Scene Shape from Texture of Objects

Nadia Payet and Sinisa Todorovic
Oregon State University
Corvallis, OR 97331, USA
payetn@onid.orst.edu, sinisa@eecs.oregonstate.edu

Abstract

Joint reasoning about objects and 3D scene layout has shown great promise in scene interpretation. One visual cue that has been overlooked is texture arising from a spatial repetition of objects in the scene (e.g., windows of a building). Such texture provides scene-specific constraints among objects, and thus facilitates scene interpretation. We present an approach to: (1) detecting distinct textures of objects in a scene, (2) reconstructing the 3D shape of detected texture surfaces, and (3) combining object detections and shape-from-texture toward a globally consistent scene interpretation. Inference is formulated within the reinforcement learning framework as a sequential interpretation of image regions, starting from confident regions to guide the interpretation of other regions. Our algorithm finds an optimal policy that maps states of detected objects and reconstructed surfaces to actions which ought to be taken in those states, including detecting new objects and identifying new textures, so as to minimize a long-term loss. Tests against ground truth obtained from stereo images demonstrate that we can coarsely reconstruct a 3D model of the scene from a single image, without learning the layout of common scene surfaces, as done in prior work. We also show that reasoning about texture of objects improves object detection.

1. Introduction

Scene interpretation is a long-standing, basic problem in computer vision. Recent work demonstrates that a synergistic treatment of diverse image-understanding tasks, including object recognition, image segmentation, and 3D-scene reconstruction, may overcome many errors induced by addressing them in isolation [10, 2, 9, 7, 20, 6, 16]. These approaches typically fuse object detections with supervised priors of spatial layouts of common scene surfaces (e.g., the sky is on the top, and the ground is planar and horizontal) .

While holistic scene intrepretation shows great promise, the treatment of the 3D scene layout in existing work has certain shorcomings. First, they make the restrictive assumption that surfaces in the scene are planar, and discretize surface orientations into a pre-specified number of classes (e.g., buildings may face only left, right, or front). Second, they typically estimate surface orientation classes too locally (e.g., per each superpixel), without accounting for the long-rage spatial relations among image parts. This may easily lead to implausible 3D layouts.

One visual cue that has been overlooked, and that could address the aforementioned shortcomings of prior work, is texture arising from a spatial repetition of objects in the scene. In general, textures of recurring objects are ubiquitous. For example, windows on a building facade jointly give the percept of window texture, and a sequence of cars parked along a street gives rise to car texture, as illustrated in Fig 1. In a cafeteria scene, tables and chairs, and people standing in a line comprise many distinct textures. Also, in natural scenes, one can easily find textures corresponding to flocks of birds, herds of animals, or tree lines.

In this paper, we focus on scenes where thickness and depth differences of spatially repeating 3D objects are much smaller than their distance from the camera. Thus, these objects can be interpreted as texture elements lying on a surface's tangent plane at a point. Given distinct textures of objects in an image, we estimate the 3D shape of their surfaces via shape-from-texture. We use the estimated 3D scene model to help detect and localize all object occurrences, and thus enable potential identification of new textures in the scene. We iterate these steps until obtaining a coherent scene interpretation in which the world is not composed of blocks in discrete, pre-specified depth and orientation arrangements, as in existing work, but rather of more realistic 3D shapes, as illustrated in Fig 1. We achieve this without supervised learning of 3D scene layouts.

Recent work [2] also uses object detections to estimate their supporting surfaces. However, they make the restrictive assumptions that the supporting surfaces are planar, and parallel. Also, they have access to training examples of object poses seen from all viewpoints. We relax their assumptions, and do not use 3D models of objects.

Our evaluation on street scenes demonstrates that rea-

soning about texture of objects facilitates holistic scene interpretation. This is because texture provides scene-specific constraints among objects, which we use to relax the aforementioned restrictive assumptions of prior work. Our key contributions include a new approach to scene interpretation, based on shape-from-texture, and an efficient sequential inference procedure for texture detection.

2. Overview

The Problem: Given an image, our goal is to: (1) Detect and localize all occurrences of target object classes; (2) Identify distinct textures whose texture elements are instances of these classes; (3) Reconstruct a 3D model of the identified texture surfaces, and appropriately place the detected objects in the reconstructed 3D scene.

To address the above problem, we need to account for long-range spatial relations in the image. The graphical-modeling framework, common in related work (e.g., CRF [6]), seems unsuitable here. Specifically, a graphical model would need to encode higher-order cliques, and thus face serious tractability issues in learning and inference. Instead, we use a simpler "interpretation by synthesis" approach, where image regions are sequentially explained, starting from confident regions to guide the interpretation of other regions. This is similar to [7]. They pick in each iteration four regions that maximize a heuristic score of the scene interpretation. By contrast, we seek to learn this scoring function using reinforcement learning (RL) [14]. RL is particularly well-suited in our case, because it finds an optimal policy that maps *states* of an environment (detected objects and reconstructed surfaces) to *actions* that an *agent* ought to take in those states (detecting new objects and identifying new textures), so as to minimize a long-term loss.

