

(DE)2CO: Deep Depth Colorization

Original

(DE)2CO: Deep Depth Colorization / Carlucci Fabio, Maria; Russo, Paolo; Caputo, Barbara. - In: IEEE ROBOTICS AND AUTOMATION LETTERS. - ISSN 2377-3766. - STAMPA. - 3:3(2018), pp. 2386-2393. [10.1109/LRA.2018.2812225]

Availability:

This version is available at: 11583/2760929 since: 2019-10-17T09:20:16Z

Publisher:

IEEE Robotics and Automation Society

Published

DOI:10.1109/LRA.2018.2812225

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

IEEE postprint/Author's Accepted Manuscript

©2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

(DE)²CO: Deep Depth Colorization

F. M. Carlucci*, P. Russo*, and B. Caputo¹

Abstract—The ability to classify objects is fundamental for robots. Besides knowledge about their visual appearance, captured by the RGB channel, robots heavily need also depth information to make sense of the world. While the use of deep networks on RGB robot images has benefited from the plethora of results obtained on databases like ImageNet, using convnets on depth images requires mapping them into three dimensional channels. This transfer learning procedure makes them processable by pre-trained deep architectures. Current mappings are based on heuristic assumptions over pre-processing steps and on what depth properties should be most preserved, resulting often in cumbersome data visualizations, and in sub-optimal performance in terms of generality and recognition results. Here we take an alternative route and we attempt instead to *learn* an optimal colorization mapping for any given pre-trained architecture, using as training data a reference RGB-D database. We propose a deep network architecture, exploiting the residual paradigm, that learns how to map depth data to three channel images. A qualitative analysis of the images obtained with this approach clearly indicates that learning the optimal mapping preserves the richness of depth information better than current hand-crafted approaches. Experiments on the Washington, JHUIT-50 and BigBIRD public benchmark databases, using CaffeNet, VGG-16, GoogleNet, and ResNet50 clearly showcase the power of our approach, with gains in performance of up to 16% compared to state of the art competitors on the depth channel only, leading to top performances when dealing with RGB-D data.

I. INTRODUCTION

Robots need to recognize what they see around them to be able to act and interact with it. Recognition must be carried out in the RGB domain, capturing mostly the visual appearance of things related to their reflectance properties, as well as in the depth domain, providing information about the shape and silhouette of objects and supporting both recognition and interaction with items. The current mainstream state of the art approaches for object recognition are based on Convolutional Neural Networks (CNNs, [1]), which use end-to-end architectures achieving feature learning and classification at the same time. Some notable advantages of these networks are their ability to reach much higher accuracies on basically any visual recognition problem, compared to what would be achievable with heuristic methods; their being domain-independent, and their conceptual simplicity. Despite these advantages, they also present some limitations, such as

This work was partially supported by the ERC grant 637076 - RoboExNovo (F.M.C., P. R., B.C.), and the CHIST-ERA project ALOOF (B.C, F. M. C., P. R.).

* Equal contribution

¹ All authors are at the Department of Computer, Control, and Management Engineering Antonio Ruberti at Sapienza Rome University, Italy and at the Italian Institute of Technology, VANDAL Laboratory, Milan, Italy { fabio.carlucci, paolo.russo, barbara.caputo } @iit.it

high computational cost, long training time and the demand for large datasets, among others.

This last issue has so far proved crucial in the attempts to leverage over the spectacular success of CNNs over RGB-based object categorization [2], [3] in the depth domain. Being CNNs data-hungry algorithms, the availability of very large scale annotated data collections is crucial for their success, and architectures trained over ImageNet [4] are the cornerstone of the vast majority of CNN-based recognition methods. Besides the notable exception of [5], the mainstream approach for using CNNs on depth-based object classification has been through transfer learning, in the form of a mapping able to make the depth input channel compatible with the data distribution expected by RGB architectures. Following recent efforts in transfer learning [6], [7], [8] that made it possible to use depth data with CNN pre-trained on a database of a different modality, several authors proposed hand-crafted mappings to colorize depth data, obtaining impressive improvements in classification over the Washington [9] database, that has become the golden reference benchmark in this field [10], [11].

We argue that this strategy is sub-optimal. By hand-crafting the mapping for the depth data colorization, one has to make strong assumptions on what information, and up to which extent, should be preserved in the transfer learning towards the RGB modality. While some choices might be valid for some classes of problems and settings, it is questionable whether the family of algorithms based on this approach can provide results combining high recognition accuracies with robustness across different settings and databases. Inspired by recent works on colorization of gray-scale photographs [12], [13], [14], we tackle the problem by exploiting the power of end-to-end convolutional networks, proposing a deep depth colorization architecture able to learn the optimal transfer learning from depth to RGB for any given pre-trained convnet. Our deep colorization network takes advantage of the residual approach [15], learning how to map between the two modalities by leveraging over a reference database (Figure 1, top), for any given architecture. After this training stage, the colorization network can be added on top of its reference pre-trained architecture, for any object classification task (Figure 1, bottom). We call our network (DE)²CO: DEep DEpth COLORization.

