Estimating Low-Rank Region Likelihood Maps*

Gabriela Csurka[1], Zoltan Kato[2,3], Andor Juhasz[2], Martin Humenberger[1]
[1]NAVER LABS Europe, France
[2]University of Szeged, Institute of Informatics, Hungary
[3]J. Selye University, Komarno, Slovakia
{kato,jandor}@inf.u-szeged.hu {gabriela.csurka,martin.humenberger}@naverlabs.com

Abstract

*Low-rank regions capture geometrically meaningful structures in an image which encompass typical local features such as edges, corners and all kinds of regular, symmetric, often repetitive patterns, that are commonly found in man-made environment. While such patterns are challenging current state-of-the-art feature correspondence methods, the recovered homography of a low-rank texture readily provides 3D structure with respect to a 3D plane, without any prior knowledge of the visual information on that plane. However, the automatic and efficient detection of the broad class of low-rank regions is unsolved. Herein, we propose a novel self-supervised low-rank region detection deep network that predicts a low-rank likelihood map from an image. The evaluation of our method on real-world datasets shows not only that it reliably predicts low-rank regions in the image similarly to our baseline method, but thanks to the data augmentations used in the training phase it generalizes well to difficult cases (*e.g. *day/night lighting, low contrast, underexposure) where the baseline prediction fails.*

1. Introduction

Many applications such as visual localization [20, 18, 17] or structure-from-motion (SFM) [23, 10] rely on analyzing correspondences between multiple images or images and 3D point clouds (maps). In the case of SFM, the new image needs to be registered in the current reconstruction to improve or extend the 3D point cloud, in the case of visual localization a query image is used to estimate the location of the camera in a given environment (map). All methods have in common that for precise camera pose estimation (relative between two cameras or absolute between camera and 3D map) accurate pixel correspondences are crucial. In order to obtain these correspondences, relevant pixels (keypoints) in the query image and 3D points in the reference map need to be described as uniquely as possible. These so called keypoint descriptors or local features are then used to compare the pixels or 3D points and the pairs with the lowest descriptor distance are kept for further processing. All feature types share similar challenges: textureless areas, perspective distortion, occlusions, strong illumination changes, and repetitive patterns. A significant amount of work on local features and feature matching (surveys in [6, 24, 3]) was presented in the past 20 years to overcome these challenges and excellent results have been achieved.

However, it is not possible to establish reliable local correspondences between pixels in textureless areas or within repetitive pattern because of the inherent uncertainty in appearance (same color or pattern). Furthermore, feature matching can only work if the two images actually show the same scenery. If the environment changed between reference image and query image acquisition, even the best descriptor cannot find valid correspondences.

Recent work [26, 29, 12] showed that it is possible to rectify so called low-rank regions and to use them in order to estimate the camera pose relative to a plane. Furthermore, knowing such a plane induced homography between camera pairs in a camera network allows us to estimate the relative pose of a complete camera network to a 3D plane as shown in [16, 7, 14]. To be more precise, a planar region in an image is low-rank if it is possible to recover a low-rank matrix and a sparse error matrix where the first one is a canonical view of the region obtained via a *rectifying homography*. While the estimation of such a homography would require at least 4 point correspondences in a classi-

*This work was partially supported by the NKFI-6 fund through project K120366; "Integrated program for training new generation of scientists in the fields of computer science", EFOP-3.6.3-VEKOP-16-2017-0002; the Research & Development Operational Program for the project "Modernization and Improvement of Technical Infrastructure for Research and Development of J. Selye University in the Fields of Nanotechnology and Intelligent Space", ITMS 26210120042, co-funded by the European Regional Development Fund; Research & Innovation Operational Program for the Project: "Support of research and development activities of J. Selye University in the field of Digital Slovakia and creative industry", ITMS code: NFP313010T504, co-funded by the European Regional Development Fund.

cal approach, the uniqueness of our approach is that we can recover such a homography *without explicit point matching* by making use of the intrinsic structure of such low-rank textures. In an urban environment, such low rank textures are commonly found *e.g*. on planar building facades (bricks, ornaments, windows, etc.). Some examples of low-rank regions are shown in Figure 1. While classical feature extraction and matching methods are challenged by such patterns, low-rank regions provide an efficient alternative way – via the recovered rectifying homography – to estimate relative 3D structure [16, 7] or camera intrinsics [30, 14].