The main steps of our approach are shown in Fig. 1.

Step 1: Given an image, we detect objects of interest using the state-of-the-art, latent-SVM detector of [5]. The detector represents an object class by six models, corresponding to six distinct object poses, as illustrated in Fig. 2. For each pose, the respective model encodes the canonical 2D locations and scales of 8 object parts. We associate with each object pose the expected value of its surface normal, N. When an object is detected, we take the following detector outputs: confidence, bounding box, relative locations and scales of 8 object parts, and the model responsible for detection (i.e., 3D pose). For every object detection, a difference between the detected and canonical locations and scales of object parts relative to the bounding box is used to estimate the amount of their spatial deformations.

Step 2: For texture detection, we use the model of texture as a marked point process [15, 8, 17, 18, 19]. A surface is textured by statistically marking its points, and placing statistically similar objects (i.e., texture elements) at these points. Given candidate object detections from Step 1, we identify one texture at a time by tracking instances of the same object class, similar to [15, 8, 17]. To this end, we use an RL-based labeler that sequentially visits object detections, and labels them as being a part of texture or non-texture. These decisions are informed by Gestalt grouping cues. After detecting a texture, all of its texture elements are removed from the pool of object detections, and the RL-based labeling is run again. This is iterated until all remaining object detections are labeled as non-texture.

Step 3: The identified textures are used for shape-from-texture [18, 19]. Intuitively, texture elements may be slanted away from the camera, causing foreshortening, and lie at different distances from the camera, resulting in a change of scale. We relate the relative locations and sizes of 8 parts of a detected texture element to their canonical values by an affine homography. The affine homography determines the surface normal at that surface point. We apply diffusion to the estimated set of surface normals, and thus reconstruct the 3D texture surfaces. The surfaces corresponding to image parts classified as non-texture are reconstructed by defusing the normals of points along the boundaries that the non-texture image regions share with the textured ones. This ultimately gives a 3D model of the scene.

Step 4: We repeat Steps 2–3 until the resulting scene interpretation reaches equilibrium.

The remainder of the paper presents details of each step of our approach, starting from Step 2.

3. Identifying Image Textures

Given responses of the object detector (Step 1), we identify elements of a texture as a sequential assignment of binary labels to every object detection. The label is 0 for non-texture, and 1 for part of texture. Let $\boldsymbol{X}=(\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n)\in\mathcal{X}$ and $\boldsymbol{Y}=(y_1,\ldots,y_n)\in\mathcal{Y}$ denote a sequence of descriptor vectors $\boldsymbol{X}_i$ associated with detections, and the corresponding sequence of their labels $y_i \in \{0,1\}$, which are obtained in steps $t = 1,\ldots,n$. Our goal is to learn the structured prediction $f : \mathcal{X} \rightarrow \mathcal{Y}$ on available training images, and use f to identify distinct textures in a new image.

We formulate the sequential labeling of objects (SL) within one of the latest RL frameworks, called SEARN [13]. SEARN integrates search and learning for solving complex structured prediction problems. It transforms RL into a classification problem, and shows that good classification performance entails good RL performance. It has been extensively evaluated in [13], and it compares favorably to other techniques for structured prediction. In SEARN, at every time step, a current state of the environment is represented by a vector of observable features. This vector is input to a classifier to predict the right action. Thus, learning the optimal policy within SEARN amounts to learning a classifier over state feature vectors, so as to minimize a loss. Below, we first review SEARN, and show

