Neural Networks for Cross-Modal Recognition and Vector Object Generation

Mathieu Aubry

Neural Networks for Cross-Modal Recognition and Vector Object Generation

Mathieu AUBRY

HDR THESIS

Defended on August 30th 2019 in front of a jury composed of:

Martial Hebert	CMU	Rapporteur
Iasonas Kokkinos	UCL	Rapporteur
Cordelia Schmid	INRIA	Rapporteuse
Alexei Efros	UC Berkeley	Examinateur
Patrick Perez	Valeo AI	Examinateur
Jean Ponce	INRIA	Examinateur
Andrew Zisserman	Univ. Oxford	Examinateur

Preamble

This document is an overview from my research presented in order to obtain an *Habilitation à Diriger des Recherches* of Université Paris Est, a diploma that is necessary in France to be the official advisor of PhD students. It does not aim at presenting in details technical contributions or careful evaluations, that can be found in the published papers. Instead, it presents how a set of research projects are part of a consistent scientific contribution.

Abstract

Over the last decade, Deep Learning became an omnipresent tool in Computer Vision, and more and more systems are learned end-to-end from raw data without explicit manual design of intermediate representations or features, which were a key element of "classical" vision systems. This end-to-end learning paradigm proved adapted to many applications, and intermediate representations learned by neural network architectures are impressively good general purpose features. However the relations between deep features and human representations of the world are unclear, resulting in an impression of "black-box" as well as in an increased difficulty to control and analyze the behavior of deep networks. In this report, I discuss how to learn neural networks more inline with the following two intuitions: (i) the same scene can appear very differently and be depicted in different modalities and (ii) complex objects can be explained by simple primitives.

In the first part, I discuss several approaches to recognize an object depiction across modalities. This is crucial in many applications where one would like to learn a task in one image domain and perform it in another domain, for example learning to drive in a video game and driving in the real world. While it is natural for a human to recognize that two objects are the same except for their color, to associate objects in a stylized video-game to their real-life counterparts, or to understand that a drawing is the study of a specific oil painting, theses task are not obvious from a computer point of view. Classical features could be designed to be robust to such variations in the depiction style, but there is no guaranty that learned representations will have similar properties and have to be trained specifically for this purpose.

In the second part I focus on the design of neural network architectures that learn to generate images or 3D objects in a way that corresponds to our intuition that they are made of elementary parts. This type of representation would be particularly important to create intuitive design tools. Concretely, I show how to design neural networks which are able to deform a 3D template into a target shape, which can build complex surfaces from parametric surface elements, and which decomposes an image into simple vector layers. Opposite to many approaches which generate pixels or voxels, these architectures generate *vector* objects, described by parametric functions in continuous space at an infinite resolution.

Importantly, the approaches I will present are designed to be trained without hand-labeled data. Indeed, labeling data is often a practical challenge and limits the development of deep learning solutions.

Résumé

Ces dernières années, l'apprentissage profond est devenu un outil omniprésent en vision artificielle. De plus en plus de systèmes sont appris de bout en bout, sans que des représentations intemédiaires des images définies manuellement soient utilisées, alors qu'elles étaient un élément central des systèmes de vision "classiques". Ce paradigme d'apprentissage de bout en bout s'est révélé adapté à de nombreuses applications, et les représentations intermédiaires apprises par les réseaux de neurones profonds être des représentations des images très génériques. Cependant, les relations entre ces représentations profondes des images et les intuitions humaines du monde restent floues, ce qui participe à l'impression que les réseaux de neurones sont une boîte noir et à la difficulté d'analyser et de contrôler le comportement des réseaux de neurones artificiels. Dans ce rapport, je discute comment apprendre des représentations neuronales qui soient plus alignés avec deux intuitions: (i) la même scène peut être apparaître très différente dans différentes modalités, comme par exemple un dessin et une photographie; (ii) les objets complexes sont composés de primitives plus simples.

Dans la première partie du rapport, je discute plusieurs approches pour reconnaitre un objet représenté dans différentes modalités. Cette question est cruciale pour un grand nombre d'applications pour lesquelles on voudrait pouvoir apprendre à effectuer une tâche dans un domaine et l'effectuer dans un autre, par exemple apprendre à conduire en simulation pour conduire en conditions réèles. Alors qu'il est naturel pour un humain de reconnaitre que deux objets sont les mêmes à l'exception de leurs couleurs, d'associer un objet stylisé dans un jeu vidéo à un objet dans le monde réèl, ou encore de comprendre qu'un dessin est une étude d'une peinture, aucune de ces tâches n'est évidente pour un ordinateur. Des représentations d'images peuvent être définies à la main pour être invariantes à de tels changements, mais il n'y a aucune garantie que des représentations apprises présentent de telles invariances si elles ne sont pas apprises spécifiquement avec cet objectif.

Dans la seconde partie du rapport, je me concentre sur la définition de réseaux de neurones pour la génération d'images et de formes 3D qui respectent l'intuition que des formes complexes sont composés à partir de formes élémentaires. Ce type d'approche est particulièrement important pour créer des outils de design intuitifs. Plus concrêtement, je montre comment créer des réseaux de neurones qui puissent déformer un gabarit 3D en une surface cible, construire des formes complexes à partir d'éléments de surface paramétriques et décomposer une image en une séries de couches vectorielles monochromes. Contrairement à beaucoup d'approches qui génèrent des pixels ou des voxels, les méthodes que je présente permettent de générer des objets vecto-

riels, décrits par des fonctions paramétriques dans un espace continu et ayant doncune résolution virtuellement infinie.

Un aspect commun important à toutes les approches que je présente est qu'elles sont prévues pour être entrainées sans données annotées par des humains. En effet, annoter des données est souvent une difficulté pratique importante pour mettre en place des outils d'apprentissage profond et le travail présenté ici vise à l'éviter.

Contents

Chapter 1

Introduction

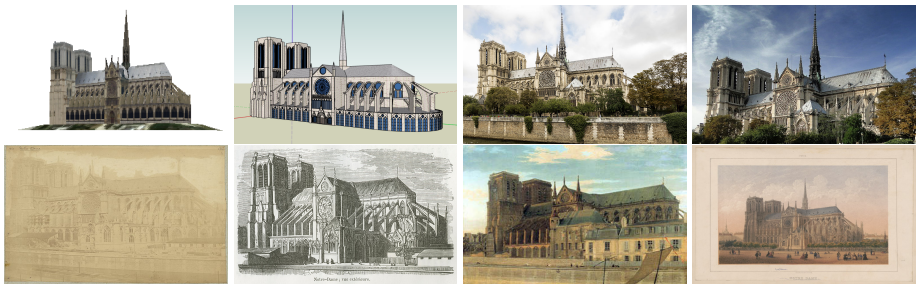
1.1 Objective and Outline

The goal of this work is to design neural networks more inline with human intuitions. We focus on two particular challenges that correspond to the two main parts of this report and are visualized in Figure 1.1.

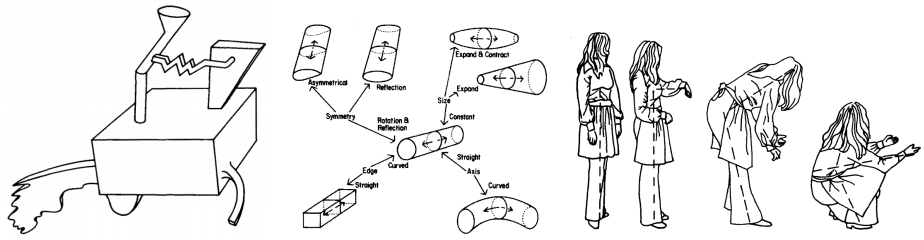
In Chapter 2, we discuss how to learn a neural network robust to changes in the depictions of the same object or scene. A human is capable of identifying the same scene depicted in different styles and thus for example of learning properties of an object on drawings and then to estimate them on photographs. Our aim is to design deep learning systems able to do the same. We will first present a domain transfer approach, based on an analysis of deep features, learning how to associate features from rendered and real images. We will then discuss how randomization of rendering parameters during the training of a CNN on a synthetic dataset allows impressive generalization on real data. Finally, going beyond the case of rendered images, we will show how redundancy in a dataset of images from different modalities can be used to explicitly learn invariant features.

In Chapter 3, we present a new image and shape generation approach, which represents them as the deformation and composition of simple primitives. This is inline with our intuition that objects are the same even when deformed, and that complex objects are composed of simpler parts. Generating objects in such a way would thus allow new simpler methods to design and edit 3D shapes and images. We start by presenting a network that learns to deform a 3D template while respecting our intuition of correspondences, which allows for example to transfer texture across shapes. We then introduce a deep learning approach which allows to reconstruct directly 3D surfaces and meshes from a single image. Finally, we show that the same intuition can be used to decompose images into a set of layers each corresponding to meaningful regions of images, such as objects or object parts.

Before developing these contributions, we first outline in the rest of this chapter the motivations for our work, the challenges we face, the most related research directions and our main contributions.



(a) Human perception of objects is robust to the modality of the depiction.



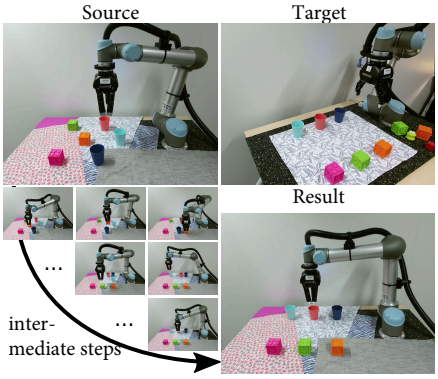
(b) Human perception of objects as being composed of parts. Figures from [13].

Figure 1.1: **Objectives.** Our goal is to design deep features inline with two simple intuitions about human perceptions: (a) it is robust with respect to changes in the modality of the depiction ; (b) it interprets the objects as being composed of parts. These two intuitions have also been long standing motivations for computer vision systems.

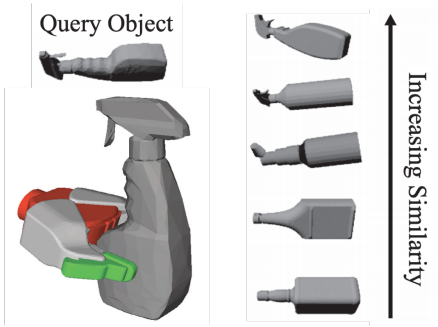
1.2 Motivation

The goal of the research presented in this report is to understand and find better representations of the spaces of images and 3D shapes. However there are also concrete motivations, the contribution we present being key for applications in particular in robotics, design and data exploration. We explored these applications in several works, different examples are visualized in Figure 1.2.

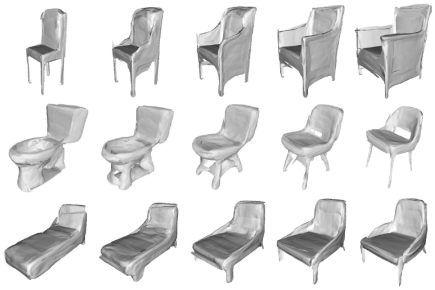
Robotics. A key problem in robotics is estimating the state of the world, i.e. the positions of objects with respect to the robot and their properties. Doing so from one or a few uncalibrated cameras would be an important step toward open-world robotics. The work we present makes several contributions in this direction. The methods discussed in Chapter 2 allow to bridge the gap between simulations, rendered from 3D models and real images. In Section 2.1, we use this ability to recognize and coarsely align 3D models of objects in images. In Section 2.2, we demonstrate a system that learns from synthetic images to accurately estimate the position of an object with respect to a robot from real uncalibrated images. The 3D shape deformation and estimation networks we introduce in Chapter 3 could be used in different ways for robotics applications. The shape deformation network introduced in Section 3.1 could transfer properties, such as grasps, between annotated 3D models.



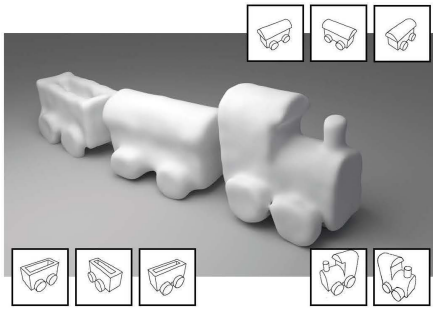
(a) Visual rearrangement planning [87].



(b) Grasp transfer [63].



(c) Shape interpolation [39].



(d) 3D shape design from drawings [26].

Figure 1.2: **Applications.** Four examples of possible applications of the approaches presented in this report for robotics (top) and design (bottom).

Data exploration and design of shapes and images. Because the approaches we present are more inline with human intuition, they can lead to new tools to explore data and design new images or 3D scenes. 3D shape design is especially challenging and the existing systems are often time consuming and require expert skills. The approach which we present in Section 2.1 could be used to search 3D models starting from a query image. An alternative would be to directly reconstruct a mesh of the model, as we do in Section 3.2. As we have demonstrated in [26], the input image could also be a non realistic depiction. Note that the representation of the models we present also allows to interpolate meaningfully between 3D models. Advanced image search and manipulation are also difficult problems. The work we present in Section 2.3 explicitly focuses on identifying repeated elements in artistic depictions. Finally, Section 3.3 targets image editing applications, with the ambition to enable designers to easily interact with the images generated by a neural network. We demonstrate that the approach developed opens up new interesting image search possibilities.

1.3 Challenges

In this report, we face two main challenges. First, because we do not want to rely on human supervision, we will only consider techniques that do not require manually labeled data. Second, we need to select data representations for our inputs and outputs that are well suited for deep learning approaches, which is especially challenging for 3D data.

Learning without manual annotations. State-of-the-art techniques in Deep Learning typically rely on very large annotated datasets. While large datasets are publicly available for some tasks, acquiring data for a new or specialized task is often costly, difficult and time consuming. This is one of the main practical limitations to many applications of Computer Vision. To avoid this issue, we develop methods that can be trained without manual annotations, either directly from unlabelled data, or from synthetic examples.

Representing 3D data. There is no natural and standard representation of 3D data, especially no representation that neural networks can leverage in a meaningful way. Voxels, rendered images, point clouds, parametric models, meshes, structured CAD models and depth maps are also popular options. Each of these representation comes with its own advantages and drawbacks. We will develop and demonstrate the advantages of two main approaches, rendered images with randomized parameters and a new learnt parametric surface representation we introduce. Note that while almost all Deep Learning approaches for images represent then as pixel arrays, we also introduce a new parametric representation of images and show its advantages.

1.4 Relation to previous work

Because they correspond to our intuition of what an artificial visual system should do, connexionnist approaches, scene decomposition into primitive elements and invariant representations design have a very long history in Computer Vision. This prohibits an exhaustive review of related approaches. We will thus only give a very brief historical introduction to these key topics and refer to the published papers for a more detailed review of approaches related to each of our contribution.

Neural Networks. A formal neuron or perceptron [68, 80] computes a non-linear function (also called activation) of a weighted average of its inputs, the weights being the parameters of the function the neuron implements. The limitations of the family of functions that a single formal neuron can represent [69] can be overcome by using a succession of layers of parallel neurons [24]. Each of these layers can be seen as a feature describing the inputs of the network, and the succession of many such layers is referred to as a deep network. If the neurons in each layer depend on all the neurons in the previous layer, the resulting architecture is called a multi-layer perceptron (MLP).

Structured connections [36] as well as shared weights [82] between neurons can be used, leading to convolutionnal neural networks (CNNs). More complex operations can also be considered, making deep learning a quite generic modular framework.

One of the keys to obtaining a neural network that can perform a given task is of course the choice of the weights of each neuron. In a standard machine learning approach, they can be learned to optimize an objective function. This objective is a highly non convex function of the weights, but using efficient gradient descent, backpropagation [81], or one of its variations, often produces surprisingly good results.

While CNNs were already demonstrated to perform well on practical tasks such as digit recognition in the late eighties [54], their importance in the Computer Vision community really started with the demonstration of their performances for large scale image recognition in 2012 [52]. All the work presented in this report is based on the deep learning framework and builds on the recent advances in this field.

Scene decompositions. There has been a long standing Computer Vision trend to look for a simplified and compact representation of the visual world, which can be dated back to Roberts' block world assumption [79]. Different types and variabilities of primitives used to represent the objects have been proposed. For example, in 1971 Binford [14] proposed to represent the 3D world using generalized cylinders and in 1987 the seminal work of Biederman [13] aimed at explaining and interpreting the 3D world and images using geons, a family of simple parametric shapes. These ideas have recently been revisited using neural networks to represent a 3D shape using a set of blocks [95]. For images, the idea of identifying elementary shapes and organizing them in layers has been successfully applied to model images [4, 48] and videos [97]. All the approaches we propose in Chapter 3 are in line with these works.