Thus, automatic extraction of such regions is the key to compute camera parameters or partial 3D structure, not only completely without the need of local features, but also by leveraging areas which have been avoided so far: repetitive textures. This is very helpful in man-made environments such as cities because of two reasons: (i) man-made environments contain many low-rank, regular often repetitive areas, (ii) the appearance of these areas change frequently (*e.g*. day-night, decoration of storefronts, seasons) which makes it difficult to keep them up to date in the reference map. Importantly, since using low-rank regions does not rely on pixel-wise correspondences, there is no need for storing visual information (images or descriptors). This significantly reduces the practical efforts for, *e.g*., large-scale camera localization where otherwise hundreds of thousands of images would need to be stored. Instead, the map could be a plane-based representation of the environment such as a digital elevation map [1, 4].

This paper is motivated by the fact that, to the best of our knowledge, there is no method available which actually detects low-rank regions in images. Thus, the contribution of this paper summarizes as follows:

- We propose a method which computes a low-rank likelihood map for an entire image by low-rank decomposition of image patches extracted at multiple scales (Section 3).
- We propose a self-supervised, pixel-wise probability estimation network which estimates this likelihood map directly from an image without solving extensive optimization problems (Section 4).

2. Related work

Most relevant work for this paper cover detection of repetitive structures, since they are particular cases of low-rank textures, as well as the detection of low-rank regions itself.

Repetitive structures. Several papers addressed the problem of detecting repetitive patterns, which are often based on the assumption of a single pattern repeated on a 2D (deformed) lattice. Hays *et al*. [8] proposed to use a higher-order feature matching algorithm to discover the lattices of near-regular textures in real images, wheras Park *et al*. [11] use mean-shift belief propagation for this. More popular methods are based on grouping of affine invariant local features [22, 15]. Torii *et al*. [27] also consider local feature grouping, but instead of grouping at image level they consider the entire dataset and learn to adjust the visual word weights in the soft-assigned bag-of-visual-words model. They then use the modified representation to better detect these so called *repttiles*, *i.e*. repetitive patterns of local invariant features.

The detected and rectified repetitive patterns can be used, *e.g*., for single view facade rectification [5], camera pose estimation of a query image assuming that the database of building facades is given [22], or single-view 3D reconstruction [28] assuming horizontal repetition or symmetry.

Low-rank structures. Peng *et al*. [12] propose a robust image alignment optimization framework called RASL (Robust Alignment by Sparse and Low-rank Decomposition for Linearly Correlated Images). RASL tries to find an optimal set of image domain transformations in a way that the matrix consisting of the transformed images can be decomposed into the sum of a sparse error matrix and a low-rank matrix describing the recovered aligned images. Another way of computing the low-rank matrix is the TILT (Transform Invariant Low-rank Texture) algorithm proposed by Zhang *et al*. [29]. It uses an iterative convex optimization on a given image area. TILT can be considered as a simpler version of RASL because it only uses one image and one domain transformation. Contrarily, RASL uses multiple images and multiple transformations (one for each image). TILT and RASL provide solid mathematical optimization frameworks to estimate the low-rank matrices for given regions, but none of these methods actually detects low-rank regions. The naive solution for detection would be computing TILT (as it is the simpler method) on all possible image patches (see examples in Figure 3). However, this is not feasible and impossible in practice because of the processing intensive optimization framework. Thus, the question of how to detect these low-rank regions remains. In this paper, we introduce two methods which provide a solution for this.