We assess the performance of (DE)²CO in several ways. A first qualitative analysis, comparing the colored depth images obtained by (DE)²CO and by other state of the art hand-crafted approaches, gives intuitive insights on the advantages brought by learning the mapping as opposed to choosing it, over several databases. We further deepen this

analysis with an experimental evaluation of our and other existing transfer learning methods on the depth channel only, using four different deep architectures and three different public databases, with and without fine-tuning. Finally, we tackle the RGB-D object recognition problem, combining (DE)²CO with off-the shelf state of the art RGB deep networks, benchmarking it against the current state of the art in the field. For all these experiments, results clearly support the value of our algorithm. All the (DE)²CO modules, for all architectures employed in this paper, are available at <https://github.com/fmcarlucci/de2co>.

II. RELATED WORK

Since 2012’s AlexNet [3] spectacular success, CNNs have become the dominant learning paradigm in visual recognition. Several architectures have been proposed in recent years, each bringing new flavors to the community. Simonyan and Zisserman [16] investigated the effect of increasing the network depth. GoogLeNet [2] also increased the depth and width of the network while restraining the computational budget, with a dramatic reduction in the number of parameters. He et al. [15] proposed a residual learning approach using a batch normalization layer and special skip connections for training deeper architectures, showing an impressive success in ILSVRC2015. All these architectures will be used in this work, to assess its generality.

Lately, several authors attempted to take advantage of pre-trained CNNs to perform RGB-D detection and recognition. Colorization of depth images can be seen as a transfer learning process across modalities, and several works explored this avenue within the deep learning framework. In the context of RGB-D object detection, a recent stream of works explicitly addressed cross modal transfer learning through sharing of weights across architectures [17], [18] and [19]. This last work is conceptually close to our approach, as it proposes to learn how to transfer RGB extracted information to the Depth domain through distillation [20]. While [19] has proved very successful in the object detection realm, it presents some constraints that might potentially be problematic in object recognition, from the requirement of paired RGB-D images, to detailed data preprocessing and preparation for training. As opposed to this, our algorithm does not require explicit pairing of images in the two modalities, can be applied successfully on raw pixel data and does not require other data preparation for training.

Within the RGB-D classification literature, [21] converts the depth map to surface normals and then re-interprets it as RGB values, while Aekerberg et al. [22] builds on this approach and suggests an effective preprocessing pipeline to increase performance. A new method has also been proposed by Gupta, Saurabh, et al. [23] : HHA is a mapping where one channel encodes the horizontal disparity, one the height above ground and the third the pixelwise angle between the surface normal and the gravity vector. Schwarz et al [24] proposed a colorization pipeline where colors are assigned to the image pixels according to the distance of the vertexes of a rendered mesh to the center of the object. Besides the naive

grayscale method, the rest of the mentioned colorization schemes are computationally expensive. Eitel et al [11] used a simple color mapping technique known as *ColorJet*, showing this simple method to be competitive with more sophisticated approaches.

All these works, and many others [25], [5], make use of an ad-hoc mapping for converting depth images into three channels. This conversion is vital as the dataset has to be compatible with the pre-trained CNN. Depth data is encoded as a 2D array where each element represents an approximate distance between the sensor and the object. Depth information is often depicted and stored as a single monochrome image. Compared to regular RGB cameras, the depth resolution is relatively low, especially when the frame is cropped to focus on a particular object. While addressing this issue, we avoid heuristic choices in our approach and we rely instead on an end-to-end, residual based deep architecture to learn the optimal mapping for the cross modal knowledge transfer.

Most of works in object recognition, against whom we compare our method, are evaluated on one single database, with Washington being the standard choice in the robot vision literature. This raises concerns about the generality of these methods, especially considering their hand-crafted nature. We circumvent this issue by evaluating (DE)²CO on three different databases.

Our work is also related to the colorization of grayscale images using deep nets. Cheng et al [14] proposed a colorization pipeline based on three different hand-designed feature extractors to determine the features from different levels of an input image. Larsson et al [13] used an architecture consisting of two parts. The first part is a fully convolutional version of VGG-16 used as feature extractor, and the second part is a fully-connected layer with 1024 channels predicting the distributions of hue and the chroma for each pixel given its feature descriptors from the previous level. Iizuka et al [12] proposed an end-to-end network able to learn global and local features, exploiting the classification labels for better image colorization. Their architecture consists of several networks followed by fusion layer for the colorization task. Sun et al. [26] propose to use large scale CAD rendered data to leverage depth information without using low level features or colorization. In Asif et al. [27], hierarchical cascaded forests were used for computing grasp poses and perform object classification, exploiting several different features like orientation angle maps, surface normals and depth information colored with *Jet* method. Our work differs from this last research thread in the specific architecture proposed, and in its main goal, as here we are interested in learning optimal mapping for categorization rather than for colorization of grayscale images.

III. COLORIZATION OF DEPTH IMAGES

Although depth and RGB are modalities with significant differences, they also share enough similarities (edges, gradients, shapes) to make it plausible that convolutional filters learned from RGB data could be re-used effectively for