The decomposition of an image into a pictorial structure of simple elements has also been developed and successfully used for object recognition in images [35, 31, 33]. The matching score we propose in Section 2.3, based on the aggregation of mid-level feature matches, as well as the part-based matching we propose in Section 2.1, are related to these approaches.

Invariant representations. Roberts' work [79], as most approaches until the nineties [70], relies on object contours to match images and recognize objects. Indeed, we have a strong intuition that edges will not depend on the modality of a scene representation, and will be robust to effects such as illumination changes which are not relevant for recognition. However, edges are in practice difficult to extract reliably and to match. Many descriptors have been designed to represent 3D models and images, with focus on different types of invariances. For 3D models, the focus has often been on invariance with respect to 3D deformation [20, 49, 11, 83, 91, 7]. For images, feature descriptors based on soft representations of edges as gradient histograms, such as SIFTs [61] for local features and HOGs [25] for dense features have been particularly successful for image matching and recognition.

More recently, features learned with neural networks proved more powerful than manually designed features for many tasks. In particular, they generalize impressively well if they have been trained with rich-enough training data. This motivates practical approaches to learn from randomized synthetic datasets such as the one we discuss in Section 2.2 and 3.1. However, the properties and invariance of intermediate features and how they can be used for other tasks is not well understood [5, 102, 57]. Theoretical analyses [16, 64] have shown that neural network architectures are able to learn invariances to some transformations, and even that the weights of the first layers of neural networks could be manually designed with little loss of performance, but this provides little intuition on the features of learned networks. In Section 2.1, we attempt to develop this understanding of learned fetures using images rendered from 3D models, and use it to propose a deep domain transfer approach.

Some works specifically try to use neural networks to learn invariant representations, for example applying metric learning to identify faces [18, 86] with nearest neighbour classification [22], or to match local patches [42]. These approaches have been applied to supervised domain invariant feature learning [84, 23]. In Section 2.3, we present an unsupervised metric learning approach to domain invariant feature learning.

1.5 Contributions

The work presented in Chapter 2 mainly builds on existing techniques, such as supervised domain adaptation, domain randomization and metric learning. Their main novelty is in the problems they define and tackle and for which they adapt these techniques. In the works presented in Section 2.1, we were the first to propose to analyze deep features using synthetic images and to use them for 2D-3D alignment. In Section 2.2, we introduce the first approach to learn single-view relative pose estimation between a robot and an object. Finally in Section 2.3, motivated by the need of art historians, we introduce the problem of style invariant discovery of near duplicate pattern in large collections of artworks.

On the contrary, the main contributions of the work presented in Chapter 3 are the new approaches they propose to classical problems, namely shape matching in Section 3.1, single-view shape reconstruction in Section 3.2 and deep image generation in Section 3.3. The main idea behind these approaches is to use multi-layer perceptrons (MLPs) to learn to parametrize a set of meaningful continuous functions of the 2D or 3D unit square. These functions are 3D deformations of a shape in Section 3.1, local parametrizations of a surface in Section 3.2 and elementary masks of image parts in Section 3.3.

The work presented here is clearly separated from the work I did during my PhD, which didn't include any deep learning. It has been produced by students I co-supervised - Francisco Massa (graduated in 2016), Vianney Loing (graduated in 2017), Xi Shen, Thibault Groueix and Othman Sbaï - in collaborations with different researchers -

Bryan Russell, Renaud Marlet, Alexei Efros, Matthew Fisher, Vladimir Kim and Camille Couprie. It has been published in top Computer Vision venues (ICCV 2015, CVPR 2016, CVPR 2018, ECCV 2018, IJCV 2018 and CVPR 2019).

Chapter 2

Cross-modal recognition

In this Chapter, we will present three different approaches to compare images from different modalities with deep features. In Section 2.1, we use pre-trained CNN features, analyze how they respond to image variations and present a domain adaptation approach to bridge the domain gap between features from rendered 3D models and real images. In Section 2.2 we discuss domain randomization, a simple way to directly learn a complex task on synthetically rendered images in such a way that the learned network applies to real images. Finally in section 2.3, we will go beyond the case of rendered scenes and we will show how consistency can be exploited in a metric learning approach, fine-tuning features to make them invariant to the strong modality changes present in an artistic dataset.

2.1 CNN features from rendered 3D models.

In this Section, we first discuss how rendered images can be leveraged to analyze and better understand deep features. Based on the intuition gained with this analysis, we will then present a domain adaptation method for pre-trained deep features. Finally, we show how to use this domain adaptation to detect and align object instances described by 3D models in real images.

The analysis described in the first paragraph is a joint work with Bryan Russell and was presented at ICCV 2015 [10]. The domain adaptation approach and its application to 2D-3D alignment were developed as part of Francisco Massa's PhD [67] and presented at CVPR 2016 [66].

Feature analysis from rendered views. In [10], we were the first to propose to use rendering to help understanding deep features. By developing a simple additive model, we were able to quantify the intuition that CNNs first extract and separate the information associated to different factors of variation in a given image, before learning invariance and specializing for a task. A similar analysis would of course be possible using a database of natural images captured in controlled conditions and spanning different factors of variations, e.g. the NORB [55], ETH-80 [56] and RGB-D object [53]

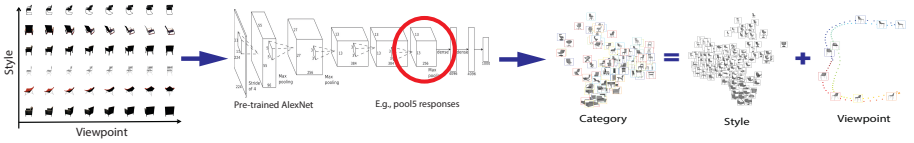


Figure 2.1: Overview of our approach to analyze deep features. We start by rendering images in controlled conditions (left), then extract their features (middle) and finally decompose these features linearly according to the different factors of variation (right). The images on the right side visualize the first two components of the Principal Component Analysis (PCA) of the feature, style factor and orientation factor.

datasets, where different objects are rotated on a turntable and lighting is varied during image capture. However, not only are these datasets difficult and costly to collect but they neither offer the variety of objects available in 3D model collections, nor the flexibility given by rendering.

Our approach, presented in Figure 2.1, was to generate images with controlled variations, seen as the realization of random variables, and then compute the features produced by off the shelve CNNs for these images. While this can be done without any 3D involved, for example generating uniform images of different colors, including rendered views of 3D objects lead to more realistic generated images, and consequently an analysis that is more relevant for CNN trained on real image. We thus first collect a large database of 3D models of objects from 5 different common object categories (car, chairs, bed, toilets and sofas). Each object category can then be rendered using different rendering factors, such as rotation, scale, translation, foreground and background color.

To introduce our model further, we need to formalize the problem. Let $\Theta_1, \ldots, \Theta_N$ be sets of parameters for N factors of variation we want to study. To simplify notations, we will assume that each of these sets Θ_k is finite. We consider an image of the scene with parameters $\theta = (\theta_1, \ldots, \theta_N)$, where $\theta \in \Theta = \Theta_1 \times \cdots \times \Theta_N$. We consider that θ is the realization of a random variable following a uniform probability distribution. We write the centered CNN features $F^L(\theta)$ of the layer L of a given network as a linear combination of terms each depending on single factors k and a residual:

$$F^L(\theta) = \sum_{k=1}^{N} F_k^L(\theta_k) + \Delta^L(\theta) \tag{2.1}$$

where $\Delta^L(\theta)$ is the residual feature and $F_k^L(\theta_k)$ marginalizes over the parameters for all factors except k:

$$F_k^L(t) = \mathbb{E}(F^L(\theta)|\theta_k = t) \tag{2.2}$$

$$= \frac{|\Theta_k|}{|\Theta|} \sum_{\theta \in \Theta | \theta_k = t} F^L(\theta) \tag{2.3}$$

Using computer-generated images, we can easily compute this decomposition by rendering the images with all the rendering parameters corresponding to the sum in equa-

(a) Relative importance of the style factor

(b) Relative importance of the viewpoint factor

(c) Relative importance of the residual

Figure 2.2: Visualization of the relative importance of the residual, factors related to viewpoint and style (i.e. choice of 3D model) for an AlexNet architecture trained on ImageNet. Warmer colors represent a higher importance. One can observe that: (i) the importance of the residual continuously decreases, showing that our linear decomposition model is more meaningful in the later layers of the network; (ii) the importance of the style increases as one go deeper in the network; (iii) the importance of viewpoint increase in the first part of the network before decreasing in a second part.

tion (2.3). Direct computation shows that all the terms in this decomposition have zero mean and are un-correlated. That implies in particular that:

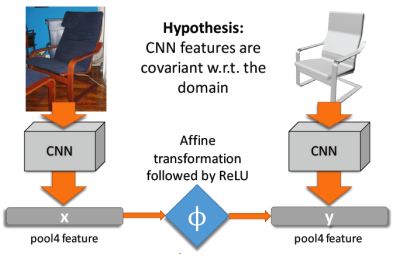
$$\text{var}(F^L) = \sum_{k=1}^{N} \text{var}(F_k^L) + \text{var}(\Delta^L) \tag{2.4}$$

We can thus decompose the variance of the features F^L as the sum of the variances associated to the different factors and a residual. The part of the variance explained by F_k^L indicates the importance of the factor k for the representation, while the part of the variance explained by the residual indicates how well the feature separates the influence of the different factors. When analyzing the decomposed features we thus report the relative variance $R_k^L = \text{var}(F_k^L)/\text{var}(F^L)$ as well as the relative variance of the residual $R_\Delta^L = \text{var}(\Delta^L)/\text{var}(F^L)$. Note that $R_\Delta^L + \sum_{k=1}^{N} R_k^L = 1$ and that the value of the R_k^L and R_Δ^L does not depend on the relative sampling for the different factors.

Detailed analysis with several factors of variations are available in the paper and its supplementary material, including experiments with real images, study of the intrinsic dimensionality of the different factors and qualitative visualizations. Here, we simply summarize the key observations.

The most surprising fact is that the simple linear decomposition presented above and visualized in figure 2.1 leads to meaningful results, and makes more and more sense when one progress in the layers of a CNN, the importance of the residual decreasing progressively while the importance of the other terms increase. In particular, intermediate features computed from image collections that vary along two different factors, such as style and viewpoint, can be relatively well approximated by a linear combination of features corresponding to the factors in the higher levels of a CNN.

We also found a consistent trend in a networks trained for object category recognition when varying object pose and type: in a first part of the CNN, the importance of the residual decreases when the importance of all factors related to viewpoint increase. Then, in a second part, both the importance of the residual and the importance of fac-



(a) Idea of our domain adaptation.

(b) Training data for our adaptation.

Figure 2.3: Our supervised domain adaptation in feature space approach.

tors related to viewpoint decrease, while the importance of the style of the object, i.e. the intra-class variations in our experiments, increased. This is visualized for AlexNet in figure 2.2. This explains why the features extracted from the middle of a network are more generic features and can more easily be transferred across tasks and datasets.

Finally, we outlined differences between AlexNet and VGG networks, as well as between networks trained for place recognition and networks trained for object recognition. In particular, we were able to quantify: (i) the fact that VGG networks better separated factors of variations and were, at the latest layers, more invariant to viewpoint; (ii) the fact that colors and especially background colors was a much more important cue for networks trained for place recognition than networks trained for object recognition. While this type of results can seem intuitive, we provided a simple way to quantify them by leveraging synthetic image.

Domain Adaptation from real to rendered images features. In this section, we leverage the intuition developed above - the CNNs features from the middle of a network trained for image recognition able to extract and separate style and content factors of variations in an image - to propose a simple domain adaptation approach on intermediate features, presented in Figure 2.3.

More formally, we seek to learn a transformation ϕ over the features of real images. Intuitively ϕ is a projection of the real image feature space to the space of features from CAD rendered views. Ideally, ϕ has the property of mapping a given real image feature depicting an object of interest to features of rendered views of CAD object models with the same geometry, style, and pose. Suppose we have as input a set of N pairs of features $\{(x_i, y_i)\}_{i=1}^{N}$ corresponding to examples of real images and rendered views of well-aligned CAD models, respectively. We seek to minimize the following cost over ϕ:

$$L(\phi) = -\sum_{i=1}^{N} S\left(\phi\left(x_i\right), y_i\right) + R(\phi), \tag{2.5}$$

where S denotes a similarity between the two features $\phi(x_i)$ and y_i, and R is a regularization function over ϕ. Note that in the case where ϕ is an affine transformation,

our formulation is similar to the one of Lenc and Vedaldi [57] where a mapping was learned given image pairs to analyze the equivariance of CNN features under geometric transformations.

While the simplest choice for ϕ is indeed an affine transformation, which we used as a reference in our experiments, we also tested more constrained and complex transformations. We focused on transformations that could be formulated as CNN layers, and in particular successions of convolutional and ReLU layers. Note that considering more complex transformations also increases the risk of overfitting. Similar to Lenc and Vedaldi [57] we attempted to constrain the structure of the transformation and its sparsity. This is easily done in a CNN by replacing a fully-connected layer by a convolutional layer with limited support, which implies translation invariance in the adaptation. We found that the best-performing transformation was only a slight modification of the affine transformation:

$$\phi(x) = ReLU(Ax + b), \tag{2.6}$$

where $ReLU(x) = \max(0, x)$ is the element-wise maximum over zero. We observed that applying the $ReLU$ function consistently improved results, and is in agreement with state-of-the-art CNN architecture design choices for object recognition.

We tried both L_2 and squared-cosine similarity to measure the similarity in Equation (2.5). We found that the squared-cosine similarity $S(a, b) = -\left(1 - \frac{a^T b}{\|a\|\|b\|}\right)^2$ leads to better results. This is expected, since cosine similarity is known to work better when comparing CNN features, but also because we later used the cosine distance to compare real and synthetic features. This result is also consistent with the observation of the importance of task-specific similarities in Lenc and Vedaldi [57].

Our adaptation formulation requires a large training set of well-aligned pairs of images and rendered views of CAD models matching the style and pose of depicted objects. Such a dataset is difficult to acquire. Instead, we built on recent approaches for effective training from rendered views [72, 89] to render views of CAD models and composite on natural image backgrounds (see examples in Figure 2.3b). This gives us access to virtually unlimited training data. The backgrounds provide "natural-looking" surrounding context and encourages the transformation ϕ to learn to subtract away the background context. To avoid color artifacts in the composite images, we used gray-scale image pairs and also used gray-scale images at test time. Note that contrary to prior approaches using manually-annotated scenes to increase the realism of the composite [72, 73], we do not directly use any object annotation in our background selection process. Because the complexity of the transformation ϕ is limited, and because the network extracting the features x is not learnt, we can expect to not overfit to synthetic composite images and generalize to real images.

Application to 2D-3D alignment. In the previous paragraph, we have assumed that the real image was centered around the object of interest. In [66] we wanted to explore whether is was possible to go further and to directly learn to detect object in images from 3D models, and even align a specific instance of a large collection of 3D model to the test images. Figure 2.4 shows the 2D-3D exemplar detection pipeline we proposed.

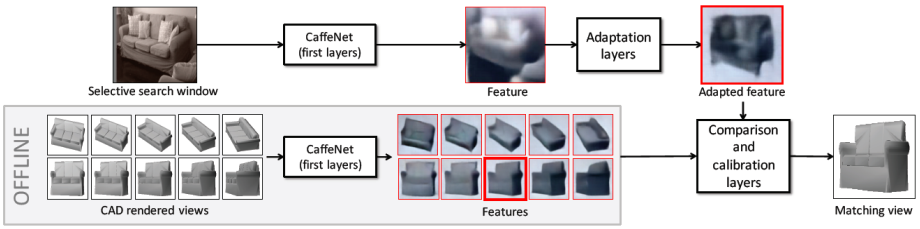


Figure 2.4: Our approach to detect object instances in real images using 3D models. Note that the intermediate features are visualized as images using the feature inversion approach of [28].

