Visual-Inertial-Semantic Scene Representation for 3D Object Detection

Jingming Dong* Xiaohan Fei* Stefano Soatto
UCLA Vision Lab, University of California, Los Angeles, CA 90095
{dong, feixh, soatto}@cs.ucla.edu

Abstract

We describe a system to detect objects in three-dimensional space using video and inertial sensors (accelerometer and gyrometer), ubiquitous in modern mobile platforms from phones to drones. Inertials afford the ability to impose class-specific scale priors for objects, and provide a global orientation reference. A minimal sufficient representation, the posterior of semantic (identity) and syntactic (pose) attributes of objects in space, can be decomposed into a geometric term, which can be maintained by a localization-and-mapping filter, and a likelihood function, which can be approximated by a discriminatively-trained convolutional neural network. The resulting system can process the video stream causally in real time, and provides a representation of objects in the scene that is persistent: Confidence in the presence of objects grows with evidence, and objects previously seen are kept in memory even when temporarily occluded, with their return into view automatically predicted to prime re-detection.

1. Introduction

We deem an “object detector” to be a system that takes as input images and produces as output decisions as to the presence of objects in the scene. We design one based on the following premises: (a) Objects exist in the scene, not in the image; (b) they persist, so confidence on their presence should grow as more evidence is accrued from multiple (test) images; (c) once seen, the system should be aware of their presence even when temporarily not visible; (d) such awareness should allow it to predict when they will return into view, based on scene geometry and topology; (e) objects have characteristic shape and size in 3D, and vestibular (inertial) sensors provide a global scale and orientation reference that the system should leverage on.

Detecting objects from images is not the same as detecting objects (Fig. 5). Objects do not flicker in-and-out of existence, and do not disappear when not seen (Fig. 6). What we call “object detectors” traditionally refers to algorithms that process a single image and return a decision as to the presence of objects of a certain class in said image, missing several critical elements (a)-(e) above. Nevertheless, such algorithms can be modified to produce not decisions, but evidence (likelihood) for the presence of objects, which can be processed over time and integrated against the geometric and topological structure of the scene, to yield an object detector that has the desired characteristics. The scene context encompasses both the identity and co-occurrence of objects (semantics) but also their spatial arrangement in three-dimensional (3D) space (syntax).

1.1. Summary of Contributions and Limitations

To design an object detector based on the premises above, we (a) formalize an explicit model of the posterior probability of object attributes, both semantic (identity) and syntactic (pose), natively in the 3D scene (Sect. 3), which (b) maintains and updates such a posterior, processing each image causally over time (Sect. 3.2); (c) the posterior distribution is a form of short-term memory (representation), which we use to (d) predict visibility and occlusion relations (Sect. 5.3). We exploit the availability of cheap inertial sensors in almost every mobile computing platform to (e) impose class-specific priors on the size of objects (Sect. 5.2).

The key insight from the formalization (a) above is that an optimal (minimal sufficient invariant [59]) representation for objects in the scene (Eq. 1) can be factored into two components: One geometric – which can be computed recursively by a localization (SLAM) system (Eq. 3) – and the other a likelihood term, which can be evaluated instantaneously by a discriminatively-trained convolutional neural network (CNN, Eq. 4) operating on a single image. Some consequences of this insight are discussed in Sect. 6. In practice, this means that we can implement our system using some off-the-shelf components, fine-tuning a pre-trained CNN, and at least for some rudimentary modeling assumptions, our system operates in real-time, generating object-scene representations at 10-30 frames per second. In Sect. 5 we report the results of a representative sample of
we start by defining an object representation as a mantic description for limitations. First, it only returns a joint geometric and segmentation is general, our real-time implementation has several deformation priors – is not inferred. Second, it models estimating which would require sophisticated class-specific visual recognition (Fig. 1). Moving objects are currently tracked points (red), viewer trajectory from a previous loop (yellow) and current pose (reference frame). All cars detected are shown as point-estimates (the best-aligned generic CAD model) in green, including those previously-seen on side streets (far left). Middle: visualization of the implicit measurement process: Objects in the state are projected onto the current image based on the mean vehicle pose estimate (green boxes) and their likelihood score is computed (visualized as contrast: sharp regions have high likelihood, dim regions low). Cars in different streets, known to not be visible, are visualized as dashed boxes and their score discarded. Bottom: Top view of the state from the entire KITTI-00 sequence (best viewed at 5×).