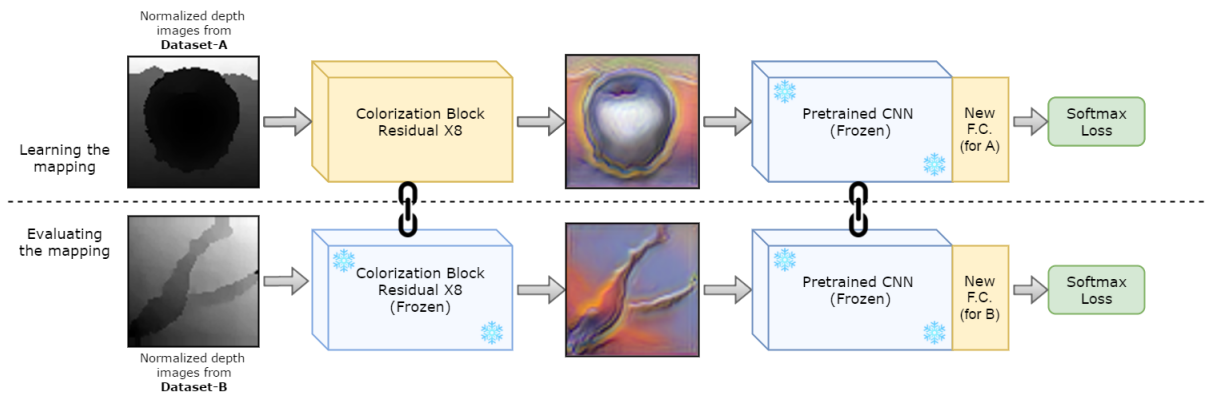


Fig. 1: The $(DE)^2CO$ pipeline consists of two phases. First, we learn the mapping, from depth to color, maximizing the discrimination capabilities of a network pre trained on ImageNet. In this step the network is frozen and we are only learning the mapping and the final layer. We then evaluate the colorization on a *different* depth dataset: here we also freeze the colorization network and only train a new final layer for the testbed dataset.

representing colorized depth images. The approach currently adopted in the literature consists of designing ad-hoc colorization algorithms, as revised in the previous section. We refer to these kind of approaches as *hand-crafted depth colorization*. Specifically, we choose ColorJet [11], SurfaceNormals [21] and *SurfaceNormals++* [22] as baselines against which we will assess our data driven approach because of their popularity and effectiveness.

In the rest of the section we first briefly summarize ColorJet (section III-A), SurfaceNormals and SurfaceNormals++ (section III-B). We then describe our deep approach to depth colorization (section III-C). To the best of our knowledge, $(DE)^2CO$ is the first deep colorization architecture applied successfully to depth images.

A. Hand-Crafted Depth Colorization: ColorJet

ColorJet works by assigning different colors to different depth values. The original depth map is firstly normalized between 0-255 values. Then the colorization works by mapping the lowest value to the blue channel and the highest value to the red channel. The value in the middle is mapped to green and the intermediate values are arranged accordingly [11]. The resulting image exploits the full RGB spectrum, with the intent of leveraging at best the filters learned by deep networks trained on very large scale RGB datasets like ImageNet. Although simple, the approach gave very strong results when tested on the Washington database, and when deployed on a robot platform. Still, ColorJet was not designed to create realistic looking RGB images for the objects depicted in the original depth data (Figure 3, bottom row). This raises the question whether this mapping, although more effective than other methods presented in the literature, might be sub-optimal. In section III-C we will show that by fully embracing the end-to-end philosophy at the core of deep learning, it is indeed possible to achieve significantly higher recognition performances while at the same time producing more realistic colorized images.

B. Hand-Crafted Depth Colorization: SurfaceNormals(++)

The SurfaceNormals mapping has been often used to convert depth images to RGB [21], [28], [11]. The process is straightforward: for each pixel in the original image the corresponding surface normal is computed as a normalized 3D vector, which is then treated as an RGB color. Due to the inherent noisiness of the depth channel, such a direct conversion results in noisy images in the color space. To address this issue, the mapping we call *SurfaceNormals++* was introduced by Aakerberg [22]: first, a recursive median filter is used to reconstruct missing depth values, subsequently a bilateral filter smooths the image to reduce noise, while preserving edges. Next, surface normals are computed for each pixel in the depth image. Finally the image is sharpened using the unsharp mask filter, to increase contrast around edges and other high-frequency components.

C. Deep Depth Colorization: $(DE)^2CO$

$(DE)^2CO$ consists of feeding the depth maps, normalized into grayscale images, to a *colorization network* linked to a standard CNN architecture, pre-trained on ImageNet.

Given the success of deep colorization networks from grayscale images, we first tested existing architectures in this context [29]. Extensive experiments showed that while the visual appearance of the colorized images was very good, the recognition performances obtained when combining such network with pre-trained RGB architectures was not competitive. Inspired by the generator network in [30], we propose here a *residual* convolutional architecture (Figure 2). By design [15], this architecture is robust and allows for deeper training. This is helpful here, as $(DE)^2CO$ requires stacking together two networks, which for not very deep architectures might lead to vanishing gradient issues. Furthermore, residual blocks works at pixel level [30] helping to preserve locality.

Our architecture works as follows: the $1 \times 228 \times 228$ input depth map, reduced to $64 \times 57 \times 57$ size by a conv&pool layer, passes through a sequence of 8 residual blocks, composed by 2 small convolutions with batch normalization layers and

leakyRelu as non linearities. The last residual block output is convolved by a three features convolution to form the 3 channels image output. Its resolution is brought back to 228x228 by a *deconvolution* (upsampling) layer.

Our whole system for object recognition in the depth domain using deep networks pre-trained over RGB images can be summarized as follows: the entire network, composed by (DE)²CO and the classification network of choice, is trained on an annotated reference depth image dataset. The weights of the chosen classification network are kept frozen in their pre-trained state, as the only layer that needs to be retrained is the last fully connected layer connected to the softmax layer. Meanwhile, the weights of (DE)²CO are updated until convergence.