Once the adaptation has been learnt, we follow the initial part of the R-CNN object detection pipeline [38] to perform the comparison with relevant parts of the image: we first extract a set of selective search windows [96] and compute CNN responses x at an intermediate layer (e.g., CaffeNet *pool5* layer) for each window. We then apply our adaptation ϕ to these features and compare the results $\phi(x)$ to the features of different CAD model rendered views. Let $s_i(x) = S(\phi(x), y_i)$ be the similarity between $\phi(x)$ and the features y_i of the ith rendered view.

As described in [9], calibrating the different similarity measures is an important step for comparing similarity across different views and CAD models. Starting from the initial similarity score $s_i(x)$, we apply the affine calibration routine of [9] to compute a new calibrated similarity $s'_i(x) = c_i s_i(x) + d_i$. The scalar parameters c_i and d_i are selected using a large set of random patches such that $s'_i(x_0) = -1$ and $s'_i(x_1) = 0$, where x_0 and x_1 correspond to random patch features with mean and 99.99-percentile similarity scores, respectively.

We take advantage of the fact that in an exemplar-based detection setup the expected aspect ratio of the alignments are known. We remove candidate rendered-view alignments when the aspect ratio has a difference of more than 25% between the selective search window and rendered view. Finally, we rank the remaining alignments by their score $s'_i(x)$ and perform non-maximum suppression to obtain the final detections.

Note that the computation of the calibrated similarity can be implemented efficiently as a CNN layer. The cosine similarity can be implemented by a feature-normalization layer followed by a fully-connected layer. The weights of the fully-connected layer correspond to a matrix Y of stacked unit-normalized features for the exemplar rendered views, computed in an offline stage. While the affine calibration could be implemented in a deep learning framework as an additional layer, we incorporated it directly into the fully-connected layer by replacing the matrix rows by $Y_i \leftarrow c_i Y_i$ and adding a bias d_i corresponding to each row i. The final exemplar rendered-view scores is $Y\phi(x) + d$ given image features x, and can be computed by a single forward pass in a CNN.

Our results clearly demonstrate the efficiency of our domain adaptation approach. We evaluated our performance both on the Ikea [58] dataset (c.f. Table 2.5a) for instance detection and the chair subset of Pascal VOC detection dataset [29] from [8]

class	chair *poang*	bookcase *billy1*	sofa *ektorp*	table *lack*	bookcase *billy2*	desk *expedit*	bookcase *billy4*	bed *malm2*	stool *poang*	mAP
Lim *et al.* [58]	19.9	**14.8**	**6.4**	9.4	**15.7**	**15.4**	**13.4**	**7.6**	6.4	12.11
Ours	**33.8**	5.5	**7.8**	**20.0**	0.1	**19.9**	0.4	**8.8**	**25.0**	**13.48**

(a) Performance on the Ikea dataset [58] (with exhaustive annotations)

	Adaptation	No Adaptation	
		Comp.	White
Aubry *et al.* [9]	33.9		
DPM [32]	41.0		
R-CNN [38]	44.8		
R-CNN + SVM [38]	54.5		
Peng *et al.* [72]	29.6		
Logistic *pool4*	12.9	3.7	1.4
Logistic *fc7*	26.6	9.2	14.0
Ours, no calibration	4.6	–	2.8
Ours with calibration	49.7	33.5	16.3

(b) Average precision for chair detection on Pascal VOC subset [9]. "White" column corresponds to synthetic images on white background, whereas "Comp" column corresponds to synthetic images composited on real-image backgrounds.

(c) Top detections without adaptation. (d) Top detections with adaptation.

Figure 2.5: Main results of our domain adaptation approach to detection.

for category detection (c.f. Table 2.5b). On both datasets, we demonstrated a clear improvement over baselines. Interestingly our approach failed on all bookcases from [58], which is understandable because in the case of bookcase, the difference between rendered 3D models and real bookcase images is not only in the background, but also in the fact that bookcases are typically full of books, leading to a larger appearance shift that our model clearly does not handle. Qualitatively, as visualized Figure 2.5c and 2.5d the top detections without the domain adaptation are very simple images, where a chair is visible on a uniform background, while with domain adaptation, they all have a natural background. This demonstrates the benefits of our deep feature adaptation approach to compare rendered views of 3D models and real images.

2.2 Learning with domain randomization.

In the previous section, we showed that it was possible to learn to compare pre-trained features on rendered views and on real images. However, one-by-dommaone comparison of features is not always feasible or meaningful. For example, representing an articulated object such as a robotic arm using rendered views in all its positions would require a number of views exponential in the number of joints. In this paragraph, we discuss an approach to learn to perform a task only from synthetic images without explicitly using domain adaptation: domain randomization. The idea is simply to vary the appearance of the synthetic training images to make the network invariant to a wide range of transformations, hoping it includes transforming synthetic images to real images. It can thus be seen as a form of data augmentation. While it comes with little theoretical understanding, this approach proves very effective in practice.

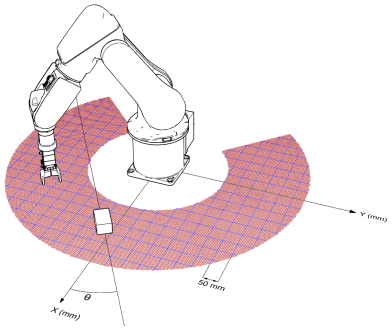
Table 2.1: Summary of results for pose estimation and comparison with baselines using AVP24

Method	aero	bike	boat	bus	car	chair	table	mbike	sofa	train	tv	mAVP24
DPM-VOC+VP [74]	9.7	16.7	2.2	42.1	24.6	4.2	2.1	10.5	4.1	20.7	12.9	13.6
Render For CNN [90]	21.5	22.0	4.1	38.6	25.5	7.4	11.0	24.4	15.0	28.0	19.8	19.8
Viewpoints & Keypoints [94]	37.0	33.4	10.0	54.1	40.0	**17.5**	19.9	34.3	28.9	43.9	22.7	31.1
Classif. approach & AlexNet	21.6	15.4	5.6	41.2	26.4	7.3	9.3	15.3	13.5	32.9	24.3	19.3
+ our joint training	24.4	16.2	4.7	49.2	25.1	7.7	10.3	17.7	14.8	36.6	25.6	21.1
+ VGG16 instead of AlexNet	26.3	29.0	8.2	56.4	36.3	13.9	14.9	27.7	20.2	41.5	26.2	27.3
+ ImageNet data	42.4	37.0	**18.0**	59.6	43.3	7.6	25.1	**39.3**	29.4	**48.1**	28.4	34.4
+ synthetic data	**43.2**	**39.4**	16.8	**61.0**	**44.2**	13.5	**29.4**	37.5	**33.5**	46.6	**32.5**	**36.1**

This type of approach is used in two works presented below. In his BMVC 2016 paper [65], Francisco Massa studied how augmenting data with synthetic images helped a standard task, viewpoint estimation. In his IJCV 2018 paper [60], Vianney Loing presented how domain randomization could be applied to perform relative viewpoint estimation between a robot and a block from an uncalibrated camera, a task which we think is key to make robotics tasks simpler and more robust.

Viewpoint estimation. In [65], we studied in detail the different components that are important for the problem of viewpoint estimation, and showed how they can be combined to improve state of the art performance on the Pascal 3D+ dataset. In particular, we studied the importance of using synthetic data, an idea introduced in [90] but with performance well below state of the art [94]. We also showed that joint training of detection and viewpoint estimation improves the results and found that for our 1D pose estimation problem, a classification approach was performing better than a simple regression, likely because it allows to better represent ambiguities. The summary of our results is presented in Table 2.1. The reported AVP24 performance is the standard measure on Pascal 3D+ and corresponds to average precision for the joint task of detection, using a threshold of 0.5 over the intersection over union measure to determine positives, and 24 viewpoints bins to determine correct viewpoints. It can be seen that while using synthetic data improves average results, it can harm performance for some classes. They are typically classes for which the 3D models are qualitatively very different from the objects seen in real images.

Relative viewpoint estimation for robotics. In [60], we developed a similar approach for a more complex problem: we wanted to be able to precisely localize a building block with respect to a robot to be able to grasp it using cameras looking at the scene but without any calibration information. Acquiring real training data at sufficient scale for such a problem is unrealistic. We thus created a synthetic training dataset for the task, randomizing all view parameters, including robot pose and texture, block size, position, orientation and texture, camera position, orientation and focal length and background room layout and texture. A qualitative example of the generated images is given Figure 2.6b. While these images are very different from the typical test image, such as the one visualized in Figure 2.6c, their diversity allowed the networks to



(a) Relative localization as classification

(b) Training image

(c) Test image

Figure 2.6: We formulate relative block-robot positioning using an uncalibrated camera as a classification problem where we predict the position (x, y) of the center of a block with respect to the robot base, and the angle θ between the block and the robot main axes **a**. We show we can train a CNN to perform this task on synthetic images, using random poses, appearances and camera parameters **b**, and then use it to perform very accurate relative positioning on real images **c**

.

transfer without any fine-tuning from training to test images.

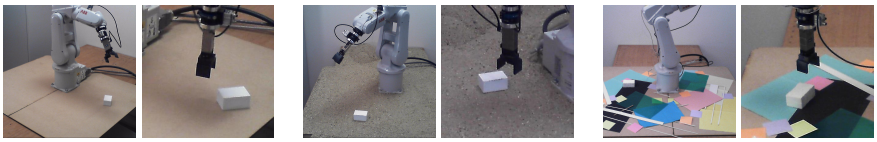
Since the accuracy necessary to actually grasp a block with our robotic arm was very high, we divided the estimation in three subtasks:

1. *Coarse relative localization subtask.* We first consider the very general case where the robot and the block are at random positions and orientations, and the robot joints are also randomly set. The cameras, which are random too, only provide overviews of the scene. The subtask here is to (coarsely) estimate the pose of the block with respect to the robot.

2. *Tool localization subtask.* After the block position is estimated, although possibly with moderate accuracy and confidence, we assume the robot clamp is moved on top of that coarse predicted location. In this setting, the camera and the block remain at unknown positions and orientations, but the clamp is located at a position close to the block, oriented towards the ground and ready to grasp. Now the second subtask is to detect the clamp in the picture, allowing camera close-ups to later perform a finer pose estimation.

3. *Fine relative localization subtask.* Last, using camera close-ups (actually crops of overviews centered on the zone of interest), the third subtask is to finely estimate the block location and orientation with respect to the clamp, hence with respect to the robot, thus enabling the actual grasp.

Each of the three subtasks is solved by training a classification CNN for estimating each variable on synthetic data only. Examples of full images and close-ups from our test data can be seen in Figure 2.7.



(a) 'lab' dataset. (b) 'field' dataset. (c) 'adv' dataset.

Figure 2.7: Example images from the three variants of our evaluation dataset, for the coarse localization (left) and fine localization tasks (right): clean laboratory conditions **a**, more field-like conditions with gravels **b**, and adverse conditions with distractors **c**

Dataset	'lab'	'field'	'adv'
$e_x \leq 5\,\text{mm}$	91.8	89.5	73.4
$e_y \leq 5\,\text{mm}$	88.5	86.8	71.0
$e_\theta \leq 2°$	97.9	93.0	86.5
all together	79.9	70.2	45.1

(a) Success rate (in %)

Dataset	'lab'	'field'	'adv'
e_x (mm)	2.3 ± 1.8	2.3 ± 1.8	4.0 ± 4.4
e_y (mm)	2.6 ± 4.4	2.3 ± 2.0	4.0 ± 5.2
e_θ (°)	0.7 ± 0.6	0.9 ± 0.7	1.1 ± 0.9

(b) Mean and standard deviation of error

Figure 2.8: Results of the robot-block relative localization on real images with 3 cameras and 2 clamp orientations on different (real) evaluation datasets.

Our experiments demonstrate that such a three-step procedure makes sense for robot control as indeed more accurate position and orientation information can be obtained thanks to the refinement subtask. To analyze our proposed pipeline, we created three evaluation datasets, representing more than 1300 different robot and block position with different visual characeristics:

(a) a dataset in laboratory condition ('lab'), where the robot and the block are on a flat table with no particular distractor or texture,

(b) a dataset in more realistic condition ('field'), where the table is covered with dirt, sand and gravels, making the flat surface uneven, with blocks thus lying not perfectly flat, and making the appearance closer to what could be expected on a construction site,

(c) a dataset in adverse condition ('adv'), where the table is covered with pieces of paper that act as distractors because they can easily be confused with cuboid blocks.

A summary of our results is presented in Figure 2.8. They demonstrate our ability to locate very precisely the block with respect to the robot, the errors being of the order of a few millimeters and a degree (Table 2.8b). The thresholds in Table 2.8a correspond to the accuracy necessary to be able to actually grab the block. In practice, using the video feed from cameras placed randomly around the robot, we were able to easily grab a block in a real scenario while training only on non-realistic synthetic images. Videos of results are available on the project website `http://imagine.enpc.fr/~loingvi/unloc`.

2.3 Self-supervised style invariant feature learning.

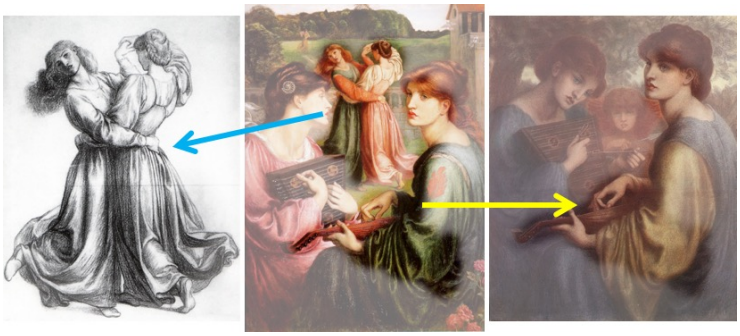


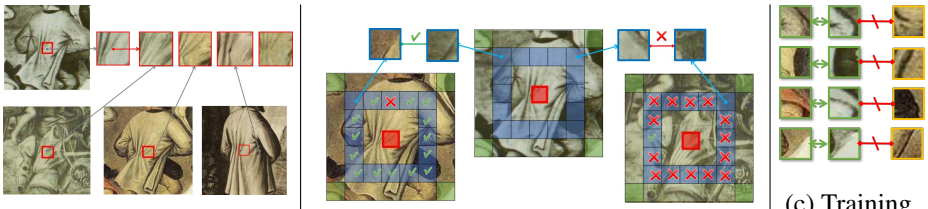
Figure 2.9: Example of relationship between painting and two studies discovered from collection of 195 artworks by Dante Gabriel Rossetti. Sources: *The Bower Meadow* (left: chalk, center: oil, right: pastel)

Domain randomization provides a practical solution for learning from synthetic data in many scenarios, but no general answer to the problem of obtaining features invariant to the style of a depiction. In this section, we focus on the problem of discovering near duplicate patterns in large collections of images including different depiction modalities, in particular artworks collections as visualized in Figure 2.9. This is harder than standard instance mining due to differences in the artistic media (oil, pastel, drawing, etc), and imperfections inherent in the copying process. Our key technical insight is to adapt a standard deep feature to this task by fine-tuning it on the specific art collection using self-supervised learning. More specifically, spatial consistency between neighbouring feature matches is used as supervisory fine-tuning signal. Beyond artworks, we also demonstrate our fine-tuning leads to improve localization on the Oxford5K photo dataset as well as on historical photograph localization on the Large Time Lags Location (LTLL) dataset.

In the following, we first present how we fine-tune our features, then explain how we use them to discover and score matches, and finally discuss some results.

This work was mainly done by my PhD student Xi Shen and will be presented at CVPR 2019 [88].

Dataset-specific Feature Learning. We first describe our strategy for adapting deep features to the task of matching artworks across styles in a specific dataset. Starting with standard ImageNet pre-trained deep features, our idea is to extract hard-positive matching regions form the dataset and use them in a metric learning approach to improve the features. Our two key hypothesis are that: (i) our dataset includes large parts of images that are copied from each other but are depicted with different styles, and (ii) the initial feature descriptor is good enough to extract some positive matches. Our training alternates between two steps that we describe bellow: (1) mining for hard-positive training samples in the dataset based on the current features using spatial consistency,



(a) Candidates from proposal region

(b) Selection with verification regions

(c) Training with positive regions

Figure 2.10: Feature Learning Strategy. (a) Our approach relies on candidate correspondences obtained by matching the features of a proposal region (in red) to the full database . (b) The candidate correspondences are then verified by matching the features of the verification region (in blue) of the query in the candidate images and checking for consistency . (c) Finally, we extract features from the positive regions (in green) from the verified candidates and use them to improve the features using a metric learning loss.

and (2) updating the features by performing a single gradient step on the selected samples.