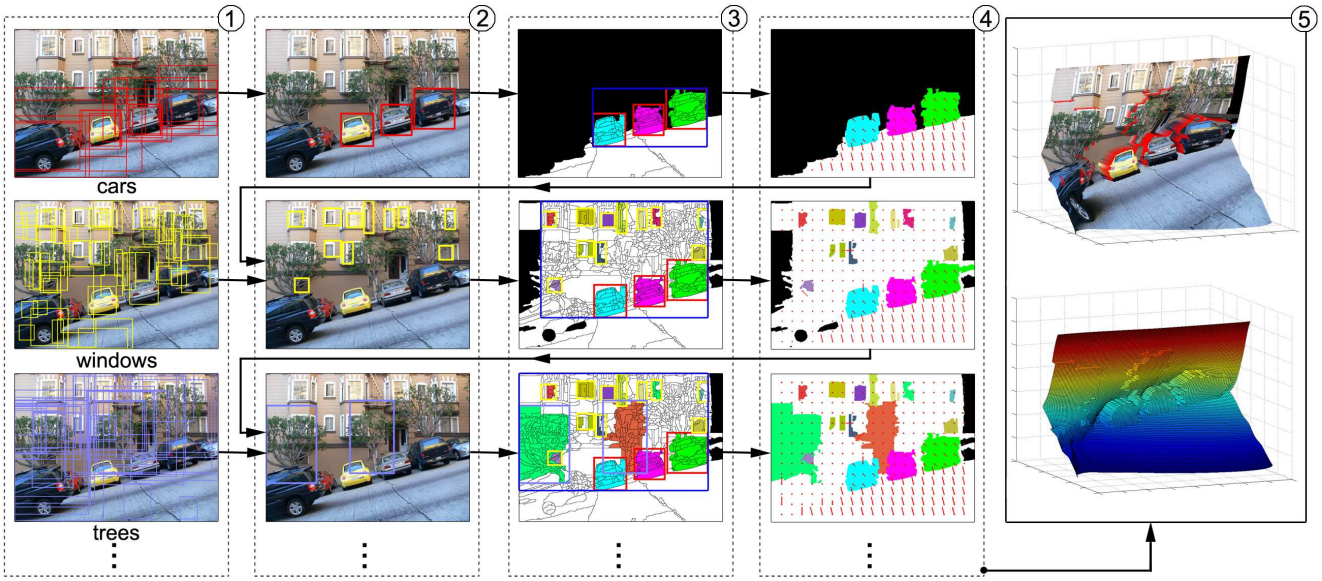
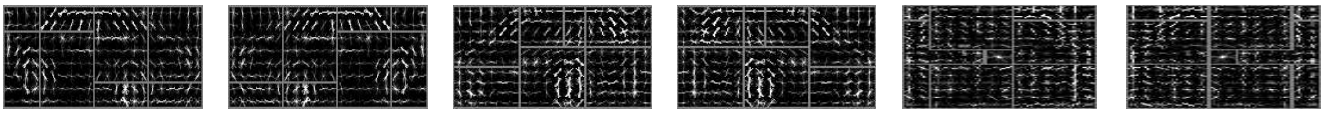


Figure 1. Overview of our approach. (1) Detections of the latent SVM detector [5] for cars, windows, trees, etc. (2) Most of the correct bounding boxes are selected by SL, which uses both the 2D and the 3D structure, when available. (3) A surface normal is estimated for each object, and all regions in the corresponding bounding box are assigned that particular normal (colored regions), then a diffusion process interpolates the normals in the zone of influence of the objects, represented by the white regions. (4) The interpolated normals. (5) The surface reconstruction. Our system correctly estimates that the ground is slanted and that the building is front-facing. This cannot be handled by existing approaches, that typically assume that the ground surface is flat.



Model 1 $N{=}[0,0,1]$ | Model 2 $N{=}[0,0,1]$ | Model 3 $N{=}[\frac{\sqrt{2}}{2},0,\frac{\sqrt{2}}{2}]$ | Model 4 $N{=}[-\frac{\sqrt{2}}{2},0,\frac{\sqrt{2}}{2}]$ | Model 5 $N{=}[0,0,1]$ | Model 6 $N{=}[0,0,1]$

Figure 2. The latent-SVM detector of [5]. An example for the cars. The detector consists of 6 models encoding 6 car poses with canonical surface normals N. Each model consists of 8 object parts. We use fewer models for the other object classes (e.g., only 1 for the trees).

how it is trained to fit our particular vision problem.

SEARN applies classifier f (e.g. SVM, or Decision Tree) to a sequence of data samples, $\boldsymbol{X}{\in}\mathcal{X}$, to infer their labels $\boldsymbol{Y}{\in}\mathcal{Y}$. It requires that the ordering of instances in $\boldsymbol{X}$ be well-defined. SEARN uses an iterative batch-learning. Specifically, in each iteration τ, the results of classification, $f^{(\tau)}{:}\boldsymbol{X}{\rightarrow}\boldsymbol{Y}^{(\tau)}$, are compared with the ground-truth labels, $\hat{\boldsymbol{Y}}$. This induces loss $L(\boldsymbol{Y}^{(\tau)},\hat{\boldsymbol{Y}})$, which is then used to learn a new classifier $h^{(\tau+1)}$. In the next iteration, SEARN applies $f^{(\tau+1)}$ to $\boldsymbol{X}$, where $f^{(\tau+1)}$ is defined as:

$$f^{(\tau+1)} = \beta h^{(\tau+1)} + (1-\beta) f^{(\tau)}, \qquad (1)$$

where $\beta \in (0,1]$ is the interpolation constant. This interpolation amounts to a probabilistic sampling of the iteratively learned classifiers $h^{(1)}, h^{(2)}, \ldots, h^{(\tau+1)}$. The classifier sampling is governed by the multinomial distribution, where, from (1), the probability of selecting classifier $h^{(k)}$ in iteration τ is $\alpha_\tau^{(k)} = \beta(1-\beta)^{\tau-k}$, $k = 1,\ldots,\tau$. After τ reaches the maximum allowed number of iterations, T, the output is the last policy $f^{(T)}$ from which $h^{(1)}$ is removed, i.e., the output is $\{h^{(2)},\ldots,h^{(T)}\}$ and their associated sampling probabilities $\{\alpha_T^{(2)},\ldots,\alpha_T^{(T)}\}$. Performance bounds of SEARN are presented in [13].