3. TILT-based low-rank likelihood maps

Low-rank regions capture geometrically meaningful structures in an image which encompass typical local features such as edges and corners as well as all kinds of regular, symmetric often repetitive patterns, that are commonly found in man-made environment. If these low-rank textures are on planar regions, it is possible to recover an intrinsic view $\mathbf{I_0}$, called Transform Invariant Low-rank Tex-

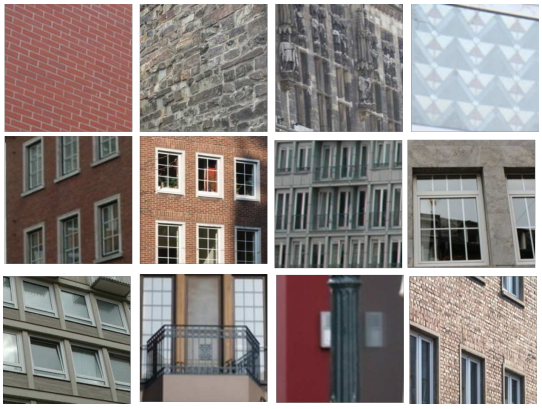

Figure 1. Low-rank pattern examples from the Aachen Day-Night dataset [19, 21].

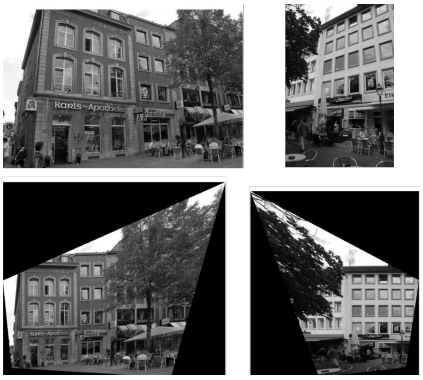

Figure 2. Images from the Aachen Day-Night dataset [19, 21] (top), where the image was rectified with TILT based on a region from the main facade (bottom).

ture (TILT) [29] with its corresponding rectifying projective transformation $\mathbf{H}$ (described in Section 3.1).

Detection of low-rank regions differs from classical detection problems because it cannot be considered as a binary decision problem. For example, a building facade composed of windows, such as the two examples in Figure 2, is low-rank but a single window cropped from the facade region is less low-rank (such as a single window in the top left image) or even not low-rank at all (such as a single window in the top right image). Strictly speaking, the degree of *low-rankness* depends on how much the intrinsic rank of a region is lower than the dimension of the rectified region containing the texture [29].

Therefore, instead of casting this as a detection problem, we propose a method to generate a low-rank likelihood map for a given image. A high value in this map means that we can center a window of a minimum size at that position such that the corresponding image region represents a transformed low-rank texture.

To obtain such maps, we estimate a *"low-rankness"* score by using TILT [29] on a sliding window at multiple scales. We then use a weighted kernel density estimate (wKDE) to build the likelihood map.

3.1. Transform invariant low-rank textures (TILT)

The TILT algorithm uses a convex optimization technique that enables robust recovery of a high-dimensional low-rank matrix despite gross sparse errors and, for planar regions, the rectifying homography.

Let us assume, that our cameras see a 3D plane π with a *low-rank* texture (see some examples in Figure 1) and the world coordinate system is attached to this plane (being the $Z = 0$ plane). Consider a 2D texture as a function R_0. According to [29], R_0 is a low-rank texture, if the family of one-dimensional functions $\{R_0(x, y_0) \mid y_0 \in \mathbb{R}\}$ span a finite low-dimensional linear subspace, *i.e.* $dim(span\{R_0(x, y_0)|y_0 \in \mathbb{R}\}) \leq k$, for some small positive integer k compared to the region size. Given a low-rank texture, obviously its rank is invariant under any scaling of the function, as well as scaling or translation in the x and y coordinates. Hence, two low-rank textures are equivalent if they are scaled and translated versions of each other, *i.e.* $R_0(x, y) \sim cR_0(ax + t_1, by + t_2)$.