(1) Mining for Positive Feature Pairs. For our approach to work, it is crucial to select positive matching examples that are both accurate and difficult. Indeed, if the features are trained with false matches, training will quickly diverge, and if the matches are too easy, no progress will be made. To find these hard-positive matching features, we rely on the procedure visualized in Figure 2.10.

First, *Proposal regions* are randomly sampled from each image in the dataset to be used as query features. These are matched densely at every scale to all the images in the dataset using cosine similarity in feature space. This can be done efficiently and in parallel for many queries using a normalization and a convolution layer, with the weights of the convolution defined by the query features. For each query we select one of its top K matches as candidate correspondences (Figure 2.10a). These candidates contain a high proportion of bad matches, since most of the queries are likely not repeated K times in the dataset and since our feature is imperfect.

To verify the quality of candidate matches given by the previous step, we rely on spatial consistency: a match will be considered valid if its neighbours agree with it. More precisely, let's assume we have a candidate match between features from the proposal region p_A in image A and a corresponding region p_B in image B, visualized in red in Figures 2.10b and 2.11. We define a *verification region* around p_A, visualized in blue. Every feature in this region is individually matched in image B, and votes for the candidate match if it matched consistently with p_B. Summing the votes of all the features in the verification region allows us to rank the candidate matches. A fixed percentage of the candidates are then considered verified. The choice of the verification region is, of course, important to the success of this verification step. The

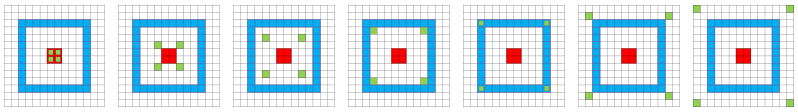


Figure 2.11: Different region configurations. (red: query regions, blue: verification regions, green: positive regions)

key aspect is that the features in the verification region should be, as much as possible, independent of the features in the proposal region. On the other hand, having them too far apart would reduce the chances of the region being completely matched. For our experiments, we used the 10x10 feature square centred around the query region (Figure 2.11).

Finally, given a set of verified correspondences, we have to decide which features to use as positive training pairs. One possibility would be to directly use features in the proposal region, since they have been verified. However, since the proposal region has already been "used" once (to verify the matches), it does not bring enough independent signal to make quality hard positives. Instead, we propose to sample positives from a different *positive region*. We evaluated different configurations for the positive region, as visualized in Figure 2.11 (in green). We choose to keep only 4 positive pairs per verified proposal, positioned at the corners of a square and denote the different setups as P2 to P14, the number corresponding to the size of the square. We showed that P12 and P14 perform better than the alternatives.

(2) Feature Fine-tuning. After each selection of positive feature pairs (Figure 2.10b in green), we update our feature (Figure 2.10c) using a single gradient step of the following triplet metric learning loss [86]:

$$L(P_1, P_2, \{F_i\}) = -\min(\lambda, s(P_1, P_2)) + \frac{1}{N_{neg}} \sum_{i=1}^{N_{neg}} \max(s(P_1, F_i), 1 - \lambda) \tag{2.7}$$

where P_1 and P_2 are corresponding features in the positive regions, $\{F_i\}_{\{i=1,2...N_{neg}\}}$ are negative samples, s is the cosine similarity metric and λ is a hyper-parameter. We select the negatives as the set of top matches to P_1 in P_2's image. This selects hard negatives and avoids any difference in the distribution of the depiction styles in our positive and negative samples. We chose a relatively high number of negatives N_{neg} to account for the fact that some of them might in fact correspond to matching regions, for example in the case of repeated elements, or for locations near the optimal match.

Spatially consistent pattern mining. We now explain how we discover and score repeated patterns in a dataset given style-invariant features.

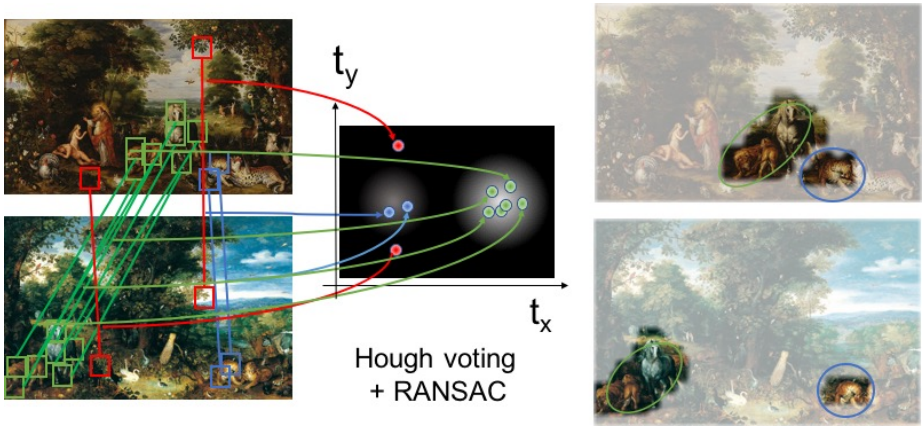


Figure 2.12: Discovery through geometric consistency.

Our discovery procedure for a pair of images is visualized in Figure 2.12 and follows a classic geometric verification approach [75]. We start by computing dense correspondences between the two images using our learned feature. These will be quite noisy. We then use Hough voting to identify potential groups of consistent correspondences. As a first approximation, each correspondence votes for a translation and change in scale. We finally extract the top 10 translation candidates, and, using a permissive threshold, focus on the correspondences in each group independently. Within each group, we use RANSAC to recover an affine transformation and the associated inliers. This allows to account for some deformation in the copy process, but also variations in the camera viewpoint with respect to the artwork.

After deformations between image regions are identified, we score the correspondence based both on the quality of the match between the features and geometric criteria. We use the following weighted average of the feature similarity:

$$S(\mathcal{I}) = \frac{1}{N} \sum_{i \in \mathcal{I}} e^{(-\frac{e_i^2}{2\sigma^2})} s_i \tag{2.8}$$

where $\mathcal{I}$ is the index of the inlier correspondences, e_i is the error between correspondence i and the geometric model, s_i the similarity of the associated descriptors and $\frac{1}{N}$ is normalization by the number of features in the source image.

Experiments. We performed most of our experiments on the Brueghel dataset [1], which contains 1,587 artworks done in different media (e.g. oil, ink, chalk, watercolour) and on different materials (e.g. paper, panel, copper), describing a wide variety of scenes (e.g. landscape, religious, still life). This dataset is especially adapted for our task since it assembles paintings from artists related to the same workshop, who thus had many interaction with each other, and includes many copies, preparatory drawings,

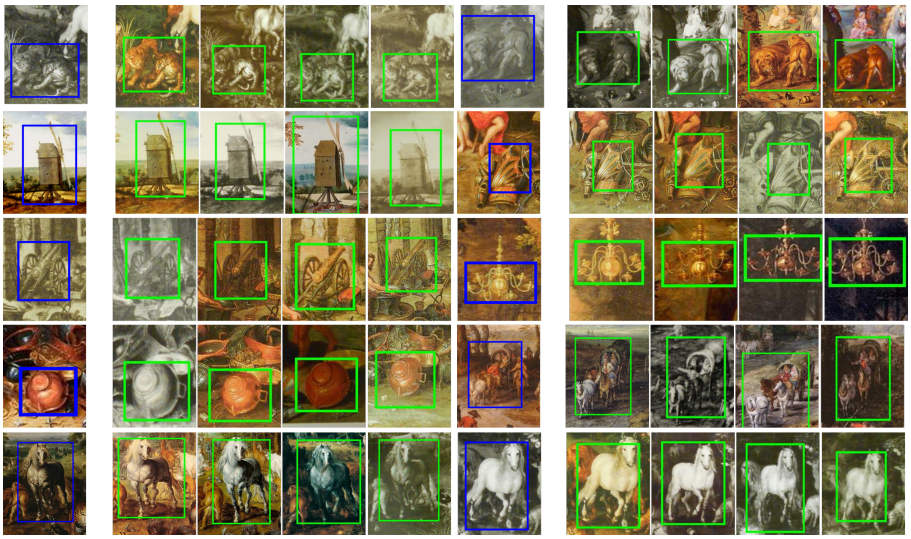


Figure 2.13: Detection example with our trained features on the Brueghel dataset. We show the top 4 matches (in green) for one example of query from each of our 10 annotated categories. Notice how the matches style can be different from the one of the query.

and borrowed details. With the help of our art history collaborators, we selected 10 of the most commonly repeated details in the dataset and annotated the visual patterns in the full dataset.

Figure 2.14: From a single query, shown on the left, we show the detection results obtained with cosine similarity with ImageNet feature (a) and our trained features (b) as well as the ones (c) obtained with our features and our discovery score (eq. 2.8).

Feature \ Method	Cosine similarity	Discovery score
ImageNet pre-taining	58.0	54.8
Context Prediction [27]	58.8	64.29
Ours (trained on Brueghel)	**75.3**	**76.4**
Ours (trained on LTLL)	65.2	69.95

Table 2.2: Experimental results on Brueghel, IoU > 0.3.

Method	LTLL (%)	Oxford (%)
B. Fernando et al.[34]	56.1	-
F. Radenović et al.[78]	-	87.8
ResNet18 max-pool, image level	59.8	14.0
ResNet18 + discovery	80.9	85.0
Ours (trained LTLL + discovery)	**88.5**	83.6
Ours (trained Oxford + discovery)	85.6	**85.7**

Table 2.3: Classification accuracy on LTLL and retrieval mAP on Oxford5K

We first evaluated our feature learning strategy for one-shot detection. Examples results for each of the 10 details we annotated on the Brueghel dataset are shown in Figure 2.13. It gives a sense of the difficulty of the task we target and the quality of the results we obtain. Note for example how the matches are of different styles, and how the two types of lions (top row) and the two types of horses (bottom row) are differentiated.

The benefits of our approach are outlined in Figure 2.14, where we present the top 6 matches from the same query using different approaches. On this example, it can be seen that while ImageNet feature only gets the matches in similar styles, our trained feature obtains duplicated horses in different styles, showing that the learned feature is more invariant to style. Moreover, the matching can still be improved with the discovery score. The corresponding quantitative results are presented in Table 2.2 and confirm these observations. Indeed learning features improves the score by 17%. The discovery procedure and score provides a small additional boost, which is a first validation of our discovery procedure. As a baseline, we also report the performance obtained deep feature learned with Context Prediction [27]. Interestingly, training our feature on the LTLL dataset also gave a boost in performance compared to the ImageNet feature, but is clearly worse than training on the Brueghel data, showing the dataset specific nature of our training.

Our results for discovery are harder to evaluate quantitatively on artworks. We show examples of our discovery results in Figure 2.15 and more results are available in our project webpage `http://imagine.enpc.fr/~shenx/ArtMiner/`. To obtain a quantitative measure, we evaluated our learned features for geolocalization on two datasets, the classical Oxford buildings [75] dataset and the LTLL [34] dataset. The LTLL [34] dataset contains historic and modern photographs captured from 25 locations over a time interval of more than 150 years. The task proposed in LTLL is to recognize the location of an old picture using annotated modern photographs, thus across two different modalities. We use our discovery procedure to find the images

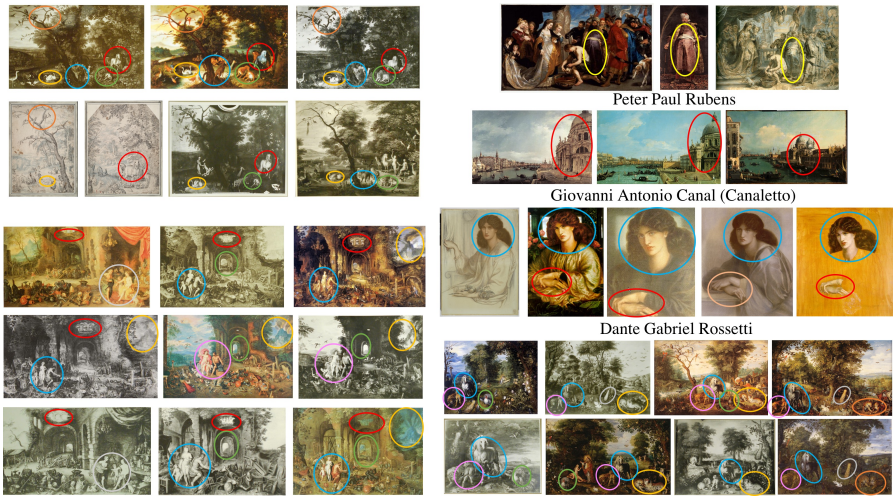


Figure 2.15: Example of image clusters discovered by our method. The three clusters without artist names correspond to data from the Brueghel dataset [1] and the three clusters with artist named were obtained using all the works of an artist available on WikiArt [2, 3].

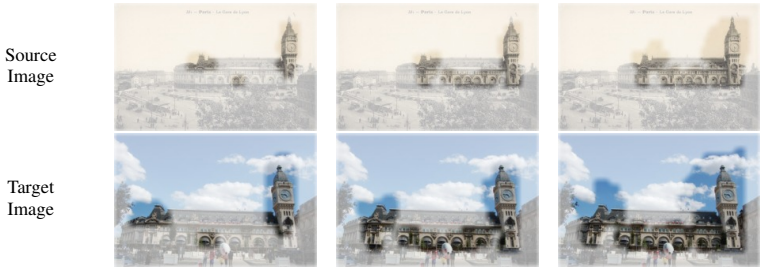


Figure 2.16: Discovery between images during training.

most similar to the query. The results are reported in Table 2.3. We compare our discovery score to cosine similarity with standard max-pooled features as well as the state of the art results of [34] on LTLL and [78] on Oxford5K. On LTLL, using the discovery score provides a very important boost compared to the results of [34] and the max-pooled features, and using our fine-tuning procedure on the LTLL dataset improves again the results, demonstrating again the interest of our proposed dataset specific fine-tunning procedure.

Similarly, on the Oxford5K dataset, we obtain an important boost using the discovery score compared to cosine similarity with max-pooled features. Fine-tuning the features on Oxford5K improves the mAP by 0.7%. This improvement is less important than on LTLL, which is expected since there is no specific domain gap between queries and targets in the Oxford5K dataset. Our result on Oxford5K is also comparable to

the state-of-the-art result obtained in [78] which performs fine-tuning on a large image collection with ResNet101. As expected the retrieval mAP is better when fine-tuning on the Oxford dataset than on LTLL.

To better understand the effect of our feature training, we visualise its influence on a pairs of matching images from the LTLL dataset in Figure 2.16. During training, larger and larger parts of the images can be matched in a consistent way, and be discovered as similar elements by our method. This shows both the efficiency of our feature training and its relevance for cross-modality matching.

2.4 Discussion.

In this Chapter, we have discussed three different approaches to apply deep learning to images from different modalities. It remains to an open problem to decide which one is more adapted for which problem and all three can be valid options. For example [77] applied successfully an explicit domain adaptation in feature space, similar to the one we presented in Section 2.1 to the problem of 3D pose estimation that we tackled through domain randomization in Section 2.2.

I am also still working on extending these approaches through different collaborations. Recently, we extended the relative pose estimation approach of Section 2.2 to multiple objects and used it to perform re-arrangement planning [87]. Finally in an ongoing work we applied all three approaches described above to the problem of historical watermark recognition from a single drawing and found that an extension of the approach presented in Section 2.3 provided much better results.

Chapter 3

Vector objects generation

In this chapter, we present methods to reconstruct 3D shapes and images from parametric - or 'vector' - primitives. We start by introducing the key idea of using neural networks as a tool to learn families of parametric functions on the example of deformable shapes. Once trained, the network take two types of input, (i) a set of parameters (ii) a vector in the unit cube. We demonstrate the interest of this approach by showing it provides state of the art shape correspondence results. We then generalize this idea to reconstruct any 3D surface by deforming, in a continuous way, a set of 2D unit squares. This lead to state of the art single-view 3D object reconstruction results. Finally, we develop a similar approach for images, decomposing them as a superposition of vector layers and we show how this decomposition can be used for image editing.

3.1 Learning template-based shape parametrization.

In this section, we tackle the problem of reconstructing a set of shapes by the deformation of a common template. Our key idea is that a Multi Layer Perceptron (MLP) can represent a family of deformations of the unit cube. The correspondences given by our learned deformation improve state-of-the art results on the FAUST dataset.