2. Related Work

This work, by its nature, relates to a vast body of literature on scene understanding in Computer Vision, Robotics [38, 47] and AI [36] dating back decades [66]. Most recently, with the advent of cheap consumer range sensors, there has been a wealth of activity in this area [41, 62, 68, 9, 58, 17, 26, 32, 52, 28, 5, 55, 65, 33, 61, 50]. The use of RGB-D cameras unfortunately restricts the domain of applicability mostly indoors and at close range whereas we target mobility applications where the camera, which typically has an inertial sensor strapped on it, but not (yet) a range sensor, can be used both indoor and outdoors. We expect that, on indoor sequences, our method would underperform a structured light or other RGB-D source, but this is subject of future investigation.

There is also work that focuses on scene understanding from visual sensors, specifically video [37, 2, 42, 57, 6, 72], although none integrates inertial data, despite a resurgent interest in sensor fusion [73]. Additional related work includes [27, 15, 8, 54].

To the best of our knowledge, no work leverages inertial sensing for object detection. This is critical to provide a scale estimate in a monocular setting, and validate object
hypotheses in a Bayesian setting, so that, for instance, a model car in our system is not classified as a car (Fig. 5).

Semantic scene understanding from a single image is also an area of research ([21] and references therein). We are instead interested in agents embedded in physical space, for which the restriction to a single image is limiting. There is also a vast literature on scene segmentation ([30] and references therein), mostly using range (RGB-D) sensors. One popular pipeline for dense semantic segmentation is adopted by [28, 44, 65, 37, 3]: Depth maps obtained either from RGB-D or stereo are fused; 2D semantic labeling is transferred to 3D and smoothed with a fully-connected CRF [35]. Also related methods on joint semantic segmentation and reconstruction are [53, 64, 7].

There is also work on 3D recognition [34, 56, 45], but again with no inertial measurements and no motion. Some focus on real-time operation [16], but most operate off-line [75, 12]. None of the datasets commonly used in these works [14, 71] provide an inertial reference, except for KITTI. In terms of 3D object detection on KITTI, some authors focus on image-based detection [24, 23, 49, 48, 43] and then place objects into the scene [69, 70], while others focus on 3D object proposal generation and verification using a network [10, 11]. [69] trains a 3D Voxel Pattern (3DVP) based detector to infer object attributes and demonstrates the ability to accurately localize cars in 3D on KITTI. Their subsequent work [70] trains a CNN to classify 3DVPs. Different representations of object proposals are also exploited, such as 3D cuboids [20] and deformable 3D wireframes [75]. Various priors are also considered: [67] exploits geo-tagged images; geometric priors of objects are incorporated into various optimization frameworks to estimate object attributes [74, 12]. While most of these algorithms report very good performance on detection (~90% mean average precision), none reports scores for the semantic-syntactic state of objects in 3D, except for [69, 70] and [11, 10]. Since the latter are dominated by the former, we take [70] as a paragon for comparison in Sect. 5.

The aforementioned 3D object recognition methods are based on 2D detection without temporal consistency. Therefore, the comparison is somewhat unfair as single-image based detectors cannot reliably detect objects in space, which is our main motivation for the proposed approach. For details on comparison methodology, see Sect. 5. [12, 60] use multiple views, but their output is a point-estimate instead of a posterior. Also, the optimization has to be re-run once new datum is available.

Recent work in data association [39] aims to directly infer the association map, which is computationally prohibitive for the scale needed in our real-time system. We therefore resort to heuristics, described in Sect. 4. More specifically to our implementation, we leverage existing visual-inertial filters [29, 40, 63] and single image-trained CNNs [24, 48, 70].

