Empty Cities: a Dynamic-Object-Invariant Space for Visual SLAM
Berta Bescos, Cesar Cadena, Jose Neira

Abstract—In this paper we present a data-driven approach to obtain the static image of a scene, eliminating dynamic objects that might have been present at the time of traversing the scene with a camera. The general objective is to improve vision-based localization and mapping tasks in dynamic environments, where the presence (or absence) of different dynamic objects in different moments makes these tasks less robust. We introduce an end-to-end deep learning framework to turn images of an urban environment that include dynamic content, such as vehicles or pedestrians, into realistic static frames suitable for localization and mapping. This objective faces two main challenges: detecting the dynamic objects, and inpainting the static occluded background. The first challenge is addressed by the use of a convolutional network that learns a multi-class semantic segmentation of the image. The second challenge is approached with a generative adversarial model that, taking as input the original dynamic image and the computed dynamic/static binary mask, is capable of generating the final static image. This framework makes use of two new losses, one based on image steganalysis techniques, useful to improve the inpainting quality, and another one based on ORB features, designed to enhance feature matching between real and hallucinated image regions.

To validate our approach, we perform an extensive evaluation on different tasks that are affected by dynamic entities, i.e., visual odometry, place recognition and multi-view stereo, with the hallucinated images. Code has been made available on https://github.com/bertabescos/EmptyCities_SLAM.

I. INTRODUCTION
Most vision-based localization systems are conceived to work in static environments [1], [2], [3]. They can deal with small fractions of dynamic content, but tend to compute dynamic objects motion as camera ego-motion. Thus, their performance is compromised. Building stable maps is also of key importance for long-term autonomy. Mapping dynamic objects prevents vision-based robotic systems from recognizing already visited places and reusing pre-computed maps.

To deal with dynamic objects, some approaches include in their model the behavior of the observed dynamic content [4], [5]. Such strategy is needed when the majority of the observed scene is not rigid. However, when scenes are mainly rigid, as in Fig. 1a, the standard strategy consists of detecting the dynamic objects within the images and not to use them for localization and mapping [6], [7], [8], [9]. To address mainly rigid scenes, we propose to instead modify these images so that dynamic content is eliminated and the scene is converted realistically into static. We consider that the combination of experience and context allows to hallucinate, i.e., inpaint, a geometrically and semantically consistent appearance of the rigid and static structure behind dynamic objects. This hallucinated structure can be used by the SLAM system to provide it with robustness against dynamic objects.

Turning images that contain dynamic objects into realistic static frames reveals several challenges:

1) Detecting such dynamic content in the image. By this, we mean to detect not only those objects that are known to move such as vehicles, people and animals, but also the shadows and reflections that they might generate, since they also change the image appearance.

2) Inpainting the resulting space left by the detected dynamic content with plausible imagery. The resulting image would succeed in being realistic if the inpainted areas are both semantically and geometrically consistent with the static content of the image.

The first challenge can be addressed with geometrical approaches if an image sequence is available. This procedure usually consists in studying the optical flow consistency along the images [8], [7]. In the case in which only one frame is available, deep learning is the approach that excels at this task by the use of Convolutional Neural Networks (CNNs) [10], [11]. These frameworks are trained with the previous knowledge of what classes are dynamic and which ones are not. Recent works show that it is possible to acquire this knowledge in a self-supervised way [12], [13].

Regarding the second challenge, some recent image inpainting approaches use image statistics of the remaining image to fill in the holes [14], [15]. The former work estimates
the pixel value with the normalized weighted sum of all the known pixels in the neighbourhood. While this approach generally produces smooth results, it is limited by the available image statistics and has no concept of visual semantics. Neural networks learn semantic priors and meaningful hidden representations in an end-to-end fashion, which have been used for recent image inpainting efforts \cite{16,17,18,19}. These networks employ convolutional filters on images, replacing the removed content with inpainted areas that have geometrical and semantic consistency with the whole image.

Both challenges can also be seen as one single task: translating a dynamic image into a corresponding static image. In this direction, Isola et al. \cite{20} propose a general-purpose solution for image-to-image translation. Our previous work \cite{21} builds on top of this idea and re-formulates the framework objectives to take advantage of a pre-computed dynamic object mask, seeking a more inpainting-oriented framework. In this work, we follow this idea of transforming images with dynamic content into realistic static frames, while optimizing for localization and mapping performance. For such task, we introduce a new loss that, combined with the integration of a semantic segmentation network achieves the final objective of creating a dynamic-object-invariant space. This loss is based on steanalysis techniques and on ORB features detection, orientation and descriptor maps \cite{22}. Such loss allows the inpainted images to be realistic and suitable for localization and mapping. These images provide a richer understanding of the stationary scene, and could also be of interest for the creation of high-detail road maps (Fig. 1), as well as for street-view imagery suppliers as a privacy measure to replace faces and license plates blurring. We provide an extensive evaluation on robotic applications such as visual odometry, place recognition, and mapping to prove the validity of our framework.

II. RELATED WORK

A. Dynamic Objects Detection

The vast majority of SLAM systems assume a static environment. As a consequence, they can only manage small fractions of dynamic content by classifying them as spurious data or outliers to such static model. The most typical outlier rejection algorithms are RANSAC (e.g., in ORB-SLAM \cite{11,23}) and robust cost functions (e.g., in PTAM \cite{24}).

There are several SLAM systems that address more specifically the dynamic scene content. Tan et al. \cite{9} detect changes that take place in the scene by projecting the map features into the current frame for appearance and structure validation. Wang and Huang \cite{8} segment the dynamic objects in the scene using RGB optical flow. Alcantarilla et al. \cite{7} detect moving objects by means of a scene flow representation with stereo cameras. More recently, thanks to the boost of deep learning, integrating semantics information into SLAM has allowed to deal with dynamic content in a different manner \cite{25,6}. This idea allows the clustering of map points belonging to independent objects with different dynamics, as well as the possibility of detecting dynamic objects in just one shot.

B. Sequence-based Inpainting

Previous works on SLAM in dynamic scenes have attempted to reconstruct the background occluded by dynamic objects in the images with information from previous frames \cite{6,26}. Such works need per-pixel depth information and only make use of the static content of the pre-built map to create the inpainted frames, but do not add semantic consistency. The work by Granados et al. \cite{27} removes marked dynamic objects from videos by aligning other candidate frames in which parts of the missing region are visible, assuming that the scene can be approximated using piecewise planar geometry. The recent work by Uittenbogaard et al. \cite{28} utilizes a Generative Adversarial Network (GAN) to learn to use information from different viewpoints and select imagery information from those views to generate a plausible inpainting which is similar to the ground-truth static background. Eventually, if only one frame is available, the static occluded background can only be reconstructed by utilizing image-based inpainting techniques.

C. Image-based Inpainting

Among the non-learning approaches to image inpainting, propagating appearance information from neighboring pixels to the target region is the usual procedure \cite{14}. Accordingly, these methods succeed in dealing with narrow holes, where color and texture vary smoothly, but fail when handling big holes, resulting in over-smoothing. Differently, patch-based methods iteratively search for relevant patches from the rest of the image \cite{29}. These approaches are computationally expensive and hence not fast enough for real-time applications. Yet, they do not make semantically-aware patch selections.