The work presented here was done by my PhD student Thibault Groueix and presented at ECCV 2018 [40].

Learning 3D shape correspondences by template deformation. Our shape matching approach has three main steps which are visualized figure 3.1. First an encoder network predicts coarse template deformation parameters for a given test shape. Second, these parameters are optimized locally so that the template deformation reconstructs as well as possible the test shape. Finally, correspondences are estimated using nearest neighbour between the test shape and its reconstruction. We will now describe in more details (1) our network architecture, (2) our training losses, (3) our shape refinement and (4) how we use our architecture to extract correspondences.

Network architecture.

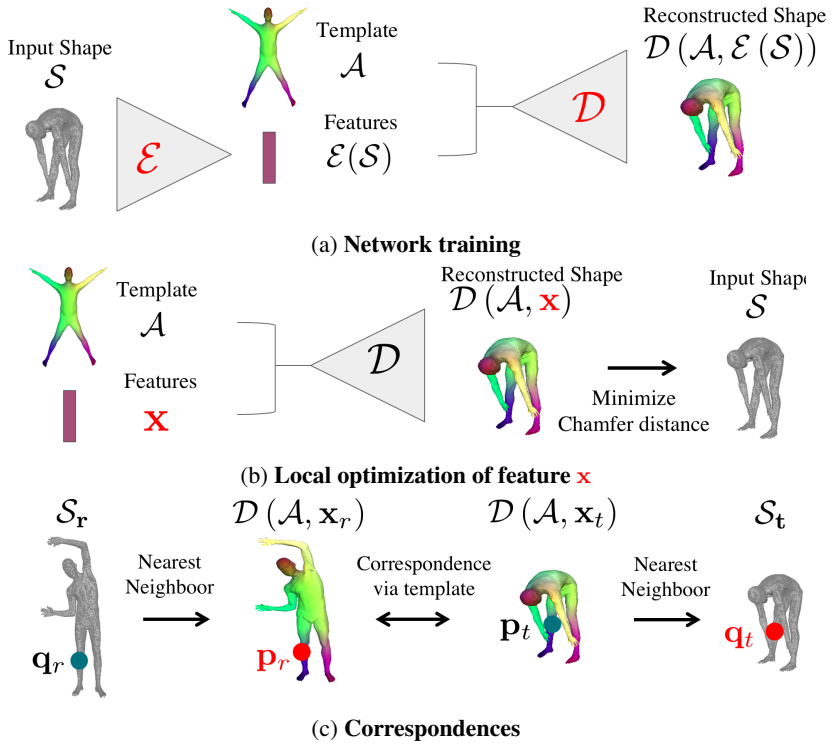


Figure 3.1: **Method overview.** (a) A feed-forward pass in our autoencoder encodes input point cloud $\mathcal{S}$ to latent code $\mathcal{E}(\mathcal{S})$ and reconstruct $\mathcal{S}$ using $\mathcal{E}(\mathcal{S})$ to deform the template $\mathcal{A}$. (b) We refine the reconstruction $\mathcal{D}(\mathcal{A}, \mathcal{E}(\mathcal{S}))$ by performing a regression step over the latent variable $\mathbf{x}$, minimizing the Chamfer distance between $\mathcal{D}(\mathcal{A}, \mathbf{x})$ and $\mathcal{S}$. (c) Finally, given two point clouds $\mathcal{S}_\mathbf{r}$ and $\mathcal{S}_\mathbf{t}$, to match a point $\mathbf{q}_r$ on $\mathcal{S}_\mathbf{r}$ to a point $\mathbf{q}_t$ on $\mathcal{S}_\mathbf{t}$, we look for the nearest neighbor $\mathbf{p}_r$ of $\mathbf{q}_r$ in $\mathcal{D}(\mathcal{A}, \mathbf{x}_r)$, which is by design in correspondence with $\mathbf{p}_t$; and look for the nearest neighbor $\mathbf{q}_t$ of $\mathbf{p}_t$ on $\mathcal{S}_\mathbf{t}$. Red indicates what is being optimised.

Formally, our goal is to put an input shape $\mathcal{S}$ in correspondence with a template $\mathcal{A}$. We do so by training an encoder-decoder architecture. The encoder $\mathcal{E}_\phi$ defined by its parameters ϕ takes as input 3D points from $\mathcal{S}$, and is a simplified version of the network presented in [76]. It applies to each input 3D point coordinate a multi-layer perceptron with hidden feature size of 64, 128 and 1024, then maxpooling over the resulting features over all points followed by a linear layer, leading to feature of size 1024 $\mathcal{E}_\phi(\mathcal{S})$. This feature, together with the 3D coordinates of a point on the template $\mathbf{p} \in \mathcal{A}$, are taken as input to the decoder $\mathcal{D}_\theta$ with parameters θ, which is trained to predict the position $\mathbf{q}$ of the corresponding point in the input shape $\mathcal{S}$. This decoder Shape Deformation Network is a multi-layer perceptron with hidden layers of size 1024, 512, 254 and 128, followed by a hyperbolic tangent. This architecture maps any points from the template domain to the reconstructed surface. By sampling the

template more or less densely, we can generate an arbitrary number of output points by sequentially applying the decoder over sampled template points.

This encoder-decoder architecture is trained end-to-end. We assume that we are given as input a training set of N shapes $\left\{\mathcal{S}^{(i)}\right\}_{i=1}^{N}$ with each shape having a set of P vertices $\{\mathbf{q}_j\}_{j=1}^{P}$. We consider two training scenarios: one where the correspondences between the template and the training shapes are known (supervised case) and one where they are unknown (unsupervised case). Supervision is typically available if the training shapes are generated by deforming a parametrized template, but real object scans are typically obtained without correspondences.

Training losses.

In the supervised case, we assume that for each point $\mathbf{q}_j$ on a training shape we know the correspondence $\mathbf{p}_j \leftrightarrow \mathbf{q}_j$ to a point $\mathbf{p}_j \in \mathcal{A}$ on the template $\mathcal{A}$. Given these training correspondences, we learn the encoder $\mathcal{E}_\phi$ and decoder $\mathcal{D}_\theta$ by simply optimizing the following reconstruction losses,

$$\mathcal{L}^{\text{sup}}(\theta, \phi) = \sum_{i=1}^{N} \sum_{j=1}^{P} |\mathcal{D}_\theta \left(\mathbf{p}_j; \mathcal{E}_\phi \left(\mathcal{S}^{(i)}\right)\right) - \mathbf{q}_j^{(i)}|^2 \tag{3.1}$$

where the sums are over all P vertices of all N example shapes.

In the case where correspondences between the exemplar shapes and the template are not available, we optimize the reconstructions, but also regularize the deformations toward isometries. For reconstruction, we use the Chamfer distance $\mathcal{L}^{\text{CD}}$ between the inputs $\mathcal{S}_i$ and reconstructed point clouds $\mathcal{D}_\theta \left(\mathcal{A}; \mathcal{E}_\phi \left(\mathcal{S}^{(i)}\right)\right)$. For regularization, we use two different terms. The first term $\mathcal{L}^{\text{Lap}}$ encourages the Laplacian operator defined on the template and the deformed template to be the same (which is the case for isometric deformations of the surface). The second term $\mathcal{L}^{\text{edges}}$ encourages the ratio between edges length in the template and its deformed version to be close to 1. More details on these different losses are given in the supplementary material of our paper [40]. The final loss we optimize is:

$$\mathcal{L}^{\text{unsup}} = \mathcal{L}^{\text{CD}} + \lambda_{Lap}\mathcal{L}^{\text{Lap}} + \lambda_{edges}\mathcal{L}^{\text{edges}} \tag{3.2}$$

where λ_{Lap} and λ_{edges} control the influence of regularizations against the data term $\mathcal{L}^{\text{CD}}$. They are both set to 5.10^{-3} in our experiments.

We optimize the loss using the Adam solver, with a learning rate of 10^{-3} for 25 epochs then 10^{-4} for 2 epochs, batches of 32 shapes, and 6890 points per shape.

One interesting aspect of our approach is that it learns jointly a parameterization of the input shapes via the decoder and to perdict the parameters $\mathcal{E}_\phi\left(\mathcal{S}\right)$ for this parameterization via the encoder. However, the predicted parameters $\mathcal{E}_\phi\left(\mathcal{S}\right)$ for an input shape $\mathcal{S}$ are not necessarily optimal, because of the limited power of the encoder. Optimizing these parameters turns out to be important for the final results, and is the focus of the second step of our pipeline.

Algorithm 1: Algorithm for finding 3D shape correspondences

Input : Reference shape $\mathcal{S}_r$ and target shape $\mathcal{S}_t$
Output: Set of 3D point correspondences $\mathcal{C}$

#Regression steps over latent code to find best reconstruction of $\mathcal{S}_r$ and $\mathcal{S}_t$
$\mathbf{x}_r \leftarrow_{\mathbf{x}} \mathcal{L}^{\text{CD}}(\mathbf{x}; \mathcal{S}_r)$ #detailed in equation (3.3)
$\mathbf{x}_t \leftarrow_{\mathbf{x}} \mathcal{L}^{\text{CD}}(\mathbf{x}; \mathcal{S}_t)$ #detailed in equation (3.3)
$\mathcal{C} \leftarrow \varnothing$
Matching of $\mathbf{q}_r \in \mathcal{S}_r$ to $\mathbf{q}_t \in \mathcal{S}_t$
foreach $\mathbf{q}_r \in \mathcal{S}_r$ **do**
    $\mathbf{p} \leftarrow_{\mathbf{p}' \in \mathcal{A}} |\mathcal{D}_\theta(\mathbf{p}'; \mathbf{x}_r) - \mathbf{q}_r|^2$
    $\mathbf{q}_t \leftarrow_{\mathbf{q}' \in \mathcal{S}_t} |\mathcal{D}_\theta(\mathbf{p}; \mathbf{x}_t) - \mathbf{q}'|^2$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\mathbf{q}_r, \mathbf{q}_t)\}$
end
return $\mathcal{C}$

Optimizing shape reconstruction.

We now assume that we are given a shape $\mathcal{S}$ as well as learned weights for the encoder $\mathcal{E}_\phi$ and decoder $\mathcal{D}_\theta$ networks. To find correspondences between the template shape and the input shape, we will use a nearest neighbor search to find correspondences between that input shape and its reconstruction. For this step to work, we need the reconstruction to be accurate. The reconstruction given by the parameters $\mathcal{E}_\phi(\mathcal{S})$ is only approximate and can be improved. Since we do not know correspondences between the input and the generated shape, we cannot minimize the loss given in equation (3.1), which requires correspondences. Instead, we minimize with respect to the global feature $\mathbf{x}$ the Chamfer distance between the reconstructed shape and the input:

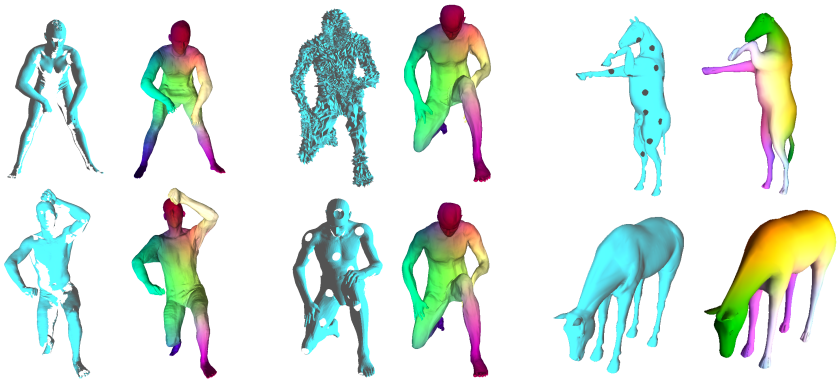
$$\mathcal{L}^{\text{CD}}(\mathbf{x}; \mathcal{S}) = \sum_{\mathbf{p} \in \mathcal{A}} \min_{\mathbf{q} \in \mathcal{S}} |\mathcal{D}_\theta(\mathbf{p}; \mathbf{x}) - \mathbf{q}|^2 + \sum_{\mathbf{q} \in \mathcal{S}} \min_{\mathbf{p} \in \mathcal{A}} |\mathcal{D}_\theta(\mathbf{p}; \mathbf{x}) - \mathbf{q}|^2 . \tag{3.3}$$

Starting from the parameters predicted by our first step $\mathbf{x} = \mathcal{E}_\phi(\mathcal{S})$, we optimize this loss using the Adam solver for 3,000 iterations with learning rate $5 * 10^{-4}$. Note that the good initialization given by our first step is key since Equation (3.3) corresponds to a highly non-convex problem.

Finding 3D shape correspondences.

To recover correspondences between two 3D shapes $\mathcal{S}_r$ and $\mathcal{S}_t$, we first compute the parameters to deform the template to these shapes, $\mathbf{x}_r$ and $\mathbf{x}_t$, using the procedure outlined above. Next, given a 3D point $\mathbf{q}_r$ on the reference shape $\mathcal{S}_r$, we first find the point $\mathbf{p}$ on the template $\mathcal{A}$ such that its transformation with parameters $\mathbf{x}_r$, $\mathcal{D}_\theta(\mathbf{p}; \mathbf{x}_r)$ is closest to $\mathbf{q}_r$. Finally we find the 3D point $\mathbf{q}_t$ on the target shape $\mathcal{S}_t$ that is the closest to the transformation of $\mathbf{p}$ with parameters $\mathbf{x}_t$, $\mathcal{D}_\theta(\mathbf{p}; \mathbf{x}_t)$. Our algorithm is summarized in Algorithm 1 and illustrated in Figure 3.1.

Results. The method presented above leads to the best results to date on the FAUST-inter dataset: 2.878 cm : **an improvement of 8% over state of the art**, 3.12cm for



(a) SCAPE [6] (b) TOSCA [15] (c) TOSCA animals [15]

Figure 3.2: **Other datasets.** Left images show the input, right images the reconstruction with colors showing correspondences. Our method works with real incomplete scans **(a)**, strong synthetic perturbations **(b)**, and on non-human shapes **(c)**.

[106] and 4.82cm for [59]. Although it cannot take advantage of the fact that two meshes represent the same person, our method is also the second best performing (average error of 1.99 cm) on FAUST-intra challenge.

The SCAPE dataset provides meshes aligned to real scans and includes poses different from our training dataset. When applying a network trained directly on our SMPL data, we obtain satisfying performance, namely 3.14cm average Euclidean error. Quantitatively, we outperform all methods except for Deep Functional Maps [59]. SCAPE also allows evaluation on real partial scans. Quantitatively, the error on these partial meshes is 4.04cm, similar to the performance on the full meshes. Qualitative results are shown in Fig 3.2a.

The TOSCA dataset provides several versions of the same synthetic mesh with different perturbations. We found that our method, still trained only on SMPL or SMAL data, is robust to all perturbations (isometry, noise, shotnoise, holes, micro-holes, topology changes, and sampling), except scale, which can be trivially fixed by normalizing all meshes to have consistent surface area. Examples of representative qualitative results are shown Fig 3.2b.

Method	Faust error (cm)
Without regression	6.29
With regression	3.255
With regression + Regular Sampling	3.048
With regression + Regular Sampling + High-Res template	**2.878**

Table 3.1: **Importance of the reconstruction optimization step.** Optimizing the latent feature is key to our results. Regular point sampling for training and high resolution for the nearest neighbor step provide an additional boost.

Loss	Faust error (cm)
Chamfer distance, eq. 3.5 (unsupervised)	8.727
Chamfer distance + Regularization, eq. 3.2 (unsupervised)	4.835
Correspondences, eq. 3.1 (supervised)	**2.878**

Table 3.2: Results with and without supervised correspondences. Adding regularization helps the network find a better local minimum in terms of correspondences.

Because the nearest neighbors used in the matching step are sensitive to small errors in alignment, the second step of our pipeline which finds the optimal features for reconstruction, is crucial to obtain high quality results. This optimization however converges to a good optimum only if it is initialized with a reasonable reconstruction. Since we optimize using Chamfer distance, and not correspondences, we also rely on the fact that the network was trained to generate humans in correspondence and we expect the optimized shape to still be meaningful.