After this step, the depth colorization network has learned the mapping that maximizes the classification accuracy on the reference training dataset. It can now be used to colorize *any* depth image, from any data collection. Figure 3, top rows, shows exemplar images colored with (DE)²CO trained over different reference databases, in combination with two different architectures (CaffeNet, an implementation variant of AlexNet, and VGG-16 [16]). We see that, compared to the images obtained with ColorJet and SurfaceNormal++, our colorization technique emphasizes the objects contours and their salient features while flattening the object background, while the other methods introduce either high frequency noise (SurfaceNormals++) or emphasize background gradient instead of focusing mainly on the objects (ColorJet). In the next section we will show how this qualitative advantage translates also into a numerical advantage, i.e. how learning (DE)²CO on one dataset and performing depth-based object recognition on another leads to a very significant increase in performance on several settings, compared to hand-crafted color mappings.

IV. EXPERIMENTS

We evaluated our colorization scheme on three main settings: an ablation study of how different depth mappings perform when the network weights are kept frozen (section IV-B), a comparison of depth performance with network fine-tuning (section IV-C) and finally an assessment of (DE)²CO when used in RGB-D object recognition tasks (section IV-D). Before reporting on our findings, we illustrate the databases and baselines we used (section IV-A).

A. Experimental Setup

Databases We considered three datasets: the Washington RGB-D [9], the JHUIT-50 [31] and the BigBIRD [32] object datasets, which are the main public datasets for RGB-D object recognition. The first consists of 41,877 RGB-D images organized into 300 instances divided in 51 classes. We performed experiments on the object categorization setting, where we followed the evaluation protocol defined in [9]. The JHUIT-50 is a challenging recent dataset that focuses on the problem of fine-grained classification. It contains 50 object instances, often very similar with each other (e.g. 9 different kinds of screwdrivers). As such, it presents different

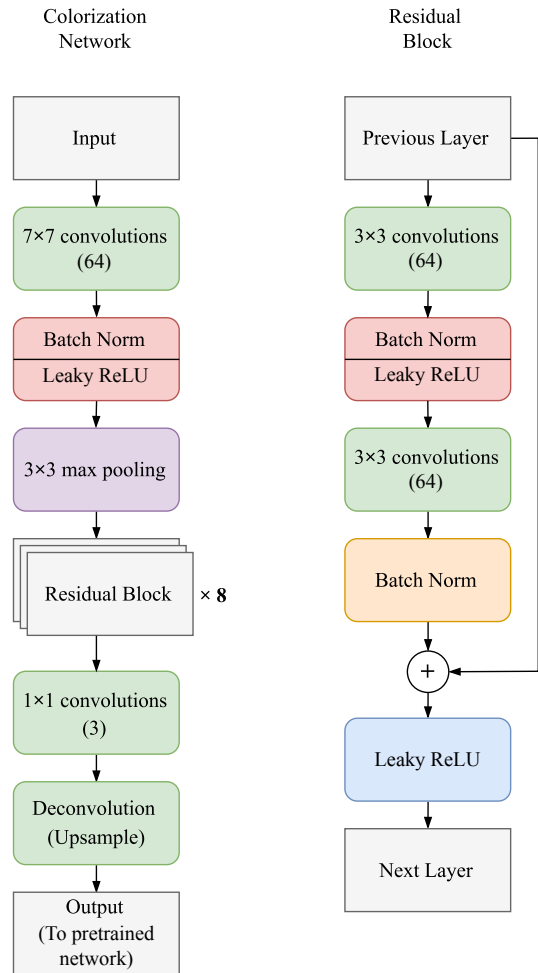


Fig. 2: Overview of the (DE)²CO colorization network. On the left, we show the overall architecture; on the right, we show details of the residual block.

recognition challenges compared to the Washington database. Here we followed the evaluation procedure defined in [31]. BigBIRD is the biggest of the datasets we considered: it contains 121 object instances and 75.000 images. Unfortunately, it is an extremely unforgiving dataset for evaluating depth features: many objects are extremely similar, and many are boxes, which are indistinguishable without texture information. To partially mitigate this, we grouped together all classes annotated with the same first word: for example *nutrigrain apple cinnamon* and *nutrigrain blueberry* were grouped into *nutrigrain*. With this procedure, we reduced the number of classes to 61 (while keeping all of the samples). As items are quite small, we used the object masks provided by [32] to crop around the object. Evaluation-wise, we followed the protocol defined in [31].

Hand-crafted Mappings According to previous works [11], [22], the two most effective mappings are *ColorJet* [11] and *SurfaceNormals* [21], [22]. For ColorJet we normalized the data between 0 and 255 and then applied the mapping using the OpenCV libraries¹. For the SurfaceNormals mapping we considered two versions: the straightforward conversion of