3. Methods

3.1. Representations

A scene $\xi$ is populated by a number of objects $z_j \in \{z_1, \ldots, z_N\}$, each with geometric (pose, shape)\(^1\) and semantic (label) attributes $y_j = \{s_j, l_j\}$. Measurements (e.g., images) up to the current time $t$, $y^t = \{y_1, \ldots, y_t\}$ are captured from a sensor at pose $g_t$. A semantic representation of the scene is the joint posterior $p(\xi, z^t | y^t)$ for up to the j-th objects seen up to time $t$, where sensor pose $g_t$ and other nuisances are marginalized. The joint posterior can be decomposed as $p(\xi, z^t | y^t) = p(\xi | y^t)p(z^t | y^t)$ with the first factor ideally updated asynchronously each time a new object $z_{j+1}$ becomes manifest starting from a prior $p(\xi)$ and the second factor updated each time a new measurement $y_{t+1}$ becomes available starting from $t = 0$ and given $p(z)$.

A representation of the scene in support of (geometric) localization tasks is the posterior $p(g_t, x | y^t)$ over sensor pose $g_t$ (which, of course, is not a nuisance for this task) and a sparse attributed\(^2\) point cloud $x = \{x_1, \ldots, x_{N_x}\}$, given all measurements (visual $P$ and inertial $u_t$) up to the current time. Conditioning the semantics on the geometry we can write the second factor above as

$$p(z^t | y^t) = \int p(z^t | g_t, x, y^t) dP(g_t, x | y^t)$$

(1)

where the integrand can be updated as more data $y_{t+1}$ becomes available as $p(z^t | g_{t+1}, x, y^{t+1})$, which is proportional to

$$p(g_{t+1} | z^t, g_{t+1}, x) \int p(g_{t+1} | g_t, u_t) dP(z^t | g_t, x, y^t).$$

(2)

3.2. Approximations

The measure in (1) can be approximated in wide-sense using an Extended Kalman Filter (EKF), as customary in simultaneous localization and mapping (SLAM): $p(g_t, x | y^t) \approx \mathcal{N}(\hat{g}_{t|t}, \hat{x}_{t|t}; P_{t|t})$. (1) is a diffusion around the mean/mode $\hat{g}_{t|t}, \hat{x}_{t|t}$; if the covariance $P_{t|t}$ is small, it can be further approximated: Given

$$\hat{g}_{t|t}, \hat{x}_{t|t} = \arg \max_{g_t, x} p_{\text{SLAM}}(g_t, x | y^t),$$

(3)

$$\hat{p}_{x|g}(z^t | g_t) \approx p(z^t | g_t = \hat{g}_{t|t}, x = \hat{x}_{t|t}, y^t) \approx p(z^t | y^t)$$

Otherwise the marginalization in (1) can be performed using samples from the SLAM system. Either way, omitting

\(^1\)Object pose is its position and orientation in world frame. With inertials, pose can be reduced to position and rotation around gravity. Sensor pose is full 6 degree-of-freedom position and orientation.

\(^2\)Attributes include sparse geometry (position in the inertial frame) and local photometry (feature descriptor, sufficient for local correspondence).
we will show how a

\[ \hat{p}(z|y^{t+1}) \propto p(y_{t+1}|z, \hat{g}_{t+1}u_{t}, \hat{x}_{t}) \hat{p}(z|y^{t}) \]  

(4)

where the likelihood term is approximated by a convolutional neural network (CNN) as shown in Sect. 3.3 and the posterior is updated by a Bayesian filter (BF) approximated by a bank of EKFs (Sect. 3.4). That only leaves the first factor \( p(\xi|z^{j}) \) in the posterior, which encodes context. While one could approximate it with a recurrent network, that would be beyond our scope here; we even forgo using the co-occurrence prior, which amounts to a matrix multiplication that rebalances the classes following \([13]\), since for the limited number of classes and context priors we experimented with, it makes little difference.