We accommodate SEARN for our problem by specifying: (i) Object descriptors that define data samples $\boldsymbol{X}$; (ii) Ranking function R, which provides an ordering of $\boldsymbol{X}$; and (iii) Loss function L for the iterative learning of policy f. These specifications are presented in the sequel. They together define SL, summarized in Alg. 1.

A Descriptor of Object Detections. Our key idea is to compute $\{\boldsymbol{X}_i\}$ online, from the cues of image parts that have already been explained. $\boldsymbol{X}_i^{(t)}$ is a descriptor that consists of intrinsic object properties, and its pairwise spatial relations with those objects that have been labeled as texture in the previous t steps. The intrinsic object properties, $\boldsymbol{\psi}_i$, include: (a) the detector confidence; (b) the model of

the object pose that was used for detection; (c) 2D location and scale of the object, (c_i, s_i), normalized w.r.t. the image; and (d) 2D location and scale of the object parts, normalized w.r.t. the object's bounding box. The pairwise properties, $\boldsymbol{\phi}_{ij}^{(t)}$, include: (e) overlap of the bounding boxes, $\frac{b_i \cap b_j}{b_i \cup b_j}$; (f) displacement $|c_i - c_j|$; (g) scale ratio $\frac{s_i}{s_j}$; and (h) spatial relation between the bounding boxes b_i and b_j whose value can be far, near, above, below, on-top, or next-to, as in [4]. Note that $\boldsymbol{\phi}_{ij}^{(t)}$ can provide evidence of perceptual grouping of objects into texture. Whether the grouping actually occurs at object i has to be inferred by SL. Thus, at a given step t of sequential labeling, we have $\boldsymbol{X}_i^{(t)} = [\boldsymbol{\psi}_i,\ [\boldsymbol{\phi}_{ij}^{(t)}, j = 1, 2...]]$.

Ranking Function R**.** At every step t, SEARN uses a ranking function to label the next object, such that its labeling reduces uncertainty about the other objects in the image. R is specified as the confidence of classifiers $h^{(\tau)}$. At t, descriptors $\boldsymbol{X}_i^{(t)}$ of all unlabeled objects are updated based on the current state, and then classified. R selects $\boldsymbol{X}_i^{(t)}$ with the highest confidence in classification.

Loss Function L**.** L is defined as the overlap error between bounding boxes b of objects that are labeled as texture and the ground truth bounding boxes $\hat{b}$. We pair bounding boxes b_i and $\hat{b}_i$ with the largest overlap. L is a sum of the overlap errors, $L(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = 1 - \frac{1}{t}\sum_{i=1}^{t} \frac{b_i \cap \hat{b}_i}{b_i \cup \hat{b}_i}$.

Algorithm 1: Learning SL

Input : Set of training images $\mathcal{I} = \{I_1, I_2 \dots\}$;
Candidate objects $\{V(I_1), V(I_2), \dots\}$;
Ground-truth labels $\{\hat{\boldsymbol{Y}}(I_1), \hat{\boldsymbol{Y}}(I_2), \dots\}$;
Loss-sensitive classifier h, and initial $h^{(1)}$;
Loss function L; Interpolation constant $\beta = 0.1$;
Maximum number of iterations T

Output : Learned policy $f^{(T)}$

Initialize: $V_{\text{un}}^{(1)} = V$;
for $\tau = 1, \dots, T$ **do**
  Initialize the set of descriptor sequences $\mathcal{X} = \emptyset$;
  for *all* $I \in \mathcal{I}$ **do**
    $V = V(I)$; $n = |V(I)|$; $\hat{\boldsymbol{Y}} = \hat{\boldsymbol{Y}}(I)$;
    for $t = 1, \dots, n$ **do**
      for $i \in V_{\text{un}}^{(t)}$ **do**
        Compute $\boldsymbol{X}_i^{(t)}$;
        Compute $y_i = f^{(\tau)}(\boldsymbol{X}_i)$ as in (1);
      end
      Select i from $V_{\text{un}}^{(t)}$ with max confidence in y_i;
      Add y_i to $\boldsymbol{Y}^{(\tau)}$;
      $V_{\text{un}}^{(t)} \leftarrow V_{\text{un}}^{(t)} \setminus \{i\}$;
    end
    Estimate loss $L(\boldsymbol{Y}^{(\tau)}, \hat{\boldsymbol{Y}})$;
    Add the estimated descriptor sequence $\boldsymbol{X}$ to $\mathcal{X}$;
  end
  Learn a new classifier $h^{(\tau+1)} \leftarrow h(\mathcal{X}; L)$;
  Interpolate: $f^{(\tau+1)} = \beta h^{(\tau+1)} + (1 - \beta) f^{(\tau)}$
end
Return $f^{(T)}$ without $h^{(1)}$.