Deep learning based methods usually initialize the image holes with a constant value, and further pass it through a CNN. Context Encoders \cite{19} were among the first ones to successfully use a standard pixel-wise reconstruction loss, as well as an adversarial loss for image inpainting tasks. Due to the resulting artifacts, Yang et al. \cite{30} take their results as input and then propagate the texture information from non-hole regions to fill the hole regions as post-processing. Song et al. \cite{31} use a refinement network in which a blurry initial hole-filling result is used as the input, then iteratively replaced with patches from the closest non-hole regions in the feature space. Izuka et al. \cite{18} extend Content Encoders by defining global and local discriminators, then apply a post-processing. Following this work, Yu et al. \cite{17} replaced the post-processing with a refinement network powered by the contextual attention layers. The recent work of Liu et al. \cite{16} obtains excellent results by using partial convolutions.

In contrast, the work by Ulyanov et al. \cite{32} proves that there is no need for external dataset training. The generative network itself can rely on its structure to complete the corrupted image. However, this approach usually applies several iterations (∼50000) to get good and detailed results.

D. Image Inpainting for a Dynamic-Object-Invariant Space

This work builds on our previous work Empty Cities \cite{21}, which bins the image sequences and treats the frames independently. It makes use of deep learning to segment out the
a priori moving objects: vehicles, animals and pedestrians, and also of image-based “inpainting”. It does not perform pure inpainting but image-to-image translation with the help of a dynamic objects’ mask, outcome of a semantic segmentation network. This choice is justified by the fact that the dynamic objects’ mask might be inaccurate or may not include their shadows. The adoption of an image-to-image translation framework allows to slightly modify the image non-hole regions for better accommodation of the reconstructed areas. Differently to inpainting methods, the “holes” cannot be initialized with any placeholder values because we do not want the framework to only modify those values and hence, our inpainting network input consists of the dynamic original image with the dynamic/static mask concatenated. Concisely, utilizing an image-to-image translation approach allows us to have the image hole regions inpainted, and the non-hole regions slightly modified for better accommodation of the reconstructed areas to cope with imprecise masks, or with dynamic objects possible shadows and reflections.

III. IMAGE-TO-IMAGE TRANSLATION

Our work makes use of the successful image-to-image translation framework by Isola et al. [20]. For the sake of completeness, we summarize the basis of their approach.

A GAN is a generative model that learns a mapping from a random noise vector $z$ to an output image $y$, $G: z \rightarrow y$ [33]. In contrast, a conditional GAN (cGAN) learns a mapping from observed image $x$ and optional random noise vector $z$, to $y$, $G: \{x, z\} \rightarrow y$ [34], or $G: x \rightarrow y$ [20]. The generator $G$ is trained to produce outputs indistinguishable from the “real” images by an adversarially trained discriminator, $D$, which is trained to do as well as possible at detecting the generator’s “fakes”. The objective of a cGAN can be expressed as

$$L_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_z[\log (1 - D(x,G(z)))],$$  

(1)

where $G$ tries to minimize this objective against an adversarial $D$ that tries to maximize it. Previous approaches have found it beneficial to mix the GAN objective with a more traditional appearance loss, such as the $L1$ or $L2$ distance [19]. The discriminator’s job remains unchanged, but the generator is tasked not only with fooling the discriminator, but also with being near the ground-truth in a $L1$ sense, as expressed in

$$G^* = \arg \min_G \max_D L_{cGAN}(G,D) + \lambda_1 \cdot L_{L1}(G),$$  

(2)

where $L_{L1}(G) = \mathbb{E}_{x,y}[\|y - G(x)\|_1]$. The recent work of Isola et al. [20] shows that cGANs are suitable for image-to-image translation tasks, where the output image is conditioned on its corresponding input image, i.e., it translates an image from one space into another (semantic labels to RGB appearance, RGB appearance to drawings, day to night, etc.). The realism of their results is also enhanced by their generator architecture. They employ a U-Net [35], which allows low-level information to shortcut across the network. In our previous work [21] we made use of this same architecture with $256 \times 256$ resolution images. However, visual localization systems see their accuracy degraded when working with low-resolution images. For this objective, we hereby employ as the architecture for our generator $G$ a UResNet [36], see Fig. 2. This architecture uses residual blocks [37] and has shown impressive results for super-resolution images [38].

It is well known that $L2$ and $L1$ losses produce blurry results on image generation problems, i.e., they can capture the low frequencies but fail to encourage high frequency crispness. This motivates restricting the GAN discriminator to only model high frequency structures. Following this idea, Isola et al. [20] adopt a discriminator architecture that classifies each $N \times N$ patch in an image as real or fake, rather than classifying the image as a whole. Due to their excellent results, we adopt this same architecture for our discriminator.

IV. OUR PROPOSAL

Our proposed system turns images of an urban environment that show dynamic content, such as vehicles or pedestrians, into realistic static frames which are suitable for localization and mapping. We first obtain the pixel-wise semantic segmentation of the RGB dynamic image (see Fig. 3). Then, the segmentation of only the dynamic objects is obtained with the convolutional network DynMS. Once we have this mask, we convert the RGB dynamic image to gray scale and we compute the static image, also in gray scale, with the use of the generator $G$, which has been trained in an adversarial way. For simplicity, the discriminator is not shown in this diagram. To fully exploit the capabilities of this framework for localization and mapping, inpainting is enriched with a loss based on ORB features detection, orientation and descriptors between the ground-truth and computed static images. Another feature of our framework for localization and mapping is the fact that we perform the inpainting in gray-scale rather than in RGB. The motivation for this is that many visual localization applications only need the images grayscale information. The different stages are described in subsections IV-A to IV-E.

A. From Image-to-Image Translation to Inpainting

For our objective, dynamic objects masks are specially considered to re-formulate the training objectives of the general purpose image-to-image translation work by Isola...
We adopt a variant of the cGAN that learns a mapping from observed image $x$ and dynamic/static binary mask $m$, to $y$, $G: \{x, m\} \rightarrow y$. Also, the discriminator $D$ learns to classify $\hat{y} = G(x, m)$ patches as “fake” from $\hat{y}$, $m$ and $x$, and the patches of $y$ as “real” from $y$, $m$ and $x$, $D: \{x, y, \hat{y}, m\} \rightarrow$ real/fake.

In most of the training dataset images, the relationship between the static and dynamic regions sizes is unbalanced, i.e., static regions occupy usually a much bigger area. This leads us to believe that the influence of dynamic regions on the final loss is significantly reduced. As a solution to this problem, we propose to re-formulate the $cGAN$ and $L_1$ losses so that there is more emphasis on the main areas that have to be inpainted, according to Eqs. 3 and 4. The weights $w$ are computed as $w = \frac{N}{N_{dyn}}$ if $m = 1$ (dynamic object), and as $w = \frac{N}{N - N_{dyn}}$ if $m = 0$ (background). $N$ stands for the number of elements in the binary mask $m$, and $N_{dyn}$ means the number of pixels where $m = 1$.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y}[w \cdot ||y - G(x, m)||_1], \quad (3)$$

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[w \cdot \log D(x, y, m)] + \mathbb{E}_x[w \cdot \log (1 - D(G(x, m), m))], \quad (4)$$

An important feature that we have also incorporated to the framework is the computation of our output and target images “noise”. This is motivated by the use of the noise domain for steganalysis to detect if an image has been tampered or not. Fig. 4 shows an example of why working in the noise domain is helpful for detecting “fake” images. While the static generated image (Fig. 4b) looks visually similar to its target image (Fig. 4d), their computed noises (Figs. 4c and 4e) are very different. It would be very easy for us, humans, to tell what parts of the original image (Fig. 4a) have been changed by analyzing their noise mapping. In the same way, the discriminator could more easily learn to distinguish “real” from “fake” images if it can take as input their noise. This idea is explained more in depth in subsection IV-B, and the whole training procedure is diagrammed in Fig. 5. To the best of our knowledge, steganalysis noise features have not been used before in the context of GANs.