Approximating the likelihood in (4) appears daunting because of the purported need to generate future data \( y_{t+1} \) (the color of each pixel) from a given object class, shape and pose, and to normalize with respect to all possible images of the object. Fortunately, the latter is not needed since the product on the right-hand side of (4) needs to be normalized anyway, which can be done easily in a particle/mixture-based representation of the posterior by dividing by the sum of the weights of the components. Generating actual images is similarly not needed. What is needed is a mechanism that, for a given image \( y_{t+1} \), allows quantifying the likelihood that an object of any class with any shape being present in any portion of the image where it projects to from the vantage point \( g_{t} \). In Sect. 3.3 we will show how a discriminatively-trained CNN can be leveraged to this end.

3.3. Measurement Process

At each instant \( t \), an image \( I_{t} \) is processed by “probing functions” \( \phi \), which can be designed or trained to be invariant to nuisance variability. The SLAM system processes all past image measurements \( I^{t} \) and current inertial measurements \( u_{t} \), which collectively we refer to as \( y_{t} = \{ \phi_{h}(I_{t}), u_{t} \} \), where \( \phi_{h}(I_{t}) \) is a collection of sparse contrast-invariant feature descriptors computed from the image for \( N_{i} \) visible regions of the scene, and produces a joint posterior distribution of poses \( g_{t} \) and a sparse geometric representation of the scene \( x = [x_{1}, \ldots, x_{N_{i}(t)}] \), assumed uni-modal and approximated by a Gaussian:

\[ p_{\text{SLAM}}(g_{t}, x|y^{t}) \sim \mathcal{N}(g_{t}|\hat{g}_{t}, \hat{x}_{t} | P_{(g,x,t)}|) \]  

(5)

where \( x \in \bigcup_{j} s_{j} \), i.e., the scene is assumed to be composed by the union of objects, including the default class “background” \( l_{0} \). This localization pipeline is borrowed from \([63]\), and is agnostic of the organization of the scene into objects and their identity. It also restricts \( x \) to a subset of the scene that is rigid, co-visible for a sufficiently long interval of time, and located on surfaces that, locally, exhibit Lambertian reflection.

To compute the marginal likelihood for each class \( l_{k} \in \{l_{0}, \ldots, l_{K}\} \), we leverage on a CNN trained discriminatively to classify a given image region \( g_{j} \) into one of \( K + 1 \) classes, including the background class. The architecture has a soft-max layer preceded by \( K + 1 \) nodes, one per class, and is trained using the cross-entropy loss, providing a normalized score \( \phi_{\text{CNN}}([I_{t}]_{b_{j}}|k) \) for each class and image bounding box \( b_{j} \). We discard the soft-max layer, and forgo class-normalization. The activations at the \( K + 1 \) nodes in the penultimate layer of the resulting network provide a mechanism for, given an image \( I_{t} \), quantifying the likelihood of each object class \( l_{k} \) being present at each bounding box \( b_{j} \), which we interpret the (marginal) likelihoods for (at least an instance of) each class being present at the given bounding box:

\[ \phi_{\text{CNN}}([I_{t}]_{b_{j}}|k) \simeq p(I_{t}|l_{k}, b_{j}). \]  

(6)

This process induces a likelihood on object classes being present in the visible portion of the scene regions of \( s_{j} \) and corresponding vantage points \( g_{t} \), via \( b_{j} = \pi(g_{t}, s_{j}) \) where \( \pi \) is the projection. Since inertials \( u_{t} \) are directly measured, up to a Gaussian noise, we have:

\[ p(y_{t}|z^{j}, g_{t}, x) \simeq \phi_{\text{CNN}}([I_{t}]_{\pi(g_{t}, s_{j})}|k)N(\bar{u}; Q) \]  

(7)

where \( \bar{u} \) are the inertial biases and \( Q \) the noise covariance; here the object attributes \( z^{j} \) are the labels \( I_{j} = l_{k} \) and geometry \( s_{j} \). Thus, given an image \( I_{t} \), for each possible object pose and shape \( s_{j} \) and vantage point \( g_{t} \), we can test the presence of at least one instance of each class \( l_{k} \) within. Note that the visibility function is implicit in the map \( \pi \). If an object is not visible, its likelihood given the image \( I_{t} \) is constant/uniform. Note that this depends on the global layout of the scene, since the map \( \pi \) must take into account occlusions, so objects cannot be considered independently.