In practice, we are never given the 2D texture as a continuous function in $\mathbb{R}$ but discretized as a matrix denoted by $\mathbf{R_0}$. Furthermore, given an image, we only have a transformed version of R_0 denoted by R. Formally, there is a h, such that $h(R) \sim R_0$, or in the discrete case we have $\mathbf{HR} \sim \mathbf{R_0}$, where $\mathbf{H} \in \mathbb{R}^{3\times3}$ is a homography.

In a real application, the R_0 pattern is unknown and, due to occlusions and other noises, its image is a not perfectly transformed versions of the pattern. This type of error can be modeled with a sparse matrix $\mathbf{S}$. Aiming to determine the planar homography $\mathbf{H}$, Zhang *et al.* [29] proposed to solve it as a robust rank minimization problem

$$\begin{aligned} \min_{\mathbf{R_0},\mathbf{S},\mathbf{H}} \quad & \text{rank}(\mathbf{R_0}) + \lambda\|\mathbf{S}\|_1 \\ s.t. \quad & \mathbf{HR} = \mathbf{R_0} + \mathbf{S}, \end{aligned} \tag{1}$$

where R is the observed low-rank texture region, $\mathbf{R_0}$ is a low-rank matrix approximating the intrinsic view R_0 of the low-rank texture, $\mathbf{S}$ is a sparse error matrix encoding the difference between $\mathbf{R_0}$ and $\mathbf{HR}$, and $\|.\|_1$ is the L1 norm.

To solve this problem, [29] proposed to use an iterative Augmented Lagrange Multipliers (ALM) method as described in Algorithm 1. It contains two interleaved iterative optimization processes which makes it rather costly. The output of TILT is the estimation of the rectified low-rank texture $\mathbf{R}_0^*$ (also called canonical view), the error matrix $\mathbf{S}^*$ and the rectifying homography $\mathbf{H}^*$. In Figure 2 we show examples of images rectified with $\mathbf{H}^*$ obtained from a manually selected, planar low-rank region.

Algorithm 1 The TILT Algorithm [29].

Input: Region R represented by the matrix $\mathbf{R}$, and $\lambda > 0$.
Initialize: h^0 with the identity ($\mathbf{H}^0 = \mathbf{I}$) and $t = 0$.
while not converged **do**
Step 1: Normalize $\mathbf{H}^t\mathbf{R}$ such that $\|\mathbf{H}^t\mathbf{R}\|_F = 1$
Step 2: Compute the Jacobian of R w.r.t. the transformation parameters at h^t.

$$\nabla_h R \leftarrow \frac{\partial}{\partial h'}\Big(\frac{h'(R)}{\|h'(R)\|_F}\Big)\Big|_{h'=h^t}$$

Step 3: Solve the problem (1) using ALM described in Algorithm 2 to get $\mathbf{R}_0^t$, $\mathbf{S}^t$ and Δ_h^t.
Step 4: Update $\mathbf{H}^{t+1} = \mathbf{H}^t + \Delta_h^t$.
end while
Output: Optimal solution $\mathbf{R}_0^*$, $\mathbf{S}^*$ and $\mathbf{H}^*$.

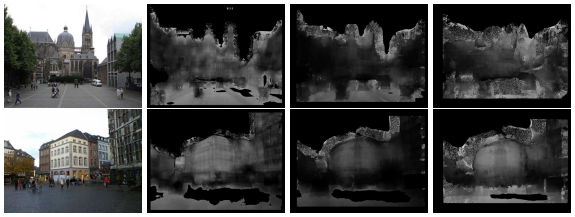

Figure 3. Low-rank score-maps obtained via (6) using TILT with sliding windows on every pixel with window sizes of 50, 100, and 150 pixels (CPU times to generate these maps were between 5 and 12 days depending on window size and image content).