Fig. 4: (b) shows the image generated by our framework when taking (a) as input. (d) is the static objective image. (b) and (d) are visually similar, but their computed noise (c) and (e) respectively clearly show what image and what parts of it have been modified the most. The noise magnitude has been amplified ($\times 10$) for visualization.
localization experiments (subsection VI-A) that not utilizing moving objects’ features leads to worse tracking results than working with fully static images. For that reason, we want to exploit our framework for obtaining both high-quality inpainting results, and to succeeding in generating reliable features for visual localization tasks. Fortunately, these two assignments are highly related. Solving one of them leads to having the other one tackled. Therefore, we have implemented a new loss based on ORB features [22]. That is, we want the output of our generator $G$ to have the same ORB features than its target image, while keeping it realistic and close to its target in a $L_1$ sense. By the same ORB features we mean the same detected keypoints with their same orientation and descriptors, following ORB’s implementation to the extent possible. This procedure is further described in subsection IV-C.

**B. Steganalysis-Based Loss**

With the advances of image editing techniques, tampered or manipulated image generation processes have become widely available. As a result, distinguishing authentic images from tampered images has become increasingly challenging.

What our framework is actually trying to achieve is to eliminate certain regions from an authentic image followed by inpainting, i.e., removal, one of the most common image manipulation techniques. It is the discriminator’s job to classify the generated image patches as tampered (fake) or real.

Images have a low-frequency component dependent on their content, and a high-frequency component dependent on their source camera. These high-frequency components are known as noise features or noise residuals, and can be extracted using linear and non-linear high-pass filters. Recent works on image forensics utilize noise features [39], [40] as clues to classify a specific patch or pixel in an image as tampered or not, and localize the tampered regions. The intuition behind this idea is that when an object is removed from one image (source) and the gap is inpainted (target), the noise features between the source and target are unlikely to match.

To provide the discriminator with better clues to distinguish real from fake inputs, we first extract the noise features from our images and concatenate them to the gray-scale images, as depicted in Fig. 5. The cGAN objective is re-formulated as

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[w \cdot \log D(x, y, m, n)] + \mathbb{E}_x[w \cdot \log (1 - D(x, \hat{y}, m, \hat{n}))],$$  \hspace{1cm} (5)

where $n = SRM(y)$ and $\hat{n} = SRM(\hat{y})$. There are many ways to produce noise features from an image. Inspired by recent progress on Steganalysis Rich Models (SRM) for image manipulation detection [39], we use SRM filter kernels to extract the local noise features from the static images as the input to our discriminator. The SRM use statistics of neighboring noise residual samples as features to capture the dependency changes caused by embedding. Zhou et al. [40] use SRM residuals, together with the RGB image to detect and localize corrupted regions in images. They only use three SRM kernels, instead of thirty (as in the original Fridrich et al.’s work [39]), and claim that they achieve comparable performance. Similarly, we use these same three filters (Fig. 6), setting the kernel size of the SRM filter layer to be $5 \times 5 \times 3$.

**C. ORB-Features-Based Loss**

ORB features allow real-time detection and description, and provide good invariance to changes in viewpoint and illumination. Furthermore, they are useful for visual SLAM and place recognition, as demonstrated in the popular ORB-SLAM [23] and its binary bag-of-words [41]. The following sections summarize how the ORB features detector, descriptors and orientation are computed, and how we have adapted them into a new loss.

1) **Detector:** The ORB Detector is based on the FAST algorithm [42]. It takes one parameter, the intensity threshold $t$ between the center pixel $p$, $I_p$, and those in a circular ring around the center. If there exists a set of contiguous pixels in the circle which are all brighter than $I_p + t$, or all darker than $I_p - t$, the pixel $p$ would be a keypoint candidate. Then the Harris corner measure is computed for each of these candidates, and the target $N$ keypoints with the highest Harris measure are finally selected. FAST does not produce multi-
scale features, therefore, ORB uses a scale pyramid of the image and extracts FAST features at each level in the pyramid.

To bring this to a differentiable solution, we have defined a convolution capable of detecting corners in an image in the same way that FAST does. We have approximated the FAST corner detection and have used instead a convolution with the kernels in Fig. 7. These images show some of the kernels used for corners detection for a circular ring of 3 pixels around the center. By convolving the image with these kernels for different kernel sizes, we obtain its corner response for the different image pyramid levels. We keep the maximum score per pixel and per level and raise each element to its 2nd power to equally leverage positive and negative responses. We then subtract a value which is equivalent to the FAST threshold \( t \), and apply a sigmoid operation. Its output is the probability of a pixel of being a FAST feature, and could also be seen as the Harris corner measure. Features for the output and target images are computed following this procedure. We define this network as \( \text{det} \), and the corresponding loss \( \mathcal{L}_{\text{det}}(G) \) can be expressed as

\[
\mathcal{L}_{\text{det}}(G) = -\mathbb{E}_{x,y}[w_{\text{det}} \cdot (\text{det}(y) \cdot \log(\text{det}(\hat{y})) + (1 - \text{det}(y)) \cdot \log(1 - \text{det}(\hat{y})))],
\]

where \( \hat{y} = G(x, m) \) and \( w_{\text{det}} \) is calculated following Eq. 7.

This weights definition allows us to leverage the uneven distribution of non-features and features pixels, and to affect a greater training time and the results are barely influenced. This can be achieved in a differentiable and convolutional manner by creating \( n \) kernels with the values \( \tau \), \( \sigma \), and \( \theta \) their orientation parameters.

\[
\mathcal{L}_{\text{ori}}(G) = -\mathbb{E}_{x,y}[w_{\text{ori}} \cdot ||\text{ori}(y) - \text{ori}(\hat{y})||_1].
\]

Even though these convolutions are applied to the whole image with a stride of 5 as in the detection loss, the weighting term \( w_{\text{ori}} \) in Eq. 8 has a value of 1 if a FAST feature has been detected in either the ground-truth static image or the output image, \( i.e. \), if \( \text{det}(y) > 0.5 \) or \( \text{det}(\hat{y}) > 0.5 \). Otherwise, the weighting term \( w_{\text{ori}} \) is set to 0.

3) Descriptor: The ORB descriptor is a bit string description of an image patch constructed from a set of binary intensity tests. Consider a smoothed image patch, \( p \), a binary test \( \tau \) is defined by

\[
\tau(p; x, y) = \begin{cases} 
1, & p(x) < p(y) \\
0, & p(x) \geq p(y) 
\end{cases}
\]

where \( p(x) \) is the intensity of \( p \) at a point \( x \). The feature is defined as a vector of \( n \) binary tests

\[
f_n(p) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; x_i, y_i)
\]