3.4. Dependencies and Co-visibility

Computing the likelihood of an object being present in the scene requires ascertaining whether it is visible in the image, which in turn depends on all other objects, so the scene has to be modeled holistically rather than as an independent collection of objects. In addition, the presence of certain objects, and their configuration, affects the probability that other objects that are not visible be present.\(^{3}\)

To capture these dependencies, we note that the geometric representation \( p(g_{t}, x|y^{t}) \) can be used to provide a joint distribution on the position of all objects and cameras \( p(g_{t}, x|y^{t}) \), which yields co-visibility information, specifically the probability of each point in \( x \) being visible by

\(^{3}\)For instance, seeing a keyboard and a monitor on a desk affects the probability that there is a mouse in the scene, even if we cannot see it at present. Their relative pose also informs the vantage point that would most reduce the uncertainty on the presence of the mouse.
any camera in \( g^t \). It is, however, of no use in determining visibility of objects, since it contains no topological information: We do not know if the space between two points is empty, or occupied by an object void of salient photometric features. To enable visibility computation, we can use the point cloud together with the images to compute the dense shape of objects in a maximum-likelihood sense: \( g_j = \arg \max p(s_j, g^t, x, y^t) \) using generic regularizers. This can be done but not at the level of accuracy and efficiency needed for live operation. An alternative is to approximate the shape of objects with a parametric family, for instance cuboids or ellipsoids, and compute visibility accordingly, also leveraging the co-visibility graph computed as a corollary from the SLAM system and priors on the size and aspect ratios of objects. To this end, we approximate

\[
\hat{p}_{g,x}(z^j | y^t) \approx p(z^j | y^t, g^t, x) \approx \prod_j p(z_j | y^t, g^t, x, z^{-j}) \tag{8}
\]

where \( z^{-j} \) indicates all objects but \( z_j \). Each factor \( p(s_j, l_j | y^t, g^t, x) \) is then expanded as the product

\[
p(s_j | l_j, y^t, g^t, x, s^{-j}) P(l_j | y^t, g^t, x, l^{-j}) \tag{9}
\]

where PMF indicates a probability mass filter; this effectively yields a bank of class-conditional EKFs. These provide samples from \( \hat{p}(z | y^t) \) in the right-hand side of (4), that are scored with the CNN to update the posterior.

4. Implementation Details

We have implemented two renditions of the above program: One operating in real-time and demonstrated live in June 2016 [18]. The other operating off-line and used for the experiments reported in Sect. 5. Fig. 2 sketches the system flow chart.

In both cases, we have taken some shortcuts to improve the efficiency of the approximation of the likelihood function implemented by a CNN. Also, the semantic filter needs initialization and data association, which requires some heuristics to be computationally viable. We describe such heuristics in order.

Visual Odometry and Baseline 2D CNN We use robust SLAM implemented from [63] to acquire sparse point clouds and camera pose \( x, g_t \) at each \( t \). This occurs in \( 10 \sim 20 \text{ms} \) per VGA frame. For the quantitative evaluation on KITTI, we use [46] as the underlying localization pipeline. For our real-time system, we use YOLO [48] as a baseline method to compute object likelihoods in \( 150 \sim 200 \text{ms} \), whereas in the off-line system we use Sub-CNN [70]. In either case, the result is, for each given window, a positive score for each class \( k \), read out from the penultimate layer. These are used both to compute the likelihood, and to generate proposals for initialization as discussed later.