Table 3.1 reports the associated quantitative results on FAUST-inter. We can see that: (i) optimizing the latent feature to minimize the Chamfer distance between input and output provides a strong boost; (ii) using a better (more uniform) sampling of the shapes when training our network provided a better initialization; (iii) using a high resolution sampling of the template ($\sim$200k vertices) for the nearest-neighbor step provide an additional small boost in performance.

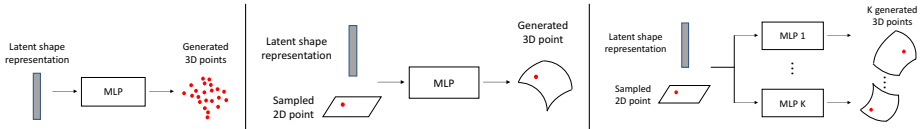
Finally, we investigate whether our method could be trained without correspondence supervision. We started by simply using the reconstruction loss but we found that the resulting deformations did not respect correspondences between the template and the input shape. However, these results improve with adequate regularization such as the one presented in Equarion (3.2), encouraging regularity of the mapping between the template and the reconstruction. We trained such a network with the same training data as in the supervised case but **without any correspondence supervision** and obtained a 4.88cm of error on the FAUST-inter data, i.e. similar to Deep Functional Map [59] which had an error of 4.83 cm. This demonstrates that our method can be efficient even without correspondence supervision.

3.2 Learning local shape parametrization.

The "surface reconstruction by deformation" approach presented in the previous section relies on a common shape template. We now extend this idea to buid an architectures, which can reconstruct any surface by deforming a set of primitive 2D patches. The resulting local parametrization of the surface is similar to an atlas, and we called the associated architecture AtlasNet

This work was also done by Thibault Groueix, and presented at CVPR 2018 [39].

Idea of the approach. We start by introducing some vocabulary about surfaces, formalize the core idea of our approach and formulate a couple of associated properties.



(a) Points baseline. (b) Our approach with one patch. (c) Our approach with K patches.

Figure 3.3: **Shape generation approaches.** All methods take as input a latent shape representation (that can be learned jointly with a reconstruction objective) and generate as output a set of points. (a) A baseline deep architecture would simply decode this latent representation into a set of points of a given size. (b) Our approach takes as additional input a 2D point sampled uniformly in the unit square and uses it to generate a single point on the surface. Our output is thus the continuous image of a planar surface. In particular, we can easily infer a mesh of arbitrary resolution on the generated surface elements. (c) This strategy can be repeated multiple times to represent a 3D shape as the union of several surface elements.

A subset $\mathcal{S}$ of $\mathbb{R}^3$ is a *2-manifold* if, for every point $\mathbf{p} \in \mathcal{S}$, there is an open set U in $\mathbb{R}^2$ and an open set W in $\mathbb{R}^3$ containing $\mathbf{p}$ such that $\mathcal{S} \cap W$ is homeomorphic to U. The set homeomorphism from $\mathcal{S} \cap W$ to U is called a *chart*, and its inverse a *parameterization*. A set of charts such that their images cover the 2-manifold is called an *atlas* of the 2-manifold. The ability to learn an atlas for a 2-manifold would allow a number of applications, such as transfer of a tessellation to the 2-manifold for meshing and texture mapping (via texture atlases). Here, we use the word *surface* in a slightly more generic sense than *2-manifold*, allowing for self-intersections and disjoint sets. Let us consider a 2-manifold $\mathcal{S}$, a point $\mathbf{p} \in \mathcal{S}$ and a parameterization φ of $\mathcal{S}$ in a local neighborhood of $\mathbf{p}$. We can assume that φ is defined on the open unit square $]0,1[^2$ by first restricting φ to an open neighborhood of $\varphi^{-1}(\mathbf{p})$ with disk topology where it is defined (which is possible because φ is continuous) and then mapping this neighborhood to the unit square.

We pose the problem of learning to generate the local 2-manifold previously defined as one of finding a parameterizations $\varphi_\theta(x)$ with parameters θ which map the open unit 2D square $]0,1[^2$ to a good approximation of the desired 2-manifold $\mathcal{S}_{\text{loc}}$. Specifically, calling $\mathcal{S}_\theta = \varphi_\theta(]0,1[^2)$, we seek to find parameters θ minimizing the following objective function,

$$\min_\theta \mathcal{L}\left(\mathcal{S}_\theta, \mathcal{S}_{\text{loc}}\right) + \lambda \mathcal{R}\left(\theta\right), \tag{3.4}$$

where $\mathcal{L}$ is a loss over 2-manifolds, $\mathcal{R}$ is a regularization function over parameters θ, and λ is a scalar weight. In practice, instead of optimizing a loss over 2-manifolds $\mathcal{L}$, we optimize a loss over point sets sampled from these 2-manifolds such as Chamfer and Earth-Mover distance.

Similar to the previous section, we represent φ_θ using multilayer perceptrons (MLPs) with rectified linear unit (ReLU) nonlinearities. Indeed, we now show that this family of function almost verifies two key properties: (i) generate 2-manifolds and (ii) be able to produce a good approximation of the desired 2-manifolds S_{loc}.

Proposition 1. *Let f be a multilayer perceptron with ReLU nonlinearities. There exists a finite set of polygons P_i, $i \in \{1, ..., N\}$ such that on each P_i f is an affine function: $\forall x \in P_i,\ f(x) = A_i x + b$, where A_i are 3×2 matrices. If for all i, $\mathrm{rank}(A_i) = 2$, then for any point $\mathbf{p}$ in the interior of one of the P_is there exists a neighborhood $\mathcal{N}$ of $\mathbf{p}$ such that $f(\mathcal{N})$ is a 2-manifold.*

Proof. The fact that f is locally affine is a direct consequence of the fact that we use ReLU non-linearities. If $\mathrm{rank}(A_i) = 2$ the inverse of $A_i x + b$ is well defined on the surface and continuous, thus the image of the interior of each P_i is a 2-manifold. □

To draw analogy to texture atlases in computer graphics, we call the local functions we learn to approximate a 2-manifold *learnable parameterizations* and the set of these functions A a *learnable atlas*. Note that in general, an MLP locally defines a rank 2 affine transformation and thus locally generates a 2-manifold, but may not globally as it may intersect or overlap with itself. The second reason to choose MLPs as a family is that they can allow us to approximate any continuous surface.

Proposition 2. *Let S be a 2-manifold that can be parameterized on the unit square. For any $\epsilon > 0$ there exists an integer K such that a multilayer perceptron with ReLU non linearities and K hidden units can approximate S with a precision ϵ.*

Proof. This is a consequence of the universal representation theorem [47] □

Learning to decode a surface. More concretely, our goal is, given a feature representation $\mathbf{x}$ for a 3D shape, to generate the surface of the shape. As shown above, an MLP with ReLUs φ_θ with parameters θ can locally generate a surface by learning to map points in $\mathbb{R}^2$ to surface points in $\mathbb{R}^3$. To generate a given surface, we need several of these learnable charts to represent a surface. In practice, we consider N learnable parameterizations ϕ_{θ_i} for $i \in \{1, ..., N\}$. Our model is illustrated in Figure 3.3c. Notice that the MLPs are not explicitly prevented from encoding the same area of space, but their union should cover the full shape. Our MLPs do depend on the random initialization, but similar to convolutional filter weights the network learns to specialize to different regions in the output without explicit biases.

Let $\mathcal{A}$ be a set of points sampled in the unit square $[0, 1]^2$ and $\mathcal{S}^\star$ a set of points sampled on the target surface. We incorporate the shape feature $\mathbf{x}$ by simply concatenating them with the sampled point coordinates $\mathbf{p} \in \mathcal{A}$ before passing them as input to the MLPs. We train the local parametrizations ϕ_{θ_i} to minimize the Chamfer loss between the set of generated 3D points and $\mathcal{S}^\star$,

$$\mathcal{L}(\theta) = \sum_{\mathbf{p} \in \mathcal{A}} \sum_{i=1}^{N} \min_{\mathbf{q} \in \mathcal{S}^\star} \left| \phi_{\theta_i}(\mathbf{p}; \mathbf{x}) - \mathbf{q} \right|^2 + \sum_{\mathbf{q} \in \mathcal{S}^\star} \min_{i \in \{1, ..., N\}} \min_{\mathbf{p} \in \mathcal{A}} \left| \phi_{\theta_i}(\mathbf{p}; \mathbf{x}) - \mathbf{q} \right|^2. \tag{3.5}$$

We consider two concrete tasks: (i) to auto-encode a 3D shape given an input 3D point cloud, and (ii) to reconstruct a 3D shape given an input RGB image. For the auto-encoder, we used an encoder based on PointNet [76], similar to the one used in Section 3.1. For images, we used ResNet-18 [44] as our encoder. We experimented

Method	CD	Metro
Oracle 2500 pts	0.85	1.56
Oracle 125K pts	-	1.26
Points baseline	1.91	-
Points baseline + normals	2.15	1.82 (PSR)
Ours - 1 patch	1.84	1.53
Ours - 1 sphere	1.72	1.52
Ours - 5 patches	1.57	1.48
Ours - 25 patches	1.56	1.47
Ours - 125 patches	**1.51**	**1.41**

Table 3.3: **3D reconstruction.** Comparison of our approach against a point-generation baseline ("CD" - Chamfer distance, multiplied by 10^3, computed on 2500 points; "Metro" values are multiplied by 10). Note that our approach can be directly evaluated by Metro while the baseline requires performing PSR [51]. These results can be compared with an Oracle sampling points directly from the ground truth 3D shape followed by PSR (top two rows). See text for details.

with different basic weight regularization options but did not notice any generalization improvement. Sampling of the learned parameterizations as well as the ground truth point-clouds is repeated at each training step to avoid over-fitting. For single-view reconstruction, we obtained the best results by training the encoder and using the decoder from the point cloud autoencoder with fixed parameters. Finally, we noticed that sampling points regularly on a grid on the learned parameterization yields better performance than sampling points randomly. All results used this regular sampling.

Results. We now present some of our results. We start by introducing the data, measures and baseline we used. We then present an analysis of AtlasNet in the shape autoencoding framework. Finally, we present single-view recostruction results.

Data, measures and baseline.
We evaluated our approach on the standard ShapeNet Core dataset (v2) [17]. The dataset consists of 3D models covering 13 object categories with 1K-10K shapes per category. We used the training and validation split provided by [19] for our experiments to be comparable with previous approaches. We used the rendered views provided by [19] and sampled 3D points on the shapes using [98]. We evaluated our generated shape outputs by comparing to ground truth shapes using two criteria. First, we compared point sets for the output and ground-truth shapes using Chamfer distance ("CD"). While this criteria compares two point sets, it does not take into account the surface/mesh connectivity. To account for mesh connectivity, we compared the output and ground-truth meshes using the "Metro" criteria using the publicly available METRO software [21], which is the average Euclidean distance between the two meshes.

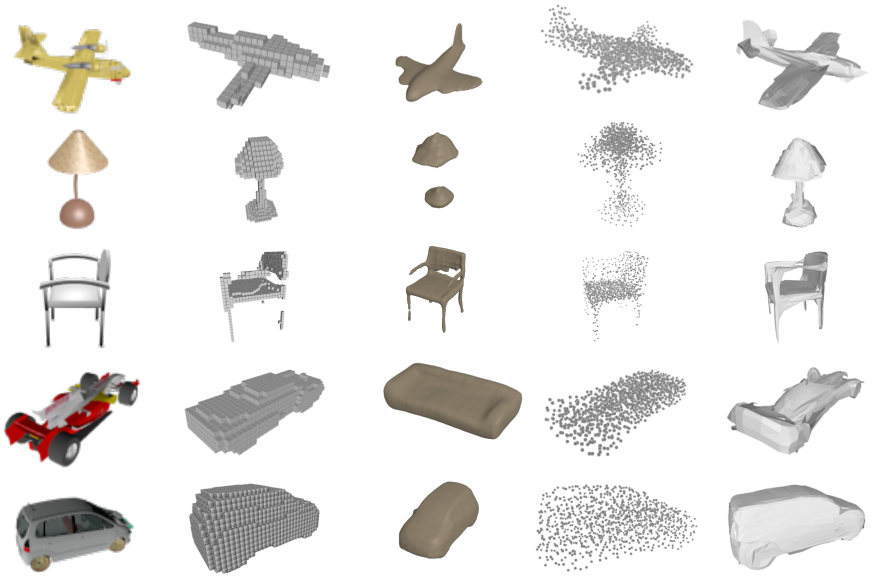
We always compare our approach to the multi-layer perceptron "Points baseline" network shown in Figure 3.3a. The Points baseline network consists of four fully

Category		Points baseline	Ours 1 patch	Ours 125 patches
chair	LOO	3.66	3.43	**2.69**
	All	1.88	1.97	**1.55**
car	LOO	3.38	2.96	**2.49**
	All	1.59	2.28	**1.56**
watercraft	LOO	2.90	2.61	**1.81**
	All	1.69	1.69	**1.23**
plane	LOO	6.47	6.15	**3.58**
	All	1.11	1.04	**0.86**

Table 3.4: **Generalization across object categories.** Comparison of our approach with varying number of patches against the point-generating baseline to generate a specific category when training on all other ShapeNet categories. Chamfer distance is reported, multiplied by 10^3, computed on 2500 points. Notice that our approach with 125 patches out-performs all baselines when generalizing to the new category. For reference, we also show performance when we train over all categories.

connected layers with output dimensions of size 1024, 512, 256, 7500 with ReLU non-linearities, batch normalization on the first three layers, and a hyperbolic-tangent non-linearity after the final fully connected layer. The network outputs 2500 3D points and has comparable number of parameters to our method with 25 learned parameterizations. The baseline architecture was designed to be as close as possible to the MLP used in AtlasNet. As the network outputs points and not a mesh, we also trained a second network that outputs 3D points and normals, which are then passed as inputs to Poisson surface reconstruction (PSR) [51] to generate a mesh ("Points baseline + normals"). The network generates outputs in $\mathbb{R}^6$ representing both the 3D spatial position and normal. We optimized Chamfer loss in this six-dimensional space and normalized the normals to 0.1 length as we found this trade-off between the spatial coordinates and normals in the loss worked best. As density is crucial to PSR quality, we augmented the number of points by sampling 20 points in a small radius in the tangent plane around each point [51]. We noticed significant qualitative and quantitative improvements and the results shown in this chapter use this augmentation scheme.

Analysis in the autoencoding framework.
We report quantitative results for shape generation from point clouds in Table 3.3, where each approach is trained over all ShapeNet categories and results are averaged over all categories. Notice that our approach out-performs the Points baseline on both the Chamfer distance and Metro criteria, even when using a single learned parameterization (patch). Also, the Points baseline + normals has worse Chamfer distance than the Points baseline without normals indicating that predicting the normals decreases the quality of the point cloud generation. We also report performance for two "oracle" outputs indicating upper bounds. The first oracle ("Oracle 2500 pts") randomly samples 2500 points+normals from the ground truth shape and applies PSR. The Chamfer distance between the random point set and the ground truth gives an upper bound on



(a) Input (b) 3D-R2N2 (c) HSP (d) PSG (e) Ours

Figure 3.4: **Single-view reconstruction comparison.** From a 2D RGB image (a), 3D-R2N2 [19] reconstructs a voxel-based 3D model (b), HSP [43] reconstructs a octree-based 3D model (c), PointSetGen [30] a point cloud based 3D model (d), and our AtlasNet a triangular mesh (e).

performance for point-cloud generation. Notice that our method out-performs the surface generated from the oracle points. The second oracle ("Oracle 125K pts") applies PSR on all 125K points+normals from the ground-truth shape. It is interesting to note that the Metro distance from this result to the ground truth is not far from the one obtained with our method. Finally, we show in Table 3.3 our approach with varying number of learnable parameterizations (patches) in the atlas. Notice how our approach improves as we increase the number of patches. Moreover, we also compare with an approach where we sampled points on the 3D unit sphere instead of 2D patches to obtain a closed mesh. Notice that sampling from a sphere quantitatively out-performs a single patch, but multiple patches perform better.