4. Reconstructing 3D Scene Layout

Deformations of texture elements from the known canonical pose can be used to estimate the underlying 3D shape of the texture surface. To this end, we assume that objects labeled as texture elements have planar parts. Then, we estimate the 3D pose of each part using an affine homography. Since parts are smaller than objects, and much smaller than surfaces, the reconstructed texture surfaces are piecewise planar.

For all parts of an object labeled as texture, we relate the detected part locations and scales with those of the canonical object pose through an affine homography. Let H_i be the homography of object i from its canonical pose to its pose in the image. We use the part centers to specify an overdetermined linear system of equations to calculate H_i with 6 degrees of freedom. After finding H_i, we compute the normal of i as $N_i = H_i N$, where N is the known canonical normal from the latent-SVM detector of [5] (see Fig. 2). N_i is further mapped to the 8 normals of individual object parts N_{ik}, $k = 1, ..., 8$, using the 8 homographies, known from the latent-SVM detector, between the reference canonical object pose and the planes of each object part.

The 3D texture surfaces can be reconstructed from the set of estimated surface normals $\{N_{ik} : i = 1, ..., n; k = 1, ..., 8\}$ of n texture elements by standard linear diffusion. The accuracy of this reconstruction depends on the number of estimated normals and their layout. The diffusion over the entire image, however, yields an over-smoothed 3D model. We address this by estimating the spatial support of detected textures, and then conducting the diffusion only within each region of support. Specifically, we segment the image with the state-of-the-art segmenter of [1]. All resulting segments that overlap with the objects of a specific texture are taken to form the spatial support of that texture. We linearly diffuse the estimated normals of object parts within the region of support of each texture. Figs. 1, 4 and 6 show examples of the needle plot obtained by this method.

The spatial support of non-texture surfaces is defined by the remaining segments that have not be assigned to any textures. Non-texture surfaces are reconstructed by deffusing, within the corresponding non-texture spatial support, the normals of points along the boundaries that the non-texture regions share with the textured ones. This ultimately gives a 3D model of the scene.

Closing the loop. After the initial reconstruction of the 3D scene in Steps 1–3 of our approach, we continue repeating Step 2 and Step 3 until the resulting scene interpretation reaches equilibrium.

5. Results

For evaluation, we use street scenes that abound with various textures. In particular, we are interested in textures of cars lined-up along the streets, windows on building facades, and pedestrians and trees on the sidewalks. The number of these textures in each image is not known.

Datasets. We use two datasets for evaluation. First, we query images from the LabelMe dataset [21] with the keyword 'building+window+car'. LabelMe images with less than 3 cars, or less than 3 windows are removed. This gives a dataset of 316 images, where 166 images are used for training, and the remaining 150 for testing. Note that our dataset of 316 LabelMe images is larger than the geometric-context dataset (GCD) [11] used as the benchmark by existing holistic approaches to scene interpretation. The GCD has only 16 images with object repetition, and thus is poor for evaluating our structure-from-texture method. Of course, this is a limitation of the benchmark GCD, and does not mean that scenes with spatially recurring objects are rare. Second, we use the stereo images of the Leuven Moving Vehicle Sequence [3]. From this sequence, we remove images that do not show at least 3 instances of cars or windows. This gives a dataset of 72 images, all of which are used for testing.

Training setup. Randomly selected 166 images of the LabelMe dataset are used for training the sequential labeler SL. For each image, we first detect candidate bounding boxes, using the detector of [5]. Bounding boxes that comprise distinct textures in the image are labeled with 1, and the remaining boxes are labeled with 0. We train a total of 10, C4.5 decision-tree classifiers, pruned with confidence factor $C = 0.25$, on these labeled bounding boxes, as summarized in Alg. 1.

Testing setup. Given a test image, we run the car, window, tree, and pedestrian detectors of [5] with a low detection threshold $\tau = -3$, so as to achieve high recall. Next, we use SL to detect all textures of objects present, one at a time, until no object detection can be labeled as belonging to texture. The surface normals of parts of all identified texture elements are estimated via an affine homography of their known canonical poses. 3D shapes of the texture surfaces are reconstructed by linear diffusion of these surface normals, within the spatial support of each texture, where the support is estimated using the segmenter of [1] with parameter $P_b = 10$, as described in Sec. 4. We assume that the identified distinct textures of windows correspond to distinct building surfaces in the scene. Also, we assume that cars, pedestrians, and trees are supported by the ground. Hence, image regions located below the detected bounding boxes of cars, pedestrians, and trees are defined as ground regions. Surface normals of the ground regions are specified as perpendicular to the estimated surface normals of cars, pedestrians, and trees. We linearly diffuse these surface normals of the ground regions to reconstruct a 3D shape of the ground. In this way, we relax the common assumption of prior work that the ground is planar and horizontal. The remaining non-texture surfaces are reconstructed by defusing the normals of points along the boundaries that the non-texture image regions share with the textured ones. For better visualization of the resulting 3D model of the scene, we place the detected cars, pedestrians, and trees in front of the reconstructed building surfaces, at some ad hoc distance δ, in the direction of the objects' normals.