¹“COLORMAPJET” from <http://opencv.org/>

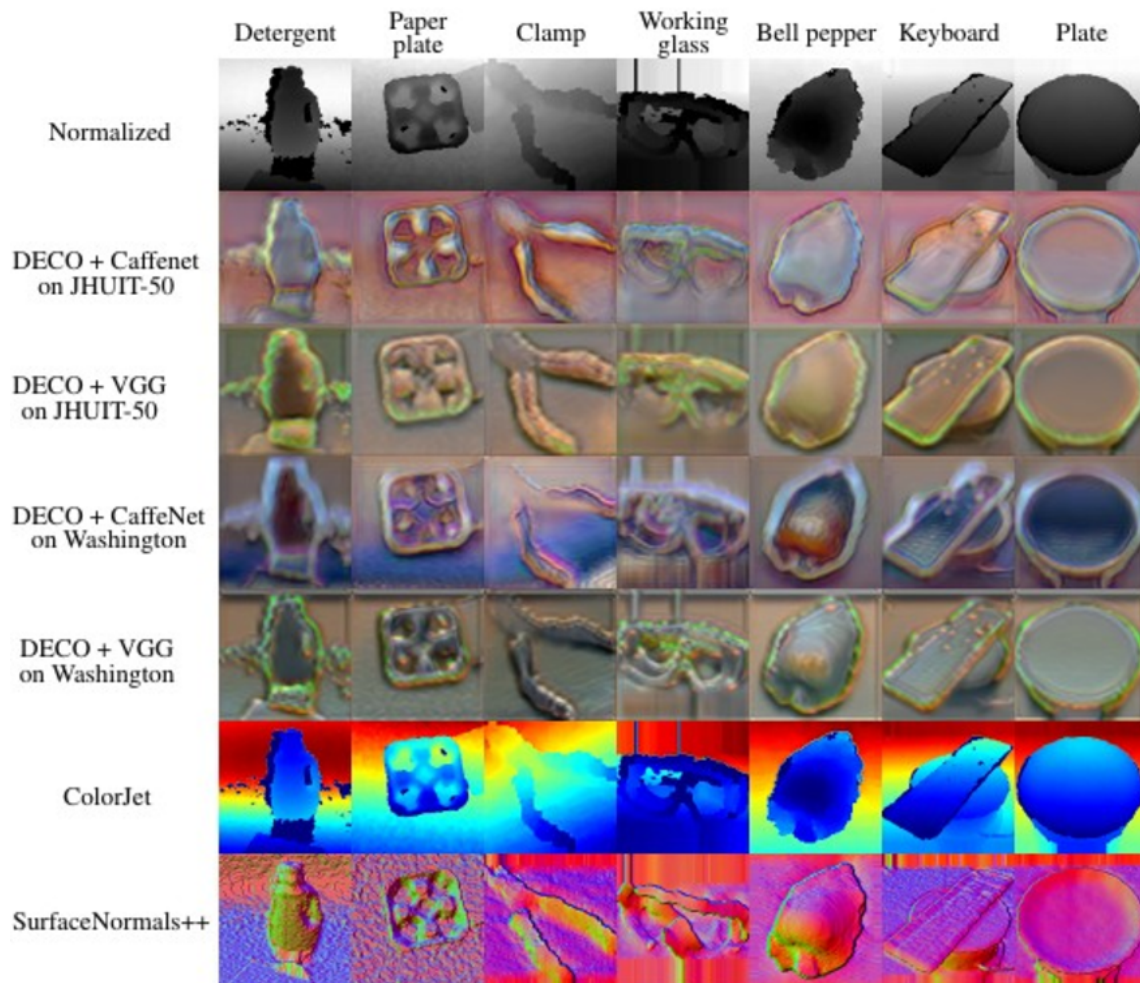


Fig. 3: $(DE)^2CO$ colorizations applied on different objects, taken from [31], [9], [32]. Top row shows the depth maps mapped to grayscale. From the second to the fourth row, we show the corresponding $(DE)^2CO$ colorizations learned on different settings. Fifth row shows *ColorJet* views [11], while the last row shows the surface normals mapping [22] *SurfaceNormals++*. These images showcase $(DE)^2CO$'s ability to emphasize the object's shape and to capture its salient features.

the depthmap to surface normals [21] and the enhanced version *SurfaceNormals++* [22] which uses extensive pre-processing and generally performs better².

B. Ablation Study

In this setting we compared our $(DE)^2CO$ method against hand crafted mappings, using pre-trained networks as feature extractors and only retraining the last classification layer. We did this on the three datasets described above, over four architectures: CaffeNet (a slight variant of the AlexNet [33]), VGG16 [16] and GoogleNet [2] were chosen because of their popularity within the robot vision community. We also tested the recent ResNet50 [15], which although not currently very used in the robotics domain, has some promising properties. In all cases we considered models pretrained on ImageNet [4], which we retrieved from Caffe's *Model Zoo*³.

Training $(DE)^2CO$ means minimizing the multinomial logistic loss of a network trained on RGB images. This means that our network is attached between the depth images

and the pre-trained network, of which we freeze the weights of all but the last layer, which are relearned from scratch (see Figure 1). We trained each network-dataset combination for 50 epochs using the Nesterov solver [34] and 0.007 starting learning rate (which is stepped down after 45%). During this phase, we used the whole *source* datasets, leaving aside only 10% of the samples for validation purposes.

When the dataset on which we train the colorizer is different from the test one, we simply retrain the new final layer (freezing all the rest) for the new classes.

Effectively we are using the pre-trained networks as feature extractors, as done in [24], [11], [25] and many others; for a performance analysis in the case of network finetuning we refer to paragraph IV-C. In this setting we used the Nesterov (for Washington and JHUIT-50) and ADAM (for BigBIRD) solvers. As we were only training the last fully connected layer, we learned a small handful of parameters with a very low risk of overfitting.