Filter Organization Each object is represented by a PMF filter over class labels and \( K \) class-conditional EKFs, one for each class (9). Thus each object is represented by a mixture of \( K \) EKFs, some of which pruned as we describe later. Each maintains a posterior estimate of position, scale and orientation relative to gravity. The state predicts the projection of (each of the \( K \) instances of) each object onto the image plane, where the CNN evaluates the likelihood. For some object classes, we use a shape prior, enforced as a pseudo-measurement with uncertainty manually tuned to the expected class-variability. For instance, people are parallelepips of \( 1 \text{m}^3 \) expected volume with an anisotropic covariance along coordinate axes in the range of few decimeters, whereas couches have significantly more uncertainty.

Data Association To avoid running the baseline CNN multiple times on overlapping regions (each object is represented by multiple, often very similar, regions, one per each current class hypothesis), we do not query the CNN sequentially for each prediction. Instead, we run the CNN once, with lax threshold so as to obtain a large number of (low-confidence) regions. While this is efficient, it does create a data association problem, as we must attribute (possibly multiple) image regions to each (of multiple) object hypotheses, each of which has multiple possible class labels [4]. We avoid explicit data association by opting simple heuristics instead: first we generate predictions from the filter; then occluded objects are excluded from likelihood evaluation. For all others, we generate four-tuple coordinates of the bounding box, as a 4-dimensional Gaussian given the projection of the current state. This is a sloppy prediction, for the image of a parallelepiped is in general not an axis-aligned rectangle on the image. Nevertheless, we use this for scoring the use of the likelihood produced by the CNN for each predicted class. A (class-dependent) threshold is used to decide if the bounding box should be used to update the object. Bounding boxes with lower likelihood are given small weights in the filter update. This requires accurate initialization, which we will describe below. The silver lining is that inter-frame motion is usually
small, so data association proceeds smoothly, unless multiple instances of the same object class are present nearby and partially occlude each other.

**Initialization**  Putative 2D CNN detections not associated to any object are used as (bottom-up) proposals for initialization. The new object is positioned at the weighted centroid of the sparse points whose projections lie within the detection region. The weight at center is the largest and decreases exponentially outwards. Orientation is initialized as the “azimuth” from SubCNN, rotated according to camera pose and gravity. Given the position and orientation, scale is optimized by minimizing the reprojection error.

**Merge**  Objects are assumed to be simply-connected and compact, so two objects cannot occupy the same space. Yet, their projected bounding boxes can overlap. If multiple instances from the same object are detected, initialized and propagated, they will eventually merge when their overlap in space is sufficiently large. Only objects from the same class are allowed to merge as different classes may appear co-located and intersecting in their sloppy parallelepipedal shape model, e.g., a chair under a table.

**Termination**  Each object maintains a probability over $K$ classes, each associated with a class-conditional filter. If one of the classes becomes dominant (maximum probability above a threshold), all other filters will be eliminated to save computational cost. Most objects converge to one or two classes (e.g., chair, couch) within few iterations. Objects that disappear from view are retained in the state (short-term memory), and if not seen for a sufficiently long time, they are stored in long-term memory (“semantic map”) for when they will be seen again.

There are more implementation details that can be described in the space available. For this reason, we make our implementation publicly available at [1].

<table>
<thead>
<tr>
<th>Orientation error</th>
<th>Position error</th>
<th>$&lt; 0.5 \text{m}$</th>
<th>$&lt; 1 \text{m}$</th>
<th>$&lt; 1.5 \text{m}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>method</td>
<td>#TP</td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>$&lt;$ 30°</td>
<td>Ours-FNL</td>
<td>150</td>
<td>0.14</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td>Ours-INST</td>
<td>135</td>
<td>0.13</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>SubCNN</td>
<td>99</td>
<td>0.10</td>
<td>0.07</td>
</tr>
<tr>
<td>$&lt;$ 45°</td>
<td>Ours-FNL</td>
<td>157</td>
<td>0.15</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>Ours-INST</td>
<td>141</td>
<td>0.13</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td>SubCNN</td>
<td>99</td>
<td>0.10</td>
<td>0.07</td>
</tr>
<tr>
<td>~</td>
<td>Ours-FNL</td>
<td>169</td>
<td>0.16</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>Ours-INST</td>
<td>149</td>
<td>0.14</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td>SubCNN</td>
<td>104</td>
<td>0.10</td>
<td>0.07</td>
</tr>
</tbody>
</table>

Table 1. Quantitative evaluation on KITTI and comparison with SubCNN [70]. The number of true positives having positional error (row), and angular error (column) less than a threshold is shown, along with Precision and Recall. Scores are aggregated across all 3501 ground-truth labeled frames in the dataset, with 498 annotated objects. The last 3 rows discard orientation error.