As in Rublee et al.’s work [22], we use a Gaussian distribution around the center of the patch and a vector length \( n = 256 \). This can be achieved in a differentiable and convolutional manner by creating \( n \) kernels with all values set to 0 except for those in the positions \( x \) and \( y \):

\[
k(z) = \begin{cases} 
1, & z = x \\
-1, & z = y \\
0, & \text{otherwise}
\end{cases}
\]

where \( k(z) \) is the value of the kernel \( k \) at a point \( z \). Convolving an image with these \( n \) kernels yields each pixel’s ORB descriptor (a negative output corresponds to the bit value 0 and a positive one to 1). This convolution is followed by a
sigmoid activation function. We define this network as \( desc \), and the corresponding loss \( L_{\text{desc}}(G) \) can be expressed as

\[
L_{\text{desc}}(G) = -E_{x,y}[w_{\text{desc}} \cdot (\text{desc}(y) \cdot \log(\text{desc}(\hat{y})) + \nonumber \\
(1 - \text{desc}(\hat{y})) \cdot \log(1 - \text{desc}(\hat{y}))],
\]

where the weights \( w_{\text{desc}} \) are defined in Eq. [13] This descriptors loss is back-propagated to the whole image, whether a feature has been detected or not, as it helps keeping the image statistics.

\[
w_{\text{desc}} = \begin{cases} 
1, & \text{desc}(y) > 0.5 \ & \& \text{desc}(\hat{y}) \leq 0.5 \\
1, & \text{desc}(y) \leq 0.5 \ & \& \text{desc}(\hat{y}) > 0.5 \\
0, & \text{otherwise}
\end{cases}
\]

All these losses are combined into one loss \( L_{\text{ORB}}(G) \), that is computed as in Eq. [14] The values for the weights of the different losses \( \lambda_{\text{det}}, \lambda_{\text{ori}} \) and \( \lambda_{\text{desc}} \) have been chosen empirically, and they are set to 10, 0.1 and 1 respectively.

\[
L_{\text{ORB}}(G) = \lambda_{\text{det}}L_{\text{det}}(G) + \lambda_{\text{ori}}L_{\text{ori}}(G) + \lambda_{\text{desc}}L_{\text{desc}}(G)
\]

The features detection, orientation and descriptor maps can be computed in a parallel way to decrease the training time, since their computation is not necessarily sequential.

Finally, the generator’s job can be expressed as in Eq. [15]

\[
G^* = \arg \min_G \max_D L_{\text{cGAN}}(G, D) + \lambda_1 \cdot L_{1}(G) + L_{\text{ORB}}(G)
\]

As an implementation detail, we have first trained the whole system without the ORB loss for 125 epochs, and then have fine-tuned including it for another 25 epochs.

D. Semantic Segmentation

Semantic Segmentation is a challenging task that addresses most of the perception needs of intelligent vehicles in a unified way. Deep neural networks excel at this task, as they can be trained end-to-end to accurately classify multiple object categories in an image at pixel level. However, very few architectures have a good trade-off between high quality and computational resources. The recent work of Romera et al. [11] runs in real time while providing accurate semantic segmentation. The core of their architecture (ERFNet) uses residual connections and factorized convolutions to remain efficient while retaining remarkable accuracy.

Romera et al. [11] have made public some of their trained models [45]. As in our preliminary work, we use for our approach the ERFNet model with encoder and decoder both trained from scratch on Cityscapes train set [46]. We have fine tuned their model to adjust it to our inpainting approach by back-propagating the loss of the semantic segmentation \( L_{\text{CE}}(SS) \), calculated with the cross entropy criterion using the class weights they suggest, \( w_{\text{SS}} \), and the adversarial loss of our final inpainting model \( L_{\text{cGAN}}(G, D) \). The semantic segmentation network’s job (SS) can be hence expressed as:

\[
SS^* = \arg \min_{SS} \max_D L_{\text{cGAN}}(G, D) + \lambda_2 \cdot L_{\text{CE}}(SS)
\]
TABLE I: Quantitative evaluations of our contributions in the inpainting task on the test synthetic images. The best results for almost all the inpainting metrics ($L_1$, PSNR and SSIM) are obtained with the generator $G(x, m)^w$ and the discriminator $D(x, y, m, n)^w$. More correct features (Feat metric) are detected though when adding the features based loss $G(x, m)^w_{ORB}$. Full image designates the per-pixel error considering the whole image. For In and Out we refer respectively to the error per pixel considering the masked and unmasked pixels.

of our work for the different reported inputs and losses. The existence of many possible solutions renders difficult to define a metric to evaluate image inpainting [17]. Nevertheless, we follow previous works and report the $L_1$, PSNR and SSIM errors [49], as well as a feature-based metric Feat. This last metric computes the FAST features detection as explained in subsection IV-C for the output and ground-truth images, and compares them, similarly to Eq. 6.

Adding the dynamic/static mask as input for both the generator and discriminator helps obtaining better inpainting results within the images hole regions (In), at the expense of having worse quality results in the non-hole regions (Out). Leveraging the unbalanced quantity of static and dynamic data within the dataset with $w$ (Eqs. 3 and 4) helps obtaining better results too. Providing the GAN’s discriminator with the images noise makes it learn better to distinguish between real and fake images, and therefore the generator learns to produce more realistic images. The ORB-based loss leads to slightly worse inpainting results according to $L_1$, PSNR and SSIM metrics, but renders this approach more useful for both localization and mapping tasks since more correct features are created.

Baselines for Inpainting
We compare qualitatively and quantitatively our “inpainting” method with four other state-of-the-art approaches:

- Geo1, Geo2: two non-learning approaches [14], [15].
- Lea1, Lea2: two deep learning based methods [17], [18].

Fig. 8: Qualitative comparison of our method (e) against other inpainting techniques (b), (c), (d) on our synthetic dataset. Our results are semantically and geometrically coherent, and do not show the dynamic objects’ shadows, even if they are not included in the input mask.
For a fair comparison, we have trained the approach by Yu et al. ([Lea1]) with our same training data. Iizuka et al. ([Lea2]) do not have their training code available. We have directly used their release model [18] trained on the Places2 dataset [51]. This dataset contains images of urban streets from a car perspective similar to ours. A more direct comparison is not possible. We provide them with the same mask than to our method to generate the holes in the images. We evaluate qualitatively on the 3000 images from our synthetic test dataset, on the 500 validation images from the Cityscapes dataset [46] and on the images from the Oxford Robotcar dataset [50]. We can see in Figs. 8, 9 and 10 the qualitative comparisons on these three datasets. Note that results generated with both [Lea1] and [Lea2] have been generated with the color images and then converted to gray scale for visual comparison. Visually, we see that our method obtains a more realistic output (these results are computed without the ORB loss for an inpainting oriented comparison). Also, it is the only one capable of removing the shadows generated by the dynamic objects even though they are not included in the dynamic/static mask (Fig. 8 row 2 and Fig. 10 row 1). The utilized masks are included in the images in Figs. 8a and 10a respectively.

Table II shows the quantitative comparison of our method against [Geo1], [Geo2], [Lea1] and [Lea2] on our CARLA dataset. It is not possible to quantitatively measure the performance of the different methods on the Cityscapes and Oxford Robotcar datasets, since ground-truth does not exist. By following these results, we can claim that our method outperforms both qualitatively and quantitatively the other approaches.

As seen in Fig. 10 row 1, the fact that our method does not perform pure inpainting but image-to-image translation with the help of a dynamic/static mask allows us to modify not only the dynamic objects themselves, but also their
shadows or reflections. We believe that the main underlying reasoning for this is the direct supervision for image-to-image translation. Also, since during the training with real-world data the segmentation masks are not 100% accurate, the model learns that it has to modify mainly the areas of the mask and, in case a smooth representation of the world is not obtained, also its surroundings. We believe that, was the training performed with perfect masks that also cover the shadows, the model would not learn to handle the shadows of dynamic objects or the inaccuracies of segmentation. We want to highlight the importance of the inpainting robustness to inaccurate segmentation masks since, in practice, partial or missing segmentation happens frequently. Empty Cities cannot handle missing detections but can cope with partial segmentations covering at least the 85% of the object image.