Qualitative results. Fig. 4 shows our scene reconstruction results on examples from the LabelMe dataset. As can be seen, in the top row, we are able to extract details of the curvy facade, circled in red, and enlarged in Fig. 5(left). In the bottom row, we accurately reconstruct the uphill street, not as a horizontal surface, circled in red, and enlarged in Fig. 5(right). These two results contrast much prior work that typically allows only planar building surfaces, and restricts the ground surface to be horizontal (e.g., [7]).

Fig. 6 compares our surface layout estimates to that of the state-of-the-art approach, presented in [7], on a few images from the benchmark GCD. Note that it is difficult to make this qualitative comparison exactly "apples-to-apples". Nevertheless, we believe that Fig. 6 shows important insights. While [7] does not use object detectors as we do, they employ a battery of other detectors that we do not use. For example, they take as input responses of the surface-layout detector of [12], the sky and ground detectors, as well as the light-medium-heavy density detector. In [7], image regions are assigned one of the following labels: ground, and vertical facing-left, facing-right, frontal, porous, or solid. For fair comparison, we discretize our results into one of these classes, as follows. Each car and pedestrian region detected by our approach is automatically labeled as vertical solid. Similarly, tree regions detected by our approach are labeled as vertical porous. For each remaining region, we average its surface normals, and label the region as ground or vertical based on the resulting average normal. For each vertical region, we compute the angle α between the average normal and the z-axis (i.e., the estimated viewing direction) to determine the region's sub-class: frontal, facing-left or facing-right. The top row of Fig. 6 shows that [7] merges two buildings with opposite orientations as facing-left (cyan), whereas we correctly classify the building on the left as facing-right (magenta). We also correctly label the cars and pedestrians regions as solid, and the pavement regions as ground, in the bottom row of Fig. 6.

Quantitative results. We evaluate SL on the task of object detection. We use the VOC challenge evaluation criteria: precision and recall are obtained for bounding boxes of the detected objects, average precision (AP) is computed over the entire test set. Tab. 1 shows that SL improves

Method	Car	Window	Tree	Ped.	All
[5](1)	0.574	0.418	0.521	0.592	0.526
[5](2)	0.812	0.543	0.678	0.878	0.727
[4]	0.824	0.617	0.680	0.881	0.807
SL	**0.871** $\pm$**0.018**	**0.793** $\pm$**0.012**	**0.719** $\pm$**0.014**	**0.897** $\pm$**0.021**	**0.820** $\pm$**0.011**

Table 1. LabelMe dataset: Average Precision (AP) of our detection for cars, windows, trees and pedestrians. SL improves the state of the art detector of [5] when we use a low detection threshold $\tau = -3$ (1), and when we use the learned detection threshold (2). SL also outperforms the CRF method of [4].

Method	[10]	**Ours**
Surface layout	64.5%	**72.1%**$\pm$**2.7%**

Table 2. LabelMe dataset: Surface layout classification accuracy over the vertical subclasses: frontal, facing-right and facing-left. Our approach outperfoms the state of the art technique.

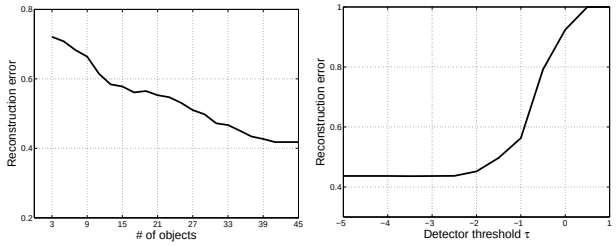


Figure 3. Leuven dataset: 3D reconstruction error as a function of (left) the number of correctly detected objects and (right) the object detector's threshold τ.

by 30.6% the average precision of the low-precision-high-recall detector of [5] used with the detection threshold set to $\tau = -3$. We also see that using the information about 3D spatial layout improves by 9.3% the precision of [5] in its standard form, i.e., when τ is learned in training.

We also compare the average precision of SL with that of the CRF-based method of [4]. This is a fair comparison, since [4] also uses the object detector of [5]. Tab. 1 shows that SL outperforms [4] for all object classes. This could be, because we incorporate 3D layout information in our object detection that is richer than the 2D spatial constraints used in [4].

For surface layout estimation, we compare against the state of the art approach of [10]. We are not able to compare with [7], since their code requires inputs from detectors that are currently not public. We evaluate our surface classification results over regions labeled as vertical, where the classification is done as described above for the results presented in Fig. 6. Table 2 shows that our classification accuracy is significantly larger than that of [10] on the LabelMe images.