Table II reports the results from the ablation while Figure 4 focuses on the class recall for a specific experiment. For every architecture, we report the results obtained using Col-

²The authors graciously gave us their code for our experiments.

³<https://github.com/BVLC/caffe/wiki/Model-Zoo>

Network	Time (ms)	Network	Time (s)
CaffeNet	695	CaffeNet	1.87
VGG	1335	(DE) ² CO + CaffeNet	1.23
GoogleNet	1610	VGG	2.91
ResNet-50	1078	(DE) ² CO + VGG	2.16
(DE) ² CO colorizer	400		

TABLE I: Left: forward-backward time for 50 iterations, as by *caffe time*. Right: feature extraction times for 100 images; note that using (DE)²CO actually speeds up the procedure. We explain this by noting that (DE)²CO uses single channel images and thus needs to transfer only $\frac{1}{3}$ of the data from memory to the GPU - clearly the bottleneck here.

orJet, SurfaceNormals (plain and enhanced) and (DE)²CO learned on a reference database between Washington or JHUIT-50, and (DE)²CO learned on the combination of Washington and JHUIT-50. For the CaffeNet and VGG networks we also present results on simple grayscale images. We attempted also to learn (DE)²CO from BigBIRD alone, and in combination with one (or both) of the other two databases. Results on BigBIRD only were disappointing, and results with/without adding it to the other two databases did not change the overall performance. We interpret this result as caused by the relatively small variability of objects in BigBIRD with respect to depth, and for sake of readability we decided to omit them in this work.

We see that, for all architectures and for all reference databases, (DE)²CO achieves higher results. The difference goes from +1.7%, obtained with CaffeNet on the Washington database, to the best of +16.8% for VGG16 on JHUIT-50. JHUIT-50 is the testbed database where, regardless of the chosen architecture, (DE)²CO achieves the strongest gains in performance compared to hand crafted mappings. Washington is, for all architectures, the database where hand crafted mappings perform best, with the combination Washington to CaffeNet being the most favorable to the shallow mapping. On average it appears the CaffeNet is the architecture that performs best on this datasets; still, it should be noted that we are using here all architectures as feature extractors rather than as classifiers. On this type of tasks, both ResNet and GoogLeNet-like networks are known to perform worse than CaffeNet [35], hence our results are consistent with what reported in the literature. In Table III we report a second ablation study performed on the width and depth of (DE)²CO architecture. Starting from the standard (DE)²CO made of 8 residual blocks with 64 filters for each convolutional layer (which we found to be the best all-around architecture), we perform additional experiments by doubling and halving the number of residual blocks and the number of filters in each convolutional layer. As it can be seen, the (DE)²CO architecture is quite robust but can be potentially finetuned to each target dataset to further increase performance. In table I we report runtimes for the considered networks. As the results show, while (DE)²CO requires some extra computation time, in real life this is actually offset by the fact that only $\frac{1}{3}$ of the data is being moved to the GPU.

C. Finetuning

In our finetuning experiments we focused on the best performing network from the ablation, the CaffeNet (which is also used by current competitors [11], [22]), to see up to which degree the network could exploit a given mapping. The protocol was quite simple: all layers were free to move equally, the starting learning rate was 0.001 (with step down after 45%) and the solver was *SGD*. Training went on for 90 epochs for the Washington and JHUIT-50 datasets and 30 eps. for BigBIRD (a longer training was detrimental for all settings). To ensure a fair comparison with the static mapping methods, the (DE)²CO part of the network was kept frozen during finetuning.

Results are reported in Table IV. We see that here the gap between hand-crafted and learned colorization methods is reduced (very likely the network is compensating existing weaknesses). *SurfaceNormals++* performs pretty well on Washington, but less so on the other two datasets (it’s actually the worse on BigBIRD). Surprisingly, the simple grayscale conversion is the one that performs best on BigBIRD, but lags clearly behind on all other settings. (DE)²CO on the other hand, performs comparably to the best mapping on every single setting **and** has a 5.9% lead on JHUIT-50; we argue that it is always possible to find a shallow mapping that performs very well on a specific dataset, but there are no guarantees it can generalize.

D. RGB-D

While this paper focuses on how to best perform recognition in the depth modality using convnets, we wanted to provide a reference value for RGB-D object classification using (DE)²CO on the depth channel. To classify RGB images we follow [22] and use a pretrained VGG16 which we finetuned on the target dataset (using the previously defined protocol). RGB-D classification is then performed, without further learning, by computing the weighted average (weight factor α was cross-validated) of the *fc8* layers from the RGB and Depth networks and simply selecting the most likely class (the one with the highest activations). This cue integration scheme can be seen as one of the simplest, off-the-shelf algorithm for doing classifications using two different modalities [36]. We excluded BigBIRD from this setting, due to lack of competing works to compare with.