We hereby report some metrics evaluating how our framework behaves with the dynamic objects's shadows. Thresholding the difference between the dynamic and static image, and subtracting the dynamic-objects mask yields its dynamic objects shadows and reflections mask. We have first generated the shadows ground truth of our CARLA dataset, and then computed the shadows masks for our inpainted images in the same way. The intersection over union of the estimated shadows against the ground truth is 42.8%. Following recent works in shadow detection [52], we also report our method’s shadow, non-shadow and total accuracy (59.7%, 99.8% and 99.5% respectively). With this framework, we can remove almost 50% of the shadows of the dynamic objects of our CARLA test dataset. Admitting this could be improved, our method’s non-shadow accuracy is almost 100%, which means that it does not modify other objects’ shadows.

C. Transfer to Real Data

Models trained on synthetic data can be useful for real world vision tasks [53], [54], [55], [56]. Accordingly, we provide a study of synthetic-to-real transfer learning using data from the Cityscapes dataset [46], which offers a variety of urban real-world environments similar to the synthetic ones.

When testing our method on real data, we see qualitatively that the synthesized images show some artifacts. This happens because such data has different statistics than the real one,
and therefore cannot be easily used. The combination of real and synthetic data is possible during training despite the lack of ground-truth static real images. In the case of the real images, the network only learns the texture and the style of the static real world by encoding its information and decoding back the original image non-hole regions. The synthetic data is substantially more plentiful and has information about the inpainting process. The rendering, however, is far from realistic. Thus, the chosen representation attempts to bridge the reality gap encountered when using simulated data, and to remove the need for domain adaptation.

We provide implementation details: we have finetuned our model with real data for 25 epochs with a real/synthetic images ratio of 1/10. On the one hand, for every ten images there are nine synthetic images that provide our model with information about the inpainting task. On the other hand, one image out of those ten is a real image from the Cityscapes train dataset. There is groundtruth of its semantic information but there is no groundtruth of its static representation. In such cases we do backpropagation of the loss derivative only on those image areas that we consider as static. This way, the model can learn both the inpainting task and the static real-world texture. Once the model is adapted to real-world data, it can be directly used in completely new real-world scenarios, e.g., the Oxford Robotcar dataset.

VI. EXPERIMENTS

A. Visual Odometry

We have evaluated Empty Cities on 20 CARLA synthetic sequences and on 9 sequences from real-world new environments. For these VO experiments we have chosen the state-of-the-art feature-based system ORB-SLAM\(^1\) and the direct method DSO. The former is ideal to test the influence of our ORB features loss, and the latter is useful to prove that different systems can also benefit from this approach.

**Baselines for VO in Our Synthetic Dataset**

Fig. 11 displays the ORB-SLAM absolute trajectory RMSE [m] computed for 20 CARLA sequences of approximately 100 m long without loop closures. Fig. 11a shows the results when many vehicles and pedestrians are moving independently. More precisely, the number of vehicles and pedestrians have been set to the maximum allowed by CARLA. Fig. 11d shows the same odometry results for the ground-truth static sequences. We can see that dynamic objects have in many sequences a big influence on ORB-SLAM’s performance (sequences 02, 07, 08, etc.). Fig. 11b shows the trajectory error obtained with our previous system DynaSLAM. This system is based on ORB-SLAM and uses the semantic segmentation network Mask R-CNN to detect the moving objects and not extract ORB features within them. Even though better odometry results are obtained compared against using the raw dynamic images, this experiment shows that using static images leads to a more accurate camera tracking. One reason for this is that dynamic objects might occlude the nearby regions in the scene, which are the most reliable for camera pose estimation. Another reason might be that using dynamic objects masks does not yield anymore a homogeneous features distribution within the image. ORB-SLAM looks for a uniform distribution of image features. Pose optimization could be degraded and drifting could increase if the features do not follow such distribution. Finally, Fig. 11c shows the ORB-SLAM error when using our inpainted images. Our odometry results show that better results are usually obtained when using our inpainted images. The inpainting is realistic enough to provide the visual odometry system with consistent features that are useful for localization.

We want to highlight the importance of the influence of using the ORB loss during training (Fig. 12). Figs. 12a and 12b present the ATE obtained by ORB-SLAM with our inpainted images without and with the ORB loss respectively. The estimated errors are smaller and more constant when using this loss. This performance gain can be due to features in the regions that originally contained dynamic objects, and to more stable features in the static-content regions. Fig. 12c presents the DynaSLAM ATE with the inpainted images generated with our model trained with this loss term. We expect these errors to be very similar to those shown in Fig. 12b and slightly bigger than those in Fig. 12d. This experiment shows that our model barely damages the static content of the scene, keeping the static features as they used to be. It also demonstrates that the hallucinated features are useful to estimate the camera SE3 pose. To support this claim, on the right-most side of Fig. 12 one can find the percentage of keypoints extracted in the inpainted areas w.r.t. to the ground-truth keypoints. We show in Fig. 13 two examples of the visual influence of such loss on enhancing high frequencies inpainted areas.

Fig. 14 shows the DSO error for the same 20 CARLA sequences for the different input images with dynamic content (Fig. 14a), without dynamic content (Fig. 14b), and the images obtained by our framework without and with the ORB-based loss (Figs. 14c and 14d). Even though direct systems are more robust to dynamic objects within the scene, utilizing our approach also yields a higher tracking accuracy. Despite the fact that our feature-based loss follows the ORB implementation, better results are also obtained with other visual odometry system that do not rely on ORB features.

**Baselines for Inpainting**

We have compared in the section V the quality of our results with respect to the inpainting metrics against four other methods. We want to compare now how our approach compares to them w.r.t. visual odometry metrics. Among these four other methods we have chosen two of them for this evaluation: Geo1, and Leaf1. The first choice is motivated by its performance on our inpainting test dataset: this method performs the best among the two non-learning based approaches. The second choice is however motivated by the fact that we have not been able to train the model by Iizuka et al. with our training data for a direct comparison. The evaluation and different results can be seen in Fig. 15.

The images inpainted by Telea’s method are usually very smooth and no features are extracted within the inpainted areas. The behaviour of ORB-SLAM when using such sequences (Fig. 15a) is very similar to using DynaSLAM. However, the
Fig. 11: The vertical axis shows the different sequences from our CARLA dataset in which we have tested our model, and the horizontal axes show the ATE [m] obtained by ORB-SLAM. (a) shows the ORB-SLAM absolute trajectory RMSE [m] for the raw dynamic images. (b) shows the ATE computed by DynaSLAM [6]: this system is based on ORB-SLAM and computes the dynamic objects masks in every frame not to use features belonging to them. (c) shows the ORB-SLAM ATE when using our inpainted frames (the ones obtained with the ORB-features-based loss), and (d) shows the ORB-SLAM ATE for the ground-truth static images.

Fig. 12: The vertical axis shows the different sequences from our CARLA dataset, and the horizontal axes show the ORB-SLAM ATE [m]. (a) and (b) show the ORB-SLAM absolute trajectory RMSE [m] for our images obtained with the model trained without and with the ORB loss term respectively. (c) shows the ATE computed by DynaSLAM for our images obtained with the model trained with the ORB loss term. On the right-most side one can find the percentage of keypoints extracted in the inpainted areas, w.r.t. to the ground-truth keypoints.

Fig. 13: Visual comparison of the improvements achieved by utilizing the ORB-based loss (c) again not using it (b). The reconstructed curbs in (c) are sharper and more straight than those in (b).