An important desired property of a shape auto-encoder is that it generalizes well to categories it has not been trained on. To evaluate this, we trained our method on all categories but one target category ("LOO") for chair, car, watercraft, and plane categories, and evaluated on the held-out category. The corresponding results are reported in Table 3.4. We also include performance when the methods are trained on all of the categories including the target category ("All") for comparison. Notice that we again out-perform the point-generating baseline on this leave-one-out experiment and that performance improves with more patches. The car category is especially interesting since when trained on all categories the baseline has better results than our method with 1 patch and similar to our method with 125 patches. If not trained on cars, both

	pla.	ben.	cab.	car	cha.	mon.	lam.	spe.	fir.	cou.	tab.	cel.	wat.	mean
Ba CD	2.91	4.39	6.01	4.45	7.24	**5.95**	7.42	10.4	1.83	**6.65**	4.83	4.66	**4.65**	5.50
PSG CD	3.36	4.31	8.51	8.63	**6.35**	6.47	7.66	15.9	**1.58**	6.92	**3.93**	**3.76**	5.94	6.41
Ours CD	**2.54**	**3.91**	**5.39**	**4.18**	6.77	6.71	**7.24**	**8.18**	1.63	6.76	4.35	3.91	4.91	**5.11**
Ours Metro	1.31	1.89	1.80	2.04	2.11	1.68	2.81	2.39	1.57	1.78	2.28	1.03	1.84	1.89

Table 3.5: **Single-View Reconstruction (per category).** The mean is taken category-wise. The Chamfer Distance reported is computed on 1024 points, after running ICP alignment with the GT point cloud, and multiplied by 10^3. The Metro distance is multiplied by 10.

our approaches clearly outperform the baseline, showing that at least in this case, our approach generalizes better than the baseline.

Single-View reconstruction

We evaluate the potential of our method for single-viewreconstruction. We compare qualitatively our results with three state-of-the-art methods, PointSetGen [30], 3D-R2N2 [19] and HSP [43] in Figure 5. To perform the comparison for PointSetGen [30] and 3D-R2N2 [19], we used the trained models made available online by the authors. For HSP [43], we asked the authors to run their method on the images presented in Figure 3.4. Figure 3.4 emphasizes the importance of the type of output (voxels for 3D-N2D2 and HSP, point cloud for PointSetGen, mesh for us) for the visual appearance of the results. Notice the small details visible on our meshes that may be hard to see on the unstructured point cloud or volumetric representation. Also, it is interesting to see that PointSetGen tends to generate points inside the volume of the 3D shape while our result, by construction, generates points on a surface.

To perform a quantitative comparison against PointSetGen [30], we evaluated the Chamfer distance between generated points and points from the original mesh for both PointSetGen and our method with 25 learned parameterizations. However, the PointSetGen network was trained with a translated, rotated, and scaled version of ShapeNet with parameters we did not have access to. We thus first had to align the point clouds resulting from PointSetGen to the ShapeNet models used by our algorithm. We randomly selected 260 shapes, 20 from each category, and ran the iterative closest point (ICP) algorithm [12] to optimize a similarity transform between PointSetGen and the target point cloud. Note that this optimization improves the Chamfer distance between the resulting point clouds, but is not globally convergent. We checked visually that the point clouds from PointSetGen were correctly aligned, and display all alignments on the project webpage[1]. To have a fair comparison we ran the same ICP alignment on our results. In Table 3.5 we compared the resulting Chamfer distance. Our method provides the best results on 6 categories whereas PointSetGen and the baseline are best on 4 and 3 categories, respectively. Our method is better on average and generates point clouds of a quality similar to the state of the art. We also report the Metro distance to the original shape, which is the most meaningful measure for our method.

[1] `http://imagine.enpc.fr/~groueixt/atlasnet/PSG.html`.

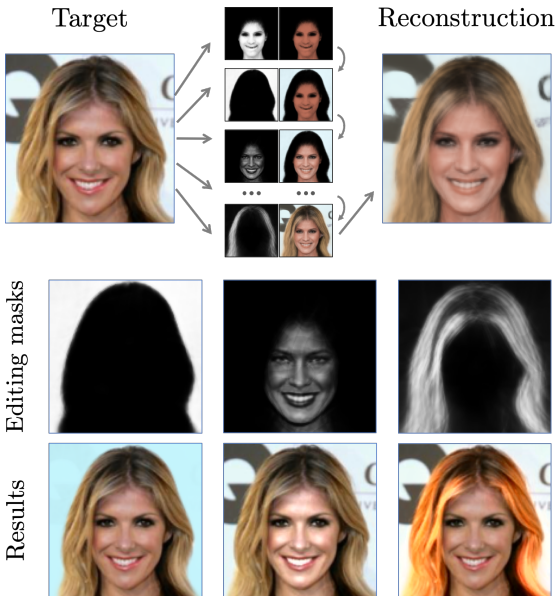


Figure 3.5: Our system learns in an unsupervised manner a decomposition of images as superimposed α-channel masks (top) that can be used for quick image editing (bottom).

3.3 Decomposing images in vector layers

In this section we show that the idea developed above to generate surfaces from parametric transformations of the unit square can be adapted to generate images from a set of vector layers. The main application of this new image generation paradigm we target and demonstrate is intuitive image editing. Indeed, our layered decomposition allows simple user interaction, for example to change the color of a selected layer. We also show we learn a representation useful for image search and that allows raster image vectorization

This work was done by my PhD student Othman Sbaï and a preliminary version is available on ArXiv [85].

Approach. Deep image generation models demonstrate breathtaking and inspiring results, e.g. [104, 105], but usually offer limited control and little interpretability. In contrast, we introduce and explore a new deep image generation paradigm, which follows an approach similar to the one used in interactive design tools. We formulate image generation as the composition of successive layers, each associated to a single color. Rather than learning high resolution image generation, we produce a vector decomposition of the image, that can easily be used to edit images at any resolution. We aim at enabling designers to easily build on the results of deep image generation methods, by editing layers individually, changing their characteristics, or intervening in the middle of the generation process.

We frame image generation as an alpha-blending composition of a sequence of lay-

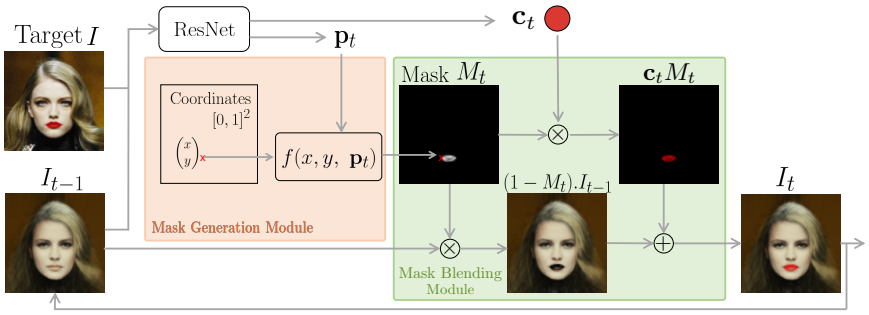


Figure 3.6: Our iterative generation pipeline for image reconstruction of target I. The previous canvas I_{t-1} (I_0 initialized to black) is concatenated with I and forwarded through a ResNet feature extractor, to obtain a color $\mathbf{c}_t$ and mask parameters $\mathbf{p}_t$. A Multi Layer Perceptron f generates a parametric mask M_t from pixelwise coordinates of a 2D grid and mask parameters $\mathbf{p}_t$. Our Mask Blending Module (in green) finally blends this mask with its corresponding color to the previous output I_{t-1}.

ers starting from an empty (black) canvas I_0. Given a fixed budget of T iterations, we iteratively blend T generated colored masks onto the canvas. We first present (1) our new architecture for vector image generation, then (2) the training loss and finally (3) discuss the advantages of our new architecture compared to existing approaches.

Architecture.
The core idea of our approach is visualized in Fig. 3.6. At each iteration $t \in \{1...T\}$, our model takes as input the concatenation of the target image $I \in \mathbb{R}^{3\times W\times H}$ and the current canvas I_t, and iteratively blends colored masks on the canvas resulting in I_t:

$$I_t = g(I_{t-1}, I), \tag{3.6}$$

where g consists of:

(i) a Residual Network (ResNet) that predicts mask parameters $\mathbf{p}_t \in \mathbb{R}^P$, with the corresponding color triplet $\mathbf{c}_t \in \mathbb{R}^3$,

(ii) a mask generator module f, which generates an alpha-blending mask M_t from the parameters $\mathbf{p}_t$, and

(iii) our mask blending module that blends the masks M_t with their color $\mathbf{c}_t$ on the previous canvas I_{t-1}.

We represent the function f generating the mask M_t from $\mathbf{p}_t$ as a standard Multi-Layer Perceptron (MLP), which takes as input the concatenation of the mask parameters $\mathbf{p}_t$ and the two spatial coordinates (x, y) of a point in image space. This MLP f defines the continuous 2D function of the mask M_t by:

$$M_t(x, y) = f(x, y, \mathbf{p}_t). \tag{3.7}$$

In practice, we evaluate the mask at discrete spatial locations corresponding to the desired resolution to produce a discrete image. We then update I_t at each spatial location (x, y) using the following blending:

$$I_t(x, y) = I_{t-1}(x, y).(1 - M_t(x, y)) + \mathbf{c}_t.M_t(x, y), \tag{3.8}$$

where $I_t(x, y) \in \mathbb{R}^3$ is the RGB value of the resulting image I_t at position (x, y). We note that, at test time, we may perform a different number of iterations N than the one during training T. Choosing $N > T$ may help to model accurately images that contain complex patterns, as we show in our experiments.

Concretely, the mask generator f consists of an MLP with three hidden layers of 128 units with group normalization [99], tanh non-linearities, and a sigmoid after the last layer. f takes as input a parameter vector $\mathbf{p}$ and pixel coordinates (x, y), and outputs a value between 0 and 1. The parameter $\mathbf{p}$ and the color $\mathbf{c}$ are predicted by a ResNet-18 network.

Training losses.
We learn the weights of our network end-to-end by minimizing a reconstruction loss between the target I and our result R. We perform experiments either using an ℓ_1 loss, which enables simple quantitative comparisons, or a perceptual loss [50], leading to visually improved results. Our perceptual loss $\mathcal{L}_{perc}$ is based on the Euclidean norm $\|.\|_2$ between feature maps $\phi(.)$ extracted from a pre-trained VGG16 network and the Frobenius norm between the Gram matrices obtained from these feature maps $G(\phi(.))$:

$$\mathcal{L}_{perc} = \mathcal{L}_{content} + \lambda \mathcal{L}_{style},$$

where

$$\mathcal{L}_{content}(I, R) = \|\phi(I) - \phi(R)\|_2 ,$$
$$\mathcal{L}_{style}(I, R) = \|G(\phi(I)) - G(\phi(R))\|_F ,$$

and λ is a non-negative scalar that controls the relative influence of the style loss.

Discussion.
Our architecture choices are related to desirable properties of the final generation model:

- Layered decomposition: This choice allows us to obtain a mask decomposition which is a key component of image editing pipelines. Defining one color per layer, similar to image simplification and quantization approaches, is important to obtain visually coherent regions. We further show that a single layer baseline does not perform as well.

- Vectorized layers: By using a lattice input for the mask generator, it is possible to perform local image editing and generation at any resolution without introducing up-sampling artifacts or changing our model architecture. This vector mask representation is especially convenient for HD image editing.

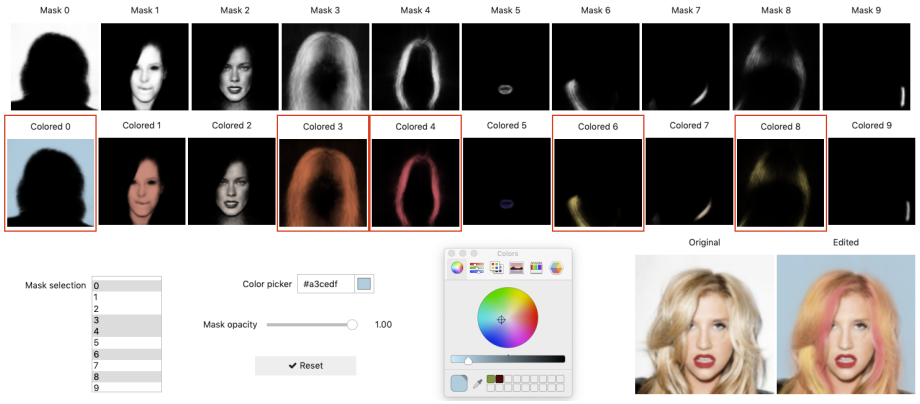


Figure 3.7: Our editing interface using automatically extracted masks to bootstrap the editing process.

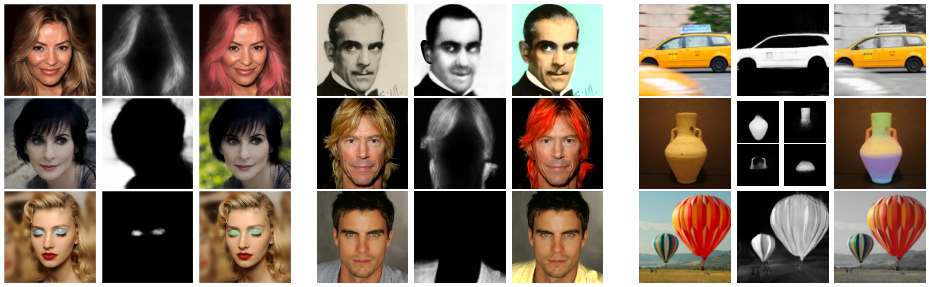


Figure 3.8: Some editings on CelebA and ImageNet, using little supervision (mask selection in one click and new style/color selection). Note that the CelebA editings are performed on 1024×1024 images. Left: original; center: mask; right: edit.

- Recursive vs one-shot: We generate the mask parameters recursively to allow the model to better take into account the interaction between the different masks. We show that a one-shot baseline, where all the mask parameters are predicted in a single pass leads to worse results. Moreover, as mentioned above and demonstrated in the experiments, our recursive procedure can be applied a larger number of times to model more complex images.

Applications. We now demonstrate how our image decomposition may serve different purposes such as image editing, retrieval and vectorization.

Image editing.
Image editing from raw pixels can be time consuming. Using our generated masks, it is possible to alter the original image by applying edits such as luminosity or color modifications on the region specified by a mask. Fig. 3.7 shows an interface we designed for

such editing showing the masks corresponding to the image. It avoids going through the tedious process of defining a blending mask manually. The learned masks capture the main components of the image, such as the background, face, hairs, lips. Fig. 3.8 demonstrate a variety of editing we performed and the associated masks. Our approach works well on the CelebA dataset, and allows to make simple image modifications on the more challenging ImageNet images.

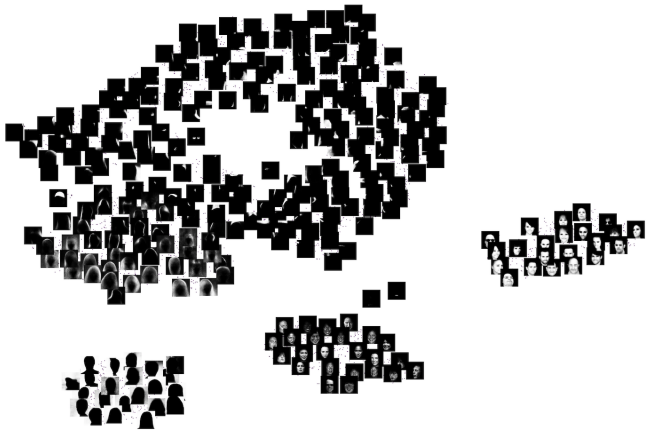


Figure 3.9: t-SNE visualization of masks obtained from 5000 reconstructions on CelebA.

Attribute-based image retrieval.
A t-SNE [62] visualization of the mask parameters obtained on CelebA is shown in Fig. 3.9. Different clusters of masks are clearly visible, for backgrounds, hairs, face shadows, etc. This experiment highlights the fact that our approach naturally extracts semantic components of face images. Thus, our approach may be used in an image content search: given a query image, a user can select a mask that displays a particular attribute of interest and search for images which decomposition includes similar masks. Suppose we would like to retrieve pictures of people wearing a hat as displayed in a query image, we can easily extract the mask that corresponds to the hat in our decomposition and its parameters. Nearest neighbor for different masks, using a simple ℓ_2 distance between mask parameters $\mathbf{p}$ are provided in Fig. 3.10. Note how different masks extracted from the same query image lead to very different retrieved images. Such a strategy could potentially be used for efficient image annotation or few-shot learning. We evaluated oneshot nearest neighbor classification for the "Wearing Hat" and "Eyeglasses" categories in CelebA using the hat and glasses examples shown in Fig. 3.10. We obtained respectively 40% and 50% average precision. Results were especially impressive for glasses, with 30% recall at 98% precision.