As explained in Sec. 2, we choose SubCNN [70] as the paragon, even though it is based on a single image, because it is the top performer for 3D recognition in KITTI among non-anonymous and reproducible ones, in particular it dominates [10]. Being single-image based, SubCNN returns different results in each frame, therefore naturally at a disadvantage. To make the comparison fair, one would have to average or integrate detections for each object across all frames when it is visible. However, SubCNN does not provide data association, making direct comparison challenging. To make comparison as fair as possible, without developing an alternate aggregation method for SubCNN, we compare it to our algorithm on a frame-by-frame basis. Specifically, for each frame, we transfer the ground truth to the camera frame, and remove occluded objects. Then we can compare detections from SubCNN to our point estimate (conditional mean) computed causally by the filter at the current time. We call this method Ours-INST. On the other hand, we can benefit from aggregating temporal information for as long as possible, so we also report results based on the point-estimate of the filter state at the last time instant when each object is seen. The estimate is then mapped back to the current frame, which we call Ours-FNL. To the best of our knowledge, there are no known methods for 3D recognition that causally update posterior estimates of object identity/presence and geometric attributes, and even naive temporal averaging of a method like [70] is not straightforward because of the ab-

5. Experiments
5.1. Quantitative Results

![Image 310x472 to 367x524](image1)

![Image 369x472 to 426x524](image2)

![Image 428x472 to 485x524](image3)

Figure 3. Qualitative comparison with SubCNN. Top: Images with back-projected objects from our method (Green), the same with SubCNN (Yellow). Bottom: top-view of the corresponding portion of the scene. Ground truth is shown in Blue.
Figure 4. Evolution of the state (Green) against ground-truth annotation (Blue) (best viewed at 5×, images shown at the top for ease of reference). When first seen (Leftmost) cars ‘A’ and ‘B’ are estimated to be side-by-side; after a few frames, however, ‘A’ and ‘B’ fall into place, but a new car ‘C’ appears to flank ‘B’. As time goes by, ‘C’ too falls into place, as new cars appear, ‘D’, ‘E’, ‘F’. The error in pose (position and orientation) relative to ground truth can be appreciated qualitatively. Quantitative results are shown in Table 1.

5.1. Dataset

There are many datasets for image-based object detection [19, 51] which provide 2D ground truth. There are also 3D object detection datasets [71], most using extra sensor data, e.g., depth from a structured-light sensor. None provide inertial measurements, except KITTI [22], whose object detection benchmark contains 7181 images, from which we exclude 3682 frames used for SubCNN training [70], leaving us a validation set of 3799 frames. We then find 10 videos which cover most of the validation set. After removing moving objects, 498 objects are observed 18468 times at 3501 instants, which is the same order of magnitude of the 2D validation set.

5.1.2 Evaluation Metrics

KITTI provides ground-truth object tracklets we use to define true positives, miss detections and false alarms. A true positive is the nearest detection of a ground truth object within a specified error threshold in both position and orientation (Table 1). A miss occurs if there is no detection within the threshold. A false alarm occurs when an object is detected despite no true object being within the threshold in distance and orientation. Precision is the fraction of true positives over all detections, and Recall is the percentage of detected instances among all true objects.

5.1.3 Benchmark Comparison

Table 1 shows result on the KITTI dataset, averaged over all sequences. On average, Ours-INST already outperforms SubCNN even if our initialization can be rather inaccurate. Note that our method requires evidence to be accumulated over time before claiming the existence of an object in the scene, so Ours-INST is penalized heavily in the first few frames when a new object is spotted. Ours-FNL further improves the results by a large margin. Fig. 4 shows how our method refines the state over time. Visual comparison is shown in Fig. 3 for ground truth (Blue), Ours-FNL (Green) and SubCNN (Yellow).