Learning-based method by Yu et al. tends to inpaint the images with low frequency patterns found in the image static content, generating many crispy artifacts (see examples in Fig. 10). A higher ATE is observed when using such images (Fig. 15b). Our method seems to be more suitable for the VO task: the inpainting is neither too smooth nor generates crispy artifacts.

**Baselines for VO in Real World New Scenarios**

For the evaluation of Empty Cities on real world environments w.r.t. visual odometry we have chosen the KITTI [48] and the Oxford Robotcar [50] datasets. Dynamic objects in the KITTI dataset do not represent a big inconvenience for camera pose estimation, as was shown in our last work [6]. Most of the vehicles that appear are not moving and lay in nearby scene areas, thus their features happen to be helpful to compute the sensor odometry. Also, the few moving pedestrians and cars along the sequences do not represent a big region within the images. The Oxford Robotcar dataset has though many sequences with representative moving objects (driving cars), having also sequences with only stationary objects (parked cars). Note that, the reported results are not for the whole sequence since the authors provide their VO solution as ground-truth, stating that it is accurate over hundreds of metres. Hence, the sequences we use are between 100 and 300 m long.

To perform the KITTI experiment we have re-trained our network with $256 \times 768$ resolution images for a better adaptation. The CARLA camera intrinsics have also been modified to match the ones used in KITTI. In this case, we have shifted the previously used semantic segmentation model trained only on Cityscapes for the ERFNet model with encoder trained on ImageNet and decoder trained on Cityscapes train set, and have finetuned it with the KITTI semantic segmentation.
training dataset. The generator and discriminator have also been finetuned with such data as explained in subsection V-C.

Fig. 16 shows the evaluation of our method performance with the sequences from the Oxford Robotcar and the KITTI datasets. The asterisk at the beginning of some sequences names means that most of the observed vehicles are either not moving or parked. The other sequences though present many moving vehicles. In the former type of sequences (*) the highest accuracy should be observed in the case in which the raw images are used (Fig. 16a). Removing the features from such vehicles as in Fig. 16b leads to a lower accuracy since the most nearby features are no longer used. inpainting the static scene behind these vehicles (Fig. 16c) would still remove nearby features but would create new static features a little bit further. That is, the visual odometry accuracy should be lower than in Fig. 16a but a little bit higher or similar to Fig. 16b. This is the case of our performance on the Oxford Robotcar sequences and the KITTI sequence 03. However, DynaSLAM achieves a result in the KITTI sequence 07 better the proposed Empty Cities. After the first half of the sequence, there are a few consecutive frames in which a truck covers almost 75% of the image. The task of inpainting becomes especially difficult, thus worsening the estimation of the camera’s trajectory. Regarding the performance of the second type of sequences, removing the features from moving vehicles and pedestrians should lead to a lower ATE (Fig. 16b compared to Fig. 16a). Empty Cities adds in these sequences an important number of features for pose estimation that usually leads to a slightly better trajectory estimation (Fig. 16c). Note that the results given in here for the KITTI sequences might not match the ones reported by ORB-SLAM and DynaSLAM respectively because of the utilized images resolution (256 × 768).

B. Visual Place Recognition

Visual place recognition (VPR) is an important task for visual SLAM. Such algorithms are useful when revisiting places to perform loop closure and correct the accumulated drift along long trajectories. Bags of visual words (BoW) is the approach that is widely used to perform such task, as can be seen in ORB-SLAM [23] and LDSO [57]. Lately, thanks to the boost of deep learning, learnt global image descriptors are also used for VPR [58]. In our previous work [21] we showed preliminary results proving the benefits of our solution for VPR by using descriptors from an off-the-shelf CNN [59].

Baseline for VPR in our Synthetic Dataset

In this subsection we show a VPR experiment performed with the bag of words work by Mur and Tardos [60], [41]. It is ideal to test our model since is based on ORB features. We also show an experiment with one of the strongest learning-based baselines NetVLAD [58], which is trained for the specific task.
of VPR. This comparison can provide a broader understanding of how and when end-to-end task-specific learning becomes more or less suitable than an explicit use of semantics-based visual description, which forms the primary pitch of this paper.

We have generated two CARLA sequences with loop closures with and without dynamic objects. Two images are defined as the same place if they are less than 10 meters apart.

The precision-recall curves for the BoW experiments are depicted in Fig. 17a. We have extracted the visual words of every frame along the trajectories and have tried to match every two images as a function of the number of common visual words. For both trajectories the results obtained with the dynamic images with and without masks are similar. This is congruent with the idea that the database of visual words mostly contains static and long-term stable words. The first trajectory is a good example of how the VPR recall drops fast in presence of dynamic objects: a place is better represented with words from the whole static image. It leads to less false positives and less false negatives mostly. Even though our method slightly brings closer in the bag-of-words space images from the same place with different dynamic objects, we would have expected our results to be closer to those of the ground-truth static images. Our intuition behind these results is that the synthetic features, despite being useful for feature matching, do not fully fall on any visual words in the BoW space.

The precision-recall curves for the NetVLAD experiments are shown in Fig. 17b. We have extracted the learnt descriptors of every frame along the trajectories and tried to match every two images as a function of their descriptors’ Euclidean distance. Despite NetVLAD’s incredible performance on VPR face to illumination, view point and clutter changes, we show that their execution is seen slightly degraded by dynamic objects.

The areas which have been consistently inpainted along the frames are mapped, even if they have never been seen. When inpainting fails or is not consistent along the sequence, the mapping photometric and geometric epipolar constraints are not met and such areas cannot be reconstructed. This idea makes our framework suitable to build stable maps.

of how and when end-to-end task-specific learning becomes more or less suitable than an explicit use of semantics-based visual description, which forms the primary pitch of this paper.

We have generated two CARLA sequences with loop closures with and without dynamic objects. Two images are defined as the same place if they are less than 10 meters apart.

The precision-recall curves for the BoW experiments are depicted in Fig. 17a. We have extracted the visual words of every frame along the trajectories and have tried to match every two images as a function of the number of common visual words. For both trajectories the results obtained with the dynamic images with and without masks are similar. This is congruent with the idea that the database of visual words mostly contains static and long-term stable words. The first trajectory is a good example of how the VPR recall drops fast in presence of dynamic objects: a place is better represented with words from the whole static image. It leads to less false positives and less false negatives mostly. Even though our method slightly brings closer in the bag-of-words space images from the same place with different dynamic objects, we would have expected our results to be closer to those of the ground-truth static images. Our intuition behind these results is that the synthetic features, despite being useful for feature matching, do not fully fall on any visual words in the BoW space.

The precision-recall curves for the NetVLAD experiments are shown in Fig. 17b. We have extracted the learnt descriptors of every frame along the trajectories and tried to match every two images as a function of their descriptors’ Euclidean distance. Despite NetVLAD’s incredible performance on VPR face to illumination, view point and clutter changes, we show that their execution is seen slightly degraded by dynamic objects. Empty Cities brings closer together the descriptors of the same place, and pulls apart the descriptors of different places with the same dynamic objects, leading to higher precision and recall. That is, the hidden semantic representations of our model match the ones learnt by NetVLAD. We can conclude that our method brings more relevant improvements in VPR if it is used in conjunction with a learning-based method. Finally, Fig. 18 shows a case in which NetVLAD fails at matching two dynamic frames of the same place, but manages to match them when dynamic objects are inpainted with our framework.