Results are reported in Tables V-VI. We see that (DE)²CO produces results on par or slightly superior to the current state of the art, even while using an extremely simple feature fusion method. This is remarkable, as competitors like [22], [11] use instead sophisticated, deep learning based cue integration methods. Hence, our ability to compete in this setting is all due to the (DE)²CO colorization mapping, clearly more powerful than the other baselines. It is worth stressing that, in spite of the different cue integration and depth mapping approaches compared in Tables V-VI, convnet results on RGB are already very high, hence in this setting the advantage brought by a superior performance on the depth channel tends to be covered. Still, on Washington

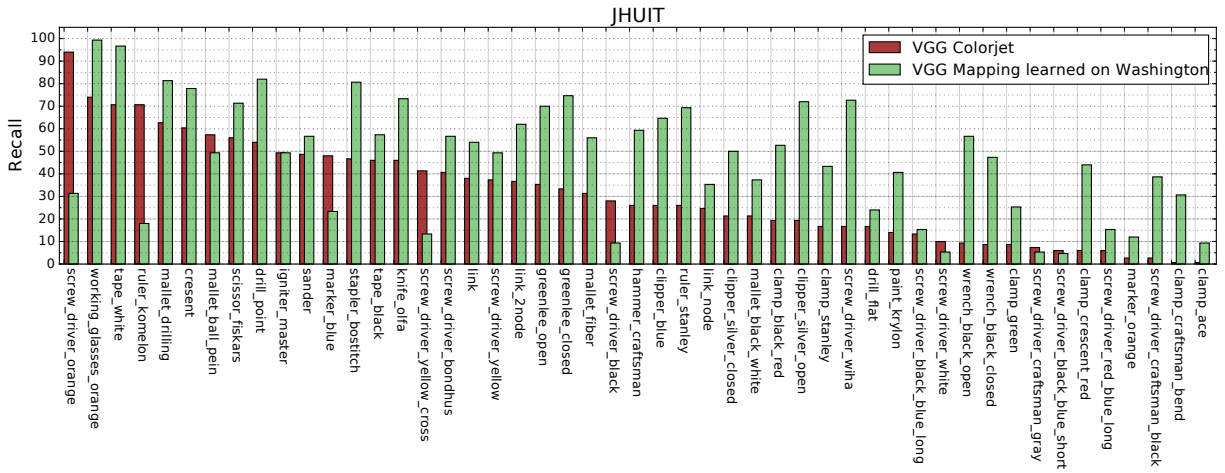


Fig. 4: Per class recall on JHUIT-50, using VGG, with (DE)²CO learned from Washington. Recalls per class are sorted in decreasing order, according to ColorJet performance. In this setting, (DE)²CO, while generally performing better, seems to focus on different perceptual properties and is thus, compared with the baseline, better at some classes rather than others.

Method:	Washington[9]	JHUIT-50[31]	BigBIRD Reduced[32]
VGG16 on Grayscale	74.9	33.7	22
VGG16 on ColorJet	75.2	35.3	19.9
VGG16 on SurfaceNormals	75.3	30.8	16.8
VGG16 on SurfaceNormals++	77.3	35.8	11.5
VGG16 (DE) ² CO learned on Washington	79.6	52.7	22.8
VGG16 (DE) ² CO learned on JHUIT-50	78.1	51.2	23.7
CaffeNet on Grayscale	76.6	44.6	22.9
CaffeNet on ColorJet	78.8	45.0	22.7
CaffeNet on SurfaceNormals	79.3	38.3	18.9
CaffeNet on SurfaceNormals++	81.4	44.8	14.0
CaffeNet (DE) ² CO learned on Washington	83.1	53.1	28.6
CaffeNet (DE) ² CO learned on JHUIT-50	79.1	57.5	25.2
GoogleNet on ColorJet	73.5	40.0	21.8
GoogleNet on SurfaceNormals	72.9	36.5	18.4
GoogleNet on SurfaceNormals++	76.7	41.5	13.9
GoogleNet (DE) ² CO learned on Washington	–	51.9	25.2
GoogleNet (DE) ² CO learned on JHUIT-50	76.6	–	24.4
ResNet50 on ColorJet	75.1	38.9	18.7
ResNet50 on SurfaceNormals	77.4	33.2	16.5
ResNet50 on SurfaceNormals++	79.6	45.4	13.8
ResNet50 (DE) ² CO learned on Washington	–	45.5	23.9
ResNet50 (DE) ² CO learned on JHUIT-50	76.4	–	24.7

TABLE II: Object classification experiments in the depth domain, comparing (DE)²CO and hand crafted mappings, using 5 pre-trained networks as feature extractors. Best results for each network-dataset combination are in **bold**, overall best in **red bold**. Extensive experiments were performed on VGG and Caffenet, while GoogleNet and ResNet act as reference.

filters/blocks	4 blocks	8 blocks	16 blocks
32 filters	56.5	52.8	57.1
64 filters	56.8	53.1	53.6
128 filters	53.1	53.9	53.3

TABLE III: (DE)²CO ablation study, learned on Washington, tested on JHUIT-50. Grid search optimization over width and depth of generator architecture shows improved results.

we achieve the second best result, and on JHUIT-50 we get the new state of the art.

V. CONCLUSIONS

This paper presented a network for learning deep colorization mappings. Our architecture follows the residual philosophy, learning how to map depth data to RGB images

for a given pre-trained convolutional neural network. By using our (DE)²CO algorithm, as opposed to the hand-crafted colorization mappings commonly used in the literature, we obtained a significant jump in performance over three different benchmark databases, using four different popular deep networks pre trained over ImageNet. The visualization of the obtained colorized images further confirms how our algorithm is able to capture the rich informative content and the different facets of depth data. All the deep depth mappings presented in this paper are available at <https://github.com/fmcarlucci/de2co>. Future work will further investigate the effectiveness and generality of our approach, testing it on other RGB-D classification and detection problems, with various fine-tuning strategies and on several

Method:	Washington[9]	JHUIT-50[31]	BigBIRD Reduced[32]
CaffeNet on Grayscale	82.7 ± 2.1	53.7	29.6
CaffeNet on ColorJet	83.8 ± 2.7	54.1	25.4
CaffeNet on SurfaceNormals++	84.5 ± 2.9	55.9	17.0
CaffeNet (DE) ² CO learned on Washington	84.0 ± 2.0	60.0	—
CaffeNet (DE) ² CO learned on JHUIT-50	82.3 ± 2.3	62.0	—
CaffeNet (DE) ² CO learned on Washington + JHUIT-50	84.0 ± 2.3	61.8	28.0

TABLE IV: CaffeNet finetuning using different colorization techniques.