We also use 72 stereo pairs of images from the Leuven dataset to quantitatively evaluate our 3D reconstruction. In particular, we reconstruct a 3D model of the scene using the standard stereo approach of [22]. From this 3D model, we compute surface normals at each pixel, and take these normals as ground truth. The ground-truth normals are compared with our reconstructed normals, obtained using only one of the two stereo images. We define the reconstruction error as the average Euclidean distance between ground-truth and reconstructed normals. On the 72 images of the Leuven sequence, we obtain an average reconstruction error of $43.7\% \pm 2.4\%$. In Fig. 3(left), we analyze the influence of the number of correctly detected objects on the reconstruction error. As expected, the error decreases as the number of objects increases, since the accuracy of the estimated normals is directly proportional to the number of objects. In Fig. 3(right), we analyze the influence of the detector's threshold on the reconstruction error. For low thresholds, the error does not change much, but it quickly increases as the threshold gets larger than -2. This indicates that our approach requires an object detector with high recall.

SL is data-driven and typically selects to label textures in the ordering cars-windows-trees-pedestrians. We evaluate a variant of our approach where we force SL to have the following two orderings cars-windows-pedestrians-trees and windows-cars-pedestrians-trees. These forced orderings give worse 3D reconstruction performance by $3.4\%\pm0.05\%$ and $5.9\% \pm 0.08\%$ resp. Other combinations produce worse results. By the nature of our images, there are more cars and windows than there are pedestrians or trees, which explains why the reconstruction is better when one of these two classes comes first in the ordering. Indeed, the more objects of a particular class are detected in the scene, the more accurate the reconstruction of its supporting surface, see Fig. 3(left). The estimate of the ground surface is critical for pruning false alarms, which is why the combination cars-windows works better than the combination windows-cars. We have tried SL with SVM classifiers, but the reconstruction error increased by $2.7\% \pm 0.4\%$ compared to SL with decision tree classifiers.

Implementation. Training SL on 166 LabelMe images takes 18 hours on a 2.66Ghz, 3.4GB RAM PC. On a test image, the Matlab implementation of SL takes on average two minutes to label all objects, and to assign a normal to every pixel. The 3D surface reconstruction is real-time.

6. Conclusion

We have presented an approach to scene interpretation that exploits shape-from-texture to yield a 3D model of the scene, and reduce the noise inherent in low-level object detectors. Our approach does not use supervised learning of 3D scene layouts. It relaxes the assumptions of prior work that supporting surfaces of objects are planar, horizontal, and parallel, and that vertical surfaces are planar with a finite set of discrete orientations. Our results demonstrate that our scene interpretation informed by texture is more in tune with the particular geometry and semantic content

Figure 4. LabelMe dataset: Our scene reconstruction results. From left to right: the objects selected by our sequential labeler SL, the needle plot of the diffused surface normals, the reconstructed surfaces with texture mapping, the surfaces viewed from the top(top row) and viewed from the left (bottom row). Fig. 5 presents the zoomed-in details of the circled regions. Both examples show that we correctly reconstruct building surfaces at a 90 degrees angle. The top row demonstrates our capability to reconstruct details of the facade (red circle), in constrast with previous work that assumes planar surfaces. The bottom row shows that we correctly reconstruct the uphill street going behind the scene (red circle), whereas [7] considers the ground to be a flat plane.

of the scene than alternative interpretations such as "blocks world" or "image pop-up". While our evaluation focuses on street scenes, our appraoch can handle any scenes in which instances of object classes spatially repeat.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *CVPR*, 2009.

[2] S. Y.-Z. Bao, M. Sun, and S. Savarese. Toward coherent object detection and scene layout understanding. In *CVPR*, pages 65–72, June 2010.

[3] N. Cornelis, B. Leibe, K. Cornelis, and L. Gool. 3d urban scene modeling integrating recognition and reconstruction. *IJCV*, 78:121–141, July 2008.

[4] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class object layout. In *ICCV*, 2009.

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1627–1645, 2010.

[6] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, pages 1–8, 2009.

[7] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010.

[8] J. H. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. In *ECCV*, volume 2, pages 522–535, 2006.

[9] V. Hedau, D. Hoiem, and D. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV*, pages VI: 224–237, 2010.

[10] D. Hoiem, A. Efros, and M. Hebert. Closing the loop in scene interpretation. In *CVPR*, 2008.

[11] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, pages 654–661, 2005.

[12] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 75(1):151–172, 2007.

[13] H. D. III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning Journal*, 2009.

[14] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *JAIR*, 4:237–285, 1996.

[15] T. K. Leung and J. Malik. Detecting, localizing and grouping repeated scene elements from an image. In *ECCV*, volume 1, pages 546–555, 1996.