5.2. Class-specific Priors

Objects have characteristic scales, which are lost in perspective projection but inferable with an inertial sensor. We impose a class-dependent prior on size and shape (e.g., volume, aspect ratios). In Fig. 5, a toy car is detected as a car by an image-based detector (Yellow), but rejected by our system as inconsistent with the scale prior (Green). Fig. 5(b) shows two background cars in the far field, whose images are smaller than the toy car, yet they are detected correctly, whereas the toy car is rejected.

5.3. Occlusion and Memory

Our system represents objects in the state even while they are not visible, or detected by an image-based detector. This allows predicting the re-appearance of objects in future frames, and to resume update if new evidences appear. Fig. 6 shows a chair first detected and then occluded by a monitor, later reappearing. The system predicts the chair to be completely occluded, and therefore does not use the image to update the chair, but resumes doing so when it reappears, by which time it is known to be the same chair that was previously seen (re-detection). In Sect. 5.4, we show the same phenomenon in a large-scale driving sequence.
5.4. Large-scale Driving Sequences

Fig. 1 and online video at [1] show our results on a 3.7km-long sequence from KITTI. It contains hundreds of cars along the route. Once recognized as a car, we replace the bounding box with a CAD model of similar car, aligned with the pose estimate from the filter, in a manner similar to [52], that however uses RGB-D data. In this sequence, we can also see cars on different streets “through walls” if they have been previously detected, which can help navigation.

5.5. Indoor Sequences

We have tested our system live in a public demo [18], operating in real time in cluttered environments with people, chairs, tables, monitors and the like. Representative examples are shown for simpler scenes, for illustrative purposes, in Fig. 7, where again CAD models of objects are rendered once detected, a’ la [52]. Our system does not produce exact orientation estimates, as seen in Fig. 7, so there is plenty of room for improvement.

6. Discussion

Inertial sensors are in every modern phone, tablet, car, even many toys, all devices embedded in physical space and occasionally in need to interact with it. It makes sense to exploit inertials, along with visual sensors, to help detecting objects that exist in 3D physical space, and have characteristic shape and size, in addition to appearance. We have recorded tremendous progress in object detection in recent years, if by object one means a group of pixels in an image. Here we leverage such progress to design a detector that follows the prescriptions (a)-(e) indicated in the introduction.

We start by defining a representation as a minimal sufficient invariant statistic of object attributes, in line with [59]. We then marginalize on camera Euclidean pose – which allows us to enforce priors on the class-specific scale of objects – and update the measure by a Bayesian filter, where a CNN is in charge of computing the likelihood function.

We note that a minimal sufficient invariant for localization is an attributed point cloud, and therefore there is no need to deploy the machineries of Deep Learning to determine camera pose (Deep Learning could still be used to infer the attributes at points, which are used for correspondence). Instead, we use an Extended Kalman Filter, conditioned on which the update for object attributes can be performed by a Mixture-of-Kalman filter.

The result is a system whereby objects do not flicker in-and-out of existence, our confidence in their presence grows with accrued evidence, we know of their presence even if temporarily occluded, we can predict when they will be seen, and we can enforce known scale priors to reject spurious hypotheses from the bottom-up proposal mechanism.

We have made stringent and admittedly restrictive assumptions in order to keep our model viable for real-time inference. One could certainly relax some of these assumptions and obtain more general models, but forgo the ability to operate in real time.

The main limitation of our system is its restriction to static objects. While in theory the framework is general, the geometry of moving and deforming objects is not represented, and therefore their attributes remain limited to what can be inferred in the image. Also, our representation of objects’ shape is rather rudimentary, and as a result visibility computation rather fragile. These are all areas prime for further future development.

Our datasets, consisting of monocular imaging sequences with time-stamped inertial measurements, are available at [1], along with our system implementation.

Acknowledgments

References