Vector image generation.
Producing vectorized images is often essential for design applications. We demonstrate in Fig. 3.11(a) the potential of our approach for producing a continuous vector

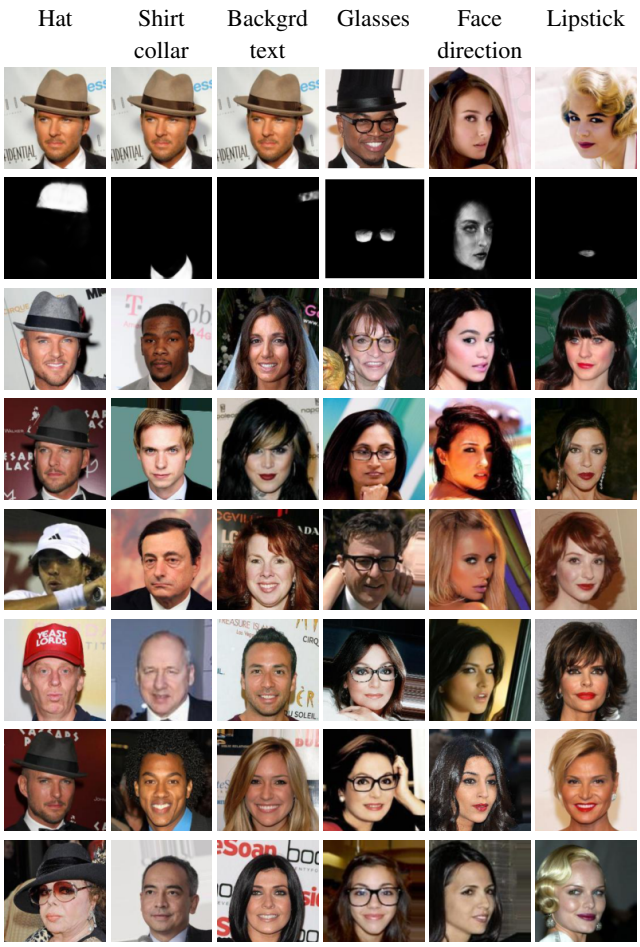
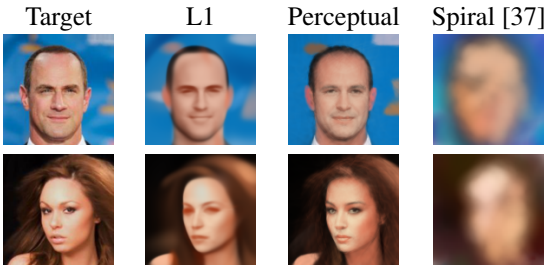


Figure 3.10: Given a target image and a mask of an area of interest extracted from it, a nearest neighbor search in the learned mask parameter space allows the retrieval of images sharing the desired attribute with the target.

image from a low resolution bitmap. Here, we train our network on the MNIST dataset (28×28), but generate the output at resolution 1024×1024. Compared to bilinear interpolation, the image we generate presents less artifacts.

We finally compare our model with SPIRAL [37] on a few images from CelebA dataset published in [37]. SPIRAL is the approach the most closely related to ours, using existing design tools and producing a vector image. As can be seen in Figure 3.11, we produce much higer quality images than this baseline.

(a) Vectorization: reconstructions of MNIST images.



(b) Comparison with SPIRAL [37] on CelebA.

Figure 3.11: Our model learns a vectorized mask representation that can be generated at any resolution without interpolation artifacts.

3.4 Discussion.

In this Chapter, we presented the idea to use neural network to learn families of continuous functions, and applied it to three very different problems: 3D shape correspondences, surface reconstruction and image reconstruction. We recently explored several natural extensions of these approaches. In recent papers, we show both that learning shape templates can improve correspondence results and surface reconstruction [92] and that it was possible to learn template-free correspondences.

Chapter 4

Discussion

4.1 Contributions

In this report, I have presented two series of works aiming to design deep features inline with two common intuitions about human perception.

First I discussed how to make deep features robust with respect to changes in the modality of a depiction, a task that is very intuitive to humans but proves challenging for a machine learning system. I introduced and discussed three different approaches: (i) explicitly learning invariance in feature space, which I motivated by an analysis of the factorization properties of deep features; (ii) learning with randomized synthetic images to encourage results which do not depend on specific factors of variations; (iii) learning dataset specific features without annotation by leveraging spatial co-occurrences. These three approaches can be used to tackle invariant scene analysis in a wide variety of problems, with different types of supervision or knowledge available. We recently adapted and compared all of these approaches on the problem of historical watermarks identification from photographs using a database of drawings [100].

Second, I presented a new way to generate 3D shapes and images by composing a restricted family of vector primitive elements, similar to our intuitions that objects are composed of parts. The key idea is to use multi-layer perceptrons to learn parametric families of continuous functions of the 2D and 3D space. This is in strong contrast with the dominant approaches, which typically generated pixels or voxels grids using convolutions. This proved a particularly efficient approach for single view reconstruction and shape matching, with a clear improvement over state of the art methods. I demonstrated that this type of approach allowed new type of applications, such as meshing or texturing of 3D shapes and editing of images. We recently extended this idea by demonstrating how primitive elements for 3D shapes could be learned automatically from the data instead of manually designed [92].

While the first line of work tends to focus on image analysis and the second line on shapes and image generation, I believe a very fruitful research direction would be to merge the two ideas.

4.2 Other works

During the last 5 years, I also worked in other related directions.

Cycle consistency as a supervisory signal. In section 2.3, I introduced a method to learn features from spatial consistency. Another type of supervisory signal that can be used in particular to learn correspondences is cycle consistency, or in other words the fact that correspondences are transitive. I was first part of a project that applied this idea to learn dense correspondences in images of different objects from the same category [103], also relying on ground truth correspondences given by rendering a 3D model from different viewpoints. More recently, we applied this idea to learn correspondences and transfer annotations between 3D shapes from different objects of the same category [41].

Vision for robotics. As mentioned in the introduction, the motivation for several of the works presented in this thesis is robotics. Beyond the domain randomization work presented in section 2.2, I worked on several projects aiming directly at robotics. In [63], we apply 3D model retrieval to compute object grasps efficiently. In [87], we estimate the world state with a technique extending to the one introduced in 2.2 to perform rearrangement planning. Finally, in [101] we introduce an instance specific pose estimation method, which leads to improved results and we argue is more useful for robotics than category pose estimation.

4.3 Perspectives

Analysis by generation. Pre-training powerful representations of images using an auto-encoder is a natural idea in deep learning [46]. However, it didn't fulfill its promises for image recognition, since auto-encoders typically compress image information in a way that is not useful in most analysis scenarios. Recently, self-supervised learning methods have revisited successfully this idea by learning to generate an unseen part of images [71, 45]. I believe it is possible to go further and learn useful representations from complete reconstructions by adapting the architecture and training objectives. The main benefit would be a new way to automatically discover important elements in image and 3D shapes. We recently started exploring these idea by performing discovery by generation in temporal image collections, using a new bilinear factorization module in [93], and learning 3D shape elements from generation in [92].

Reasoning about image and scenes. I believe developing approaches corresponding to human intuitions is key to designing systems that can reason about images and 3D scenes. While reasoning is a relatively vague concept, I plan to work with two concrete goals: (i) integrating very different types of cues, such as visual and physical cues for 3D scenes, or visual and semantic cues for document analysis; (ii) unsupervised visual learning, extracting meaningful groups of objects from a dataset, for example discovering all repeated letters in a document, or all cylinders in a 3D model. A fundamental challenge is the development of approaches to represent a probabilistic model

of the world able to consider several possible interpretations and their uncertainty. Indeed, while losses used in deep learning typically have probabilistic interpretations, the score predicted by a network are often bad estimate of the prediction confidence and predictions are often uni-modal.

Bibliography

[1] Brueghel family: Jan brueghel the elder." the brueghel family database. university of california, berkeley. http://www.janbrueghel.net/. Accessed: 2018-10-16.

[2] Wikiart. https://www.wikiart.org/. Accessed: 2018-10-16.

[3] Wikiart retriever. https://github.com/lucasdavid/wikiart/. Accessed: 2018-10-16.

[4] Edward Adelson. *Layered representations for image coding*. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, 1991.

[5] Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. 2014.

[6] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. *ACM transactions on graphics (TOG)*, 24(3):408–416, 2005.

[7] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. *IEEE International Conference on Computer Vision (ICCV) - Workshop on Dynamic Shape Capture and Analysis (4DMOD)*, 2011.

[8] Mathieu Aubry, Daniel Maturana, Alexei Efros, Bryan Russell, and Josef Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[9] Mathieu Aubry, Daniel Maturana, Alexei A. Efros, Bryan C. Russell, and Josef Sivic. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. 2014.

[10] Mathieu Aubry and Bryan C Russell. Understanding deep features with computer-generated imagery. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. 24(4):509–522, 2002.

[12] P. J. Besl and N. McKay. A method for registration of 3-D shapes. 14(2):239–256, 1992.

[13] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[14] I Binford. Visual perception by computer. In *IEEE Conference of Systems and Control*, 1971.

[15] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[16] J. Bruna and S. Mallat. Invariant scattering convolution networks. 35(8):1872–1886, 2013.

[17] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical report, 2015.

[18] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005.

[19] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. 2016.

[20] Chin Seng Chua and Ray Jarvis. Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision*, 25(1):63–85, 1997.

[21] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: Measuring error on simplified surfaces. In *Computer Graphics Forum*, volume 17, pages 167–174. Wiley Online Library, 1998.

[22] Thomas M Cover, Peter E Hart, et al. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[23] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

[24] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[25] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE, 2005.

[26] Johanna Delanoy, Mathieu Aubry, Phillip Isola, Alexei Efros, and Adrien Bousseau. 3d sketching using multi-view deep volumetric prediction. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2018.

[27] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[28] Alexey Dosovitskiy and Thomas Brox. Inverting convolutional networks with convolutional networks. *arXiv preprint arXiv:1506.02753*, 2015.

[29] M. Everingham, A. Zisserman, C.K.I. Williams, and L. Van Gool. The pascal visual object classes challenge 2006 (VOC 2006) results. Technical report, September 2006.

[30] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3D object reconstruction from a single image. 2017.

[31] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. 61(1), 2005.

[32] 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(9):1627–1645, 2010.

[33] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(9):1627–1645, 2010.

[34] Basura Fernando, Tatiana Tommasi, and Tinne Tuytelaars. Location recognition over large time lags. *Computer Vision and Image Understanding*, 139:21–28, 2015.

[35] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Trans. on Computer*, 22(1), 1973.

[36] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[37] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. *ICML*, 2018.

[38] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. IEEE, 2014.

[39] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[40] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Shape correspondences from learnt template-based parametrization. 2018.

[41] Thibault Groueix, Matthew Fisher, Vova Kim, Bryan Russell, and Mathieu Aubry. Unsupervised cycle-consistent deformation for shape matching. In *Symposium on Geometry Processing (SGP)*, 2019.

[42] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015.

[43] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *International Conference on 3D Vision (3DV)*. 2017.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[45] Olivier J Hénaff, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

[46] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[47] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[48] Phillip Isola and Ce Liu. Scene collaging: Analysis and synthesis of natural images with semantic layers. In *ICCV*, 2013.

[49] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[50] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[51] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):29, 2013.

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012.

[53] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. 2011.

[54] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[55] Yann LeCun, Fu-Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. 2004.

[56] Bastian Leibe and Bernt Schiele. Analyzing appearance and contour based methods for object categorization. volume 2, pages II–409. IEEE, 2003.

[57] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, 2015.

[58] Joseph J. Lim, Hamed Pirsiavash, and Antonio Torralba. Parsing IKEA objects: Fine pose estimation. In *ICCV*, 2013.

[59] O. Litany, T. Remez, E. Rodola, A. M. Bronstein, and M. M. Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. *ICCV*, 2017.

[60] Vianney Loing, Renaud Marlet, and Mathieu Aubry. Virtual training for a real application: Accurate object-robot relative localization without calibration. *International Journal of Computer Vision*, 2017.

[61] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[62] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[63] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.

[64] Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, 2016.

[65] Francisco Massa, Renaud Marlet, and Mathieu Aubry. Crafting a multi-task cnn for viewpoint estimation. In *British Machine Vision Conference*. British Machine Vision Association.

[66] Francisco Massa, Bryan C Russell, and Mathieu Aubry. Deep exemplar 2d-3d detection by adapting from real to rendered views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[67] Francisco Vitor Suzano Massa. *Relating images and 3D models with convolutional neural networks*. PhD thesis, Université Paris-Est, 2017.

[68] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[69] Marvin Minsky and Seymour Papert. Perceptron: an introduction to computational geometry. *The MIT Press, Cambridge, expanded edition*, 19(88):2, 1969.

[70] Joseph L Mundy. Object recognition in the geometric era: A retrospective. In *Toward category-level object recognition*, pages 3–28. Springer, 2006.

[71] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[72] Xingchao Peng, Kate Saenko, Baochen Sun, and Karim Ali. Learning deep object detectors from 3D models. In *ICCV*, 2015.

[73] B. Pepik, R. Benenson, T. Ritschel, and B. Schiele. What is holding back convnets for detection? In *arXiv:1508.02844*, 2015.

[74] Bojan Pepik, Michael Stark, Peter Gehler, and Bernt Schiele. Teaching 3D geometry to deformable part models. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3362–3369. IEEE, 2012.

[75] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[76] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. 2017.

[77] Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Feature mapping for learning fast and accurate 3d pose inference from synthetic images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4663–4672, 2018.

[78] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *arXiv preprint arXiv:1711.02512*, 2017.

[79] Lawrence G Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology (MIT), 1963.

[80] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[81] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[82] DE RUMELHART. Learning internal representations by error propagation. *Parallel Distributed Processing*, 1:chap–8, 1986.

[83] Raif M Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 225–233. Eurographics Association, 2007.

[84] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

[85] Othman Sbai, Camille Couprie, and Mathieu Aubry. Vector image generation by learning parametric layer decomposition. *arXiv preprint arXiv:1812.05484*, 2018.

[86] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[87] Zagoruyko Sergey, Yann Labbé, Igor Kalevatykh, Ivan Laptev, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Monte-carlo tree search for efficient visually guided rearrangement planning. *arXiv preprint arXiv:1904.10348*, 2019.

[88] Xi Shen, Alexei A Efros, and Aubry Mathieu. Discovering visual patterns in art collections with spatially-consistent feature learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[89] Hao Su, Charles Qi, Yangyan Li, and Leonidas Guibas. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views. In *ICCV*, 2015.

[90] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2686–2694, 2015.

[91] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature-based on heat diffusion". *Computer Graphics Forum (Proc. of SGP)*, 2009.

[92] Deprelle Théo, Thibault Groueix, Matthew Fisher, Vova Kim, Bryan Russell, and Mathieu Aubry. Learning elementary structures
for 3d shape generation and matching. In *Advances in Neural Information Processing Systems*, 2019.

[93] Dalens Théophile, Mathieu Aubry, and Josef Sivic. Bilinear image translation for temporal analysis of photo collections. *to appear*, 2019.

[94] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1510–1519. IEEE, 2015.

[95] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proc. CVPR*, volume 2, 2017.

[96] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[97] John Wang and Edward Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.

[98] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):72, 2017.

[99] Yuxin Wu and Kaiming He. Group normalization. *arXiv preprint arXiv:1803.08494*, 2018.

[100] Shen Xi, Pastrolin Ilaria, Bounou Oumayma, Gidaris Spyros, Smith Marc, Poncet Olivier, and Mathieu Aubry. Large scale historical watermarks recognition: dataset and a new consistency-based approach. *ArXiv*, 2019.

[101] Yang Xiao, Xuchong Qiu, Pierre-Alain Langlois, Mathieu Aubry, and Renaud Marlet. Pose from shape: Deep pose estimation for arbitrary 3D objects. In *British Machine Vision Conference (BMVC)*, 2019.

[102] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? 2014.

[103] Tinghui Zhou, Philipp Krahenbuhl, Mathieu Aubry, Qixing Huang, and Alexei A Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[104] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv:1703.10593*, 2017.

[105] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. *NIPS*, 2017.

[106] S. Zuffi and M. J. Black. The stitched puppet: A graphical model of 3d human shape and pose. *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.