[16] L.-J. Li, R. Socher, and L. Fei-Fei. Towards total scene understanding:classification, annotation and segmentation in an automatic framework. In *CVPR*, 2009.

[17] W.-C. Lin and Y. Liu. A lattice-based MRF model for dynamic near-regular texture tracking. *IEEE TPAMI*, 29(5):777–792, 2007.

[18] A. Lobay and D. A. Forsyth. Shape from texture without boundaries. *IJCV*, 67(1):71–91, 2006.

[19] A. Loh and R. Hartley. Shape from non-homogeneous, non-stationary, anisotropic, perspective texture. In *BMVC*, 2005.

[20] V. Nedovic, A. W. M. Smeulders, A. Redert, and J. M. Geusebroek. Stages as models of scene geometry. *IEEE TPAMI*, 32(9):1673–1687, 2010.

[21] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 77(1-3):157–173, 2008.

[22] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47:7–42, April 2002.

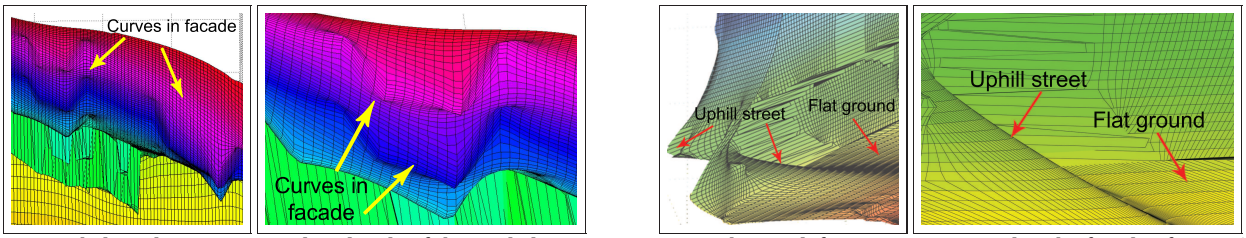


Figure 5. LabelMe dataset: Zoomed-in details of the circled regions in Fig. 4. The two left images correspond to the facade of Fig. 4(top) viewed from the top. The two right images correspond to the ground surface of Fig. 4(bottom) viewed from the left. We correctly reconstruct the curved parts of the facade, as well as the uphill street.

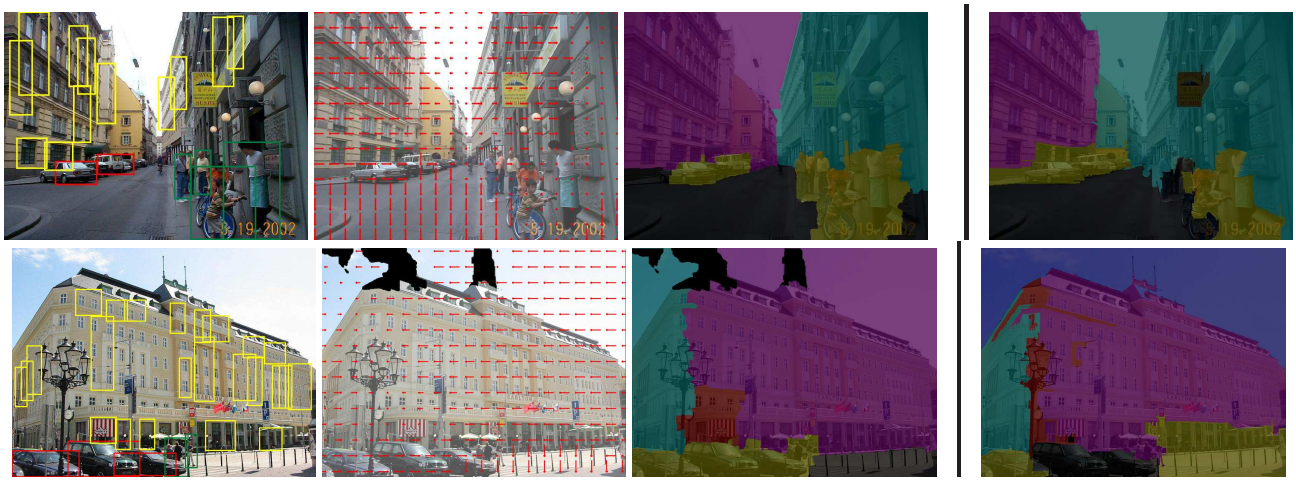


Figure 6. Comparison of our scene reconstruction results to those of [7] on example images from the Geometric Context dataset. From left to right: the detected objects after SL, our computed surface normals, the regions after discretization of our normals into surface layout labels, the regions labeled by [7]. The color-coding of the labels is the same as in [7]. Top row shows that [7] merges the right building with a part of the left building, whereas we succeed in separating buildings with different orientations. Bottom row shows that we are able to correctly label the ground and car regions. We do not detect the sky region, because we do not use a sky detector.