Baseline for Inpainting
We compare our approach to the inpainting methods (Geo1 [14] and Lea1 [17]) w.r.t. the VPR metrics in Fig. 17. For the BoW curves the conclusion is similar to what we have seen in the previous experiment. Even if the extracted synthetic features were useful for matching, they seem to be of less help for place recognition with BoW. Few synthetic ORB features match any existing visual word in the BoW space. Our method though creates more useful visual words than Lea1 and Geo1. As for the results with NetVLAD, the use of the geometric method Geo1 brings little improvement. However, the learning-based method Lea1 decreases NetVLAD’s performance on the second trajectory. NetVLAD can –up to some extent– ignore the dynamic classes clues, but cannot ignore the transformed regions. That is, the hidden semantic representations of these inpainted regions do not always match the static scene representation learnt by NetVLAD.

C. Mapping
Another important application of our framework is the creation of high-detail road maps. Our inpainting framework allows to create long-term and reusable maps that neither show dynamic objects nor the empty spaces left by them. To this end we use the MVS and SfM software COLMAP [62], [63].

Baseline for Mapping in our Synthetic Dataset
Fig. 19 shows the dense map of a simulated city environment (CARLA) with the original dynamic sequence, with the images processed by our framework and with the ground-truth static images. The map seen in Fig. 19a is not useful for future use since it shows dynamic objects that might not be there any more. Fig. 19c shows the map built with the ground-truth static images, and Fig. 19b shows the map computed with our generated images. The areas which have been consistently inpainted along the frames are mapped, even if they have never been seen. When inpainting fails or is not consistent along the sequence, the mapping photometric and geometric epipolar constraints are not met and such areas cannot be reconstructed. This idea makes our framework suitable to build stable maps.
Baseline for Inpainting

We also compare our approach with the two other state-of-the-art inpainting methods (Geo1 [14] and Lea1 [17]) w.r.t. the map quality. The qualitative results are depicted in Fig. 20 and the ICP scores are shown in Table III. It can be seen that with both other inpainting methods the shadows of the dynamic objects are reconstructed, as well as their prolongation into the inpainted image regions. This leads to an ICP score that is higher than that of the map built with dynamic objects.

Baseline for Mapping in Real World Environments

Fig. 21 shows an example of the computed dense maps for both types of inputs with the sequence 04 from the KITTI dataset. These maps have been computed with the camera poses given by ORB-SLAM using respectively the dynamic and inpainted images. Our framework, when being able to inpaint a coherent context along the sequence, allows us to densely reconstruct unseen areas. When the inpainting task is not coherent along the sequence, the epipolar constraints are not met and therefore such areas cannot be reconstructed.

D. Timing Analysis

Reporting our framework efficiency is crucial to judge its suitability for autonomous driving and robotic tasks in general. The end-to-end pipeline runs at 100 fps on a nVidia Titan Xp 12GB with images of a $512 \times 512$ resolution. Out of the 10 ms it takes to process one frame, 8 ms are invested into obtaining its semantic segmentation, and 2 ms are used for the inpainting task. Other than to deal with dynamic objects, the semantic segmentation may be needed for many other tasks involved in autonomous navigation. In such cases, our framework would only add 2 extra ms per frame. Based on our analysis, we consider that the inpainting task is not a bottleneck.

VII. Failure Modes and Future Work

The aim of this section is to provide the reader with an understanding on the benefits and limitations of our proposal, and on how to integrate it on a VO, SLAM or MVS pipeline.

Having a scene static representation leads to better visual odometry results than just excluding the features belonging to moving objects. The presented inpainting approach has though some weaknesses: the bigger the image dynamic region is, the smaller the ICP score is.

TABLE III: Quantitative mapping results for Figs. 19 and 20. We report the Euclidean fitness score given by the ICP algorithm. This score has been computed between the different point clouds w.r.t. the ground-truth static images. Note that, since the maps do not have scale, the reported scores are up-to-scale.

| Method       | Dynamic | Dynamic w/ masks | Ours  | Geo1 [14] | Lea1 [17] |
|--------------|---------|------------------|-------|-----------|-----------|
| ICP score    | 63.21   | 49.02            | **33.34** | 82.23     | 70.89     |

Fig. 17: (a) and (b) show the precision and recall curves for the VPR results with BoW and NetVLAD respectively. We report the results for the dynamic, the inpainted and ground-truth static sequences, as well as the results obtained when masking out the dynamic objects as in DynaSLAM (only in (a)). The precision and recall curves for the inpainting methods Geo1 [14] and Lea1 [17] are also presented.

Fig. 18: (a) and (b) show the same location with different viewpoints and objects setups. NetVLAD [58] fails to match them, but it succeeds when our framework is previously employed (c) and (d).

To give a quantitative experiment on the validity of our maps, we choose the standard Iterative Closest Point (ICP) algorithm [64] based on the Euclidean fitness score. That is, having the point cloud built with the ground-truth static images as a fix reference, for each of its points the algorithm searches for the closest point in the target point cloud and calculates the distance based on the result of the algorithm search. To have a baseline for our experiment, we compute the point cloud with the dynamic images and with the CARLA segmentation. That is, the pixels belonging to the dynamic objects have not been used in the multi-view-stereo pipeline. The results can be seen in Table III. Since the map has no scale, the ICP Euclidean fitness score is also up-to-scale. Even though the similarity score is improved when masking out dynamic objects, the map built with our images has triangulated more 3D points from the inpainted regions, and such points have a low error.
the lower inpainting quality the resulting image has. Empty
Cities would be suitable in setups in which approximately less
than 15% of the camera field of view is covered by dynamic
objects. In such setups, the reconstructed L1 error is acceptable
and usually lies between 1 and 10%. The L1 error goes above
10% when more than 15% of the image pixels are covered.4
Work remains to be done to tackle extreme situations. Also,
developing a system that processes the sequence as a whole,
rather than binning it into independent frames, would result in
a more consistent image inpainting along time.

Our system processes the image streams outside of the
application pipeline and hence can be used naturally as a front
to many existing systems. We hereby want to discuss other
application-dependent possibilities to boost its performance.

Visual Odometry: Removing the features that belong to
stationary objects certainly damages VO. However, e.g., a car
can from one frame to another change from static to moving.
Had we a movement detector, we would use the static objects'features and the inpainted ones behind moving objects.

Place Recognition: Using the features of stationary dy-
namic objects damages the performance of place recognition
algorithms. For example, two frames of the same place with
a different setup of parked cars can be incorrectly tagged as a
different place. Also, two frames of different places with the
same parked cars setup can be wrongly matched as the same
place. Only the features of objects that remain stable in the
long term (buildings, sidewalks, etc.) would benefit VPR.

Mapping: A map containing information about dynamic
objects would be useless for future reuse. Only the information
belonging to objects that remain stable in the long term, as
well as the most likely static representation of the static scene
behind dynamic objects should be included in the map.

Ideally, one would use a movement detector to identify the
status of the different observed instances and also to allow
the discovery of new dynamic classes on the fly [13]. The
features belonging to static instances would be used for visual
odometry, and the corresponding inpainted features would be
used for place recognition and mapping. This approach would
bring the highest accuracy results but would entail a series
of modifications to the existing pipeline. Our method would
currently pose problems in the case in which one wanted to
inpaint the static scene behind a car that is currently moving,
and this static scene contained parked cars. Our model would
fail to reconstruct the unseen parts of the parked cars. It would
be interesting for future work to include such scenarios in
our training data. Our suggestion is, for now, inpainting all
instances, regardless of their current dynamic status.

