-detection.

Finally, when we compared our approach to a laser scanner, we have seen that CDVS has generated far better results.

References