Method:	RGB	Depth	RGB-D
FusionNet[11]	84.1 ± 2.7	83.8 ± 2.7	91.3 ± 1.4
CNN + Fisher [37]	90.8 ± 1.6	81.8 ± 2.4	93.8 ± 0.9
DepthNet [5]	88.4 ± 1.8	83.8 ± 2.0	92.2 ± 1.3
CIMDL [28]	87.3 ± 1.6	84.2 ± 1.7	92.4 ± 1.8
FusionNet enhanced[22]	89.5 ± 1.9	84.5 ± 2.9	93.5 ± 1.1
(DE) ² CO	89.5 ± 1.6	84.0 ± 2.3	93.6 ± 0.9

TABLE V: Selected results on Washington RGB-D

Method:	RGB	Depth	RGB-D
DepthNet [5]	88.0	55.0	90.3
Beyond Pooling [31]	—	—	91.2
FusionNet enhanced[22]	94.7	56.0	95.3
(DE) ² CO	94.7	61.8	95.7

TABLE VI: Selected results on JHUIT-50

deep networks, pre-trained over different RGB databases, and in combination with RGB convnet with more advanced multimodal fusion approaches.

REFERENCES

- [1] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *NIPS*, 1990.
- [2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.
- [3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.
- [4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
- [5] F. M. Carlucci, P. Russo, and B. Caputo, "A deep representation for depth images from synthetic data," in *ICRA*, 2017.
- [6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *ICML*, 2014.
- [8] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NIPS*, 2014.
- [9] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1817–1824.
- [10] M. Schwarz, H. Schulz, and S. Behnke, "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features," in *ICRA*, 2015.
- [11] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 681–687.
- [12] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification," *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, vol. 35, no. 4, 2016.
- [13] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization," in *ECCV*, 2016.
- [14] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *ICCV*, 2015.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
- [16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2015.
- [17] J. Hoffman, S. Gupta, J. Leong, S. Guadarrama, and T. Darrell, "Cross-modal adaptation for rgb-d detection," in *International Conference in Robotics and Automation (ICRA)*, 2016.
- [18] J. Hoffman, S. Gupta, and T. Darrell, "Learning with side information through modality hallucination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 826–834.
- [19] S. Gupta, J. Hoffman, and J. Malik, "Cross modal distillation for supervision transfer," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [20] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
- [21] L. Bo, X. Ren, and D. Fox, "Unsupervised feature learning for RGB-D based object recognition," in *13th International Symposium on Experimental Robotics, ISER*, 2012.
- [22] A. Aakerberg and K. Nasrollahi, "Depth value pre-processing for accurate transfer learning based rgb-d object recognition," in *IJCAI*, 2017.
- [23] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *ECCV*, 2014.
- [24] M. Schwarz, H. Schulz, and S. Behnke, "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features," in *ICRA*, 2015.
- [25] H. F. Zaki, F. Shafait, and A. Mian, "Convolutional hypercube pyramid for accurate rgb-d object category and instance recognition," in *ICRA*, 2016.
- [26] L. Sun, C. Zhao, and R. Stolkin, "Weakly-supervised dcnn for rgb-d object recognition in real-world applications which lack large-scale annotated training data," *arXiv preprint arXiv:1703.06370*, 2017.
- [27] U. Asif, M. Bennamoun, and F. A. Sohel, "Rgb-d object recognition and grasp detection using hierarchical cascaded forests," *IEEE Transactions on Robotics*, 2017.
- [28] Z. Wang, R. Lin, J. Lu, J. Feng, et al., "Correlated and individual multi-modal deep learning for rgb-d object recognition," *arXiv preprint arXiv:1604.01655*, 2016.
- [29] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," *ECCV*, 2016.
- [30] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," *arXiv preprint arXiv:1612.05424*, 2016.
- [31] C. Li, A. Reiter, and G. D. Hager, "Beyond spatial pooling: Fine-grained representation learning in multiple domains," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4913–4922.
- [32] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *ICRA*, 2014.
- [33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012.
- [34] Y. Nesterov, "A method of solving a convex programming problem with convergence rate o(1/k2)," in *Soviet Mathematics Doklady*, vol. 27, no. 2, 1983, pp. 372–376.
- [35] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability for a generic convnet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1790–1802, 2016.
- [36] T. Tommasi, F. Orabona, and B. Caputo, "Discriminative cue integration for medical image annotation," *Pattern Recognition Letters*, vol. 29, no. 15, pp. 1996–2002, 2008.
- [37] W. Li, Z. Cao, Y. Xiao, and Z. Fang, "Hybrid rgb-d object recognition using convolutional neural network and fisher vector," in *Chinese Automation Congress (CAC), 2015*. IEEE, 2015, pp. 506–511.