---

4As a practical example, the Oxford Robotcar sequence 2014-05-06-12-
54-54 has 56% of images with less than 5% of covered pixels, 30% with
a percentage of dynamic pixels between 5 and 10%, 12% between 10 and
15%, and 2% between 15 and 20%. This sequence is not Manhattan at 11
am, but shows cars parked at both sides of the road and cars driving nearby.
Fig. 21: Dense maps built with images from the KITTI dataset [48]. The upper row shows some of the sequence images used to build each map. (a) shows the case in which the original dynamic images have been used, and (b) shows the resulting map with the images previously processed by our framework. Both maps are computed with the camera poses that ORB-SLAM estimates for the different sequences.

VIII. CONCLUSIONS

We have presented an end-to-end deep learning framework that translates images that contain dynamic objects within a city environment, such as vehicles or pedestrians, into a realistic image with only static content. These images are suitable for visual odometry, place recognition and mapping tasks thanks to a new loss based on steganalysis techniques and ORB features maps, descriptors and orientation. We motivate this extra complexity by showing quantitatively that the systems ORB-SLAM and DSO obtain a higher accuracy when utilizing the images synthesized with this loss. Also, mapping systems can benefit from this approach since not only they would not map dynamic objects but also would they map the plausible static scene behind them. Finally, an architectural nicety is that our system processes the image streams outside of the localization pipeline, either offline or online, and hence can be used naturally as a front end to many existing systems.

IX. ACKNOWLEDGEMENTS

This work is supported by the Spanish Ministry of Economy and Competitiveness (project PID2019-108398GB-I00 and FPI grant BES-2016-077836), the Aragón regional government (Grupos DGA T45-17R, T45-20R), the EU H2020 research project under grant agreement No 688652 and the Swiss State Secretariat for Education, Research and Innovation (SERI) No 15.0284 and NVIDIA, through the donation of a Titan X GPU.

Berta Bescos was born in Zaragoza, Spain, in 1993. She received a Bachelor’s and a M.S. Degree in Industrial Engineering with mention in Robotics and Computer Vision from the University of Zaragoza, where she is currently pursuing her Ph.D. degree at the I3A Robotics, Perception and Real Time Group. Her research interests lie in the intersection between perception and learning for robotics. More concisely, her Ph.D. topic involves dealing with dynamic objects in SLAM for a better scene understanding.

Cesar Cadena received the Ph.D. degree in Computer Science from University of Zaragoza, Zaragoza, Spain, in 2011. He is a Senior Researcher at ETH Zurich, Switzerland. His main research interest lies in the interception of perception and learning in robotics. He is particularly interested on how to provide machines the capability of understanding this ever changing world through the sensory information they can gather. He has work intensively on Robotic Scene Understanding, both geometry and semantics, covering Semantic Mapping, Data Association and Place Recognition tasks, Simultaneous Localization and Mapping problems, as well as persistent mapping in dynamic environments.

José Neira was born in Bogotá, Colombia, in 1963. He received the M.S. degree in computer science from the Universidad de los Andes, Colombia, in 1986, and the Ph.D. degree in computer science from the University of Zaragoza, Spain, in 1993. Since 2010, José is Full Professor at the Departamento de Informática e Ingeniería de Sistemas, University of Zaragoza, where he is in charge of courses in compiler theory, computer vision, machine learning and mobile robotics. His current research interests are centered around Robust, Life-long Simultaneous Localization and Mapping. José also coordinates the university’s Master Program in Robotics, Graphics and Computer Vision.
[44] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” Computer vision and image understanding, vol. 110, no. 3, pp. 346–359, 2008.
[45] E. Romera, J. M. Alvarez, L. Bergasa, and R. Arroyo, “ERFNet,” https://github.com/Eromera/erfnet, 2017.
[46] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3213–3223, 2016.
[47] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in Proceedings of the 1st Annual Conference on Robot Learning, pp. 1–16, 2017.
[48] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” IJRR, vol. 32, no. 11, pp. 1231–1237, 2013.
[49] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, et al., “Image quality assessment: from error visibility to structural similarity,” IEEE transactions on image processing, vol. 13, no. 4, pp. 600–612, 2004.
[50] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 Year, 1000km: The Oxford RobotCar Dataset,” The International Journal of Robotics Research (IJRR), vol. 36, no. 1, pp. 3–15, 2017.
[51] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.
[52] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.
[53] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.
[54] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.
[55] J. Skinner, S. Garg, N. S. Schröder, P. Corke, B. Upcroft, and M. Milford, “High-fidelity simulation for evaluating robotic vision performance,” in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pp. 2737–2744, IEEE, 2016.
[56] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, pp. 23–30, IEEE, 2017.
[57] X. Gao, R. Wang, N. Demmel, and D. Cremers, “LDSO: Direct Sparse Odometry with Loop Closure,” International Conference on Intelligent Robots and Systems, 2018.
[58] M. Perić, S. Martull, A. Maki, Y. Ohkawa, and K. Fukui, “Towards a simulation driven stereo vision system,” in Pattern Recognition (ICPR), 2012 21st International Conference on, pp. 1038–1042, IEEE, 2012.
[59] J. Skinner, S. Garg, N. S. Schröder, P. Corke, B. Upcroft, and M. Milford, “High-fidelity simulation for evaluating robotic vision performance,” in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pp. 2737–2744, IEEE, 2016.
[60] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pp. 23–30, IEEE, 2017.
[61] X. Gao, R. Wang, N. Demmel, and D. Cremers, “LDSO: Direct Sparse Odometry with Loop Closure,” International Conference on Intelligent Robots and Systems, 2018.
[62] M. Perić, S. Martull, A. Maki, Y. Ohkawa, and K. Fukui, “Towards a simulation driven stereo vision system,” in Pattern Recognition (ICPR), 2012 21st International Conference on, pp. 1038–1042, IEEE, 2012.
[63] J. Skinner, S. Garg, N. S. Schröder, P. Corke, B. Upcroft, and M. Milford, “High-fidelity simulation for evaluating robotic vision performance,” in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pp. 2737–2744, IEEE, 2016.
[64] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, pp. 23–30, IEEE, 2017.
[65] X. Gao, R. Wang, N. Demmel, and D. Cremers, “LDSO: Direct Sparse Odometry with Loop Closure,” International Conference on Intelligent Robots and Systems, 2018.
[66] M. Perić, S. Martull, A. Maki, Y. Ohkawa, and K. Fukui, “Towards a simulation driven stereo vision system,” in Pattern Recognition (ICPR), 2012 21st International Conference on, pp. 1038–1042, IEEE, 2012.
[67] J. Skinner, S. Garg, N. S. Schröder, P. Corke, B. Upcroft, and M. Milford, “High-fidelity simulation for evaluating robotic vision performance,” in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pp. 2737–2744, IEEE, 2016.
[68] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, pp. 23–30, IEEE, 2017.
[69] X. Gao, R. Wang, N. Demmel, and D. Cremers, “LDSO: Direct Sparse Odometry with Loop Closure,” International Conference on Intelligent Robots and Systems, 2018.
[70] M. Perić, S. Martull, A. Maki, Y. Ohkawa, and K. Fukui, “Towards a simulation driven stereo vision system,” in Pattern Recognition (ICPR), 2012 21st International Conference on, pp. 1038–1042, IEEE, 2012.
[71] J. Skinner, S. Garg, N. S. Schröder, P. Corke, B. Upcroft, and M. Milford, “High-fidelity simulation for evaluating robotic vision performance,” in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pp. 2737–2744, IEEE, 2016.