3.2. Building the low-rank likelihood map

Let us assume that in the ideal case we can get a likelihood for each window reflecting the probability of containing a low-rank texture. Even in this case, in order to get the likelihood map for the full image would mean that we sample all possible windows at each pixel position (see examples obtained with different window sizes in Figure 3 for a few test images). As already computing such a score on a single window is costly (because of the two layer iterative optimization used in TILT), it is quasi impossible to directly compute this map (it took several days to obtain the maps in Figure 3 for a single image using the Matlab implementation[1] provided by the authors of [29]).

Instead, we estimate it by considering overlapping sliding windows at multiple scales and predefined steps. At each such position, we fit a local Gaussian with a "low-rankness" score (defined below) selecting a bandwidth depending on the window size. This can be seen as a weighted kernel density estimation (wKDE) of the targeted probability density function. Kernel density estimation is a fundamental data smoothing problem where inferences about the population are made based on a finite data sample, which is

[1] We used the MATLAB code available at https://people.eecs.berkeley.edu/~yima/matrix-rank/tilt.html.

Algorithm 2 The ALM Solver [29].

Input: The current transformed region $h^t(R)$ represented by $\mathbf{R}_t = \mathbf{H}^t\mathbf{R}$.
Input: The Jacobian $\mathbf{J}_t = \nabla_h R$ and its Moore-Penrose pseudo-inverse denoted by $\mathbf{J}_t^\dagger$;
Input: The weight $\lambda > 0$ and the element-wise shrinkage operator: $\sigma(\mathbf{A}, \mu)(i,j) = sign(\mathbf{A}(i,j))(|\mathbf{A}(i,j)| - \mu)$.
Initialize: $k = 0$, $\rho > 1$ $\mu_0 > 0$ and $\mathbf{Y}^0 = \mathbf{S}^0 = \Delta_h^0 = \mathbf{0}$.
while not converged **do**
$(\mathbf{U}_k, \mathbf{D}_k, \mathbf{V}_k) = \text{SVD}(\mathbf{R}_t + \mathbf{J}_t\Delta_h^k + \mu_k\mathbf{Y}^k - \mathbf{S}^k)$;
$\mathbf{R}_0^{k+1} = \mathbf{U}_k\sigma(\mathbf{D}_k, \mu^k)\mathbf{V}_k^\top$;
$\mathbf{S}^{k+1} = \sigma(\mathbf{R}^t + \mathbf{J}^t\Delta_h^k + \mu_k\mathbf{Y} - \mathbf{R}_0^k, \lambda\mu_k)$;
$\mathbf{\Delta}_h^{k+1} = \mathbf{J}_t^\dagger(-\mathbf{R}_t + \mathbf{R}_0^{k+1} + \mathbf{S}^{k+1} - \mu_k\mathbf{Y}^k)$;
$\mathbf{Y}^{k+1} = \mathbf{Y}^k + (\mathbf{R}_t - \mathbf{R}_0^{k+1}\mathbf{J}_t\mathbf{\Delta}_h^{k+1} - \mathbf{S}^{k+1})/\mu_k$;
$\mu_{k+1} = \mu_k/\rho$.
end while
Output: Optimal solution $\mathbf{R}_0^*$, $\mathbf{S}^*$ and Δ_h^*.

in our case the set of sampled windows. If $\{x_1, x_2, \ldots, x_n\}$ represent random samples drawn from some probability distribution q, we can approximate the shape of q by using a kernel density estimator

$$Q(x|P) = \frac{1}{n}\sum_{i=1}^{n} p_i\mathcal{G}(w_i, \sigma), \tag{2}$$

where, in our case, the kernels are zero mean bivariate Gaussians centered on the sampled window w_i with variance σ depending on the window size. p_i are the probability weights obtained as described below.

Given any image window, considered as a matrix, the optimization problem described in (1) can be solved efficiently by ALM, yielding a triplet $(\mathbf{R}_0, \mathbf{S}, \mathbf{H})$. In case of a quasi-planar low-rank texture, $\mathbf{R}_0$ is the intrinsic low-rank structure, $\mathbf{S}$ is a sparse matrix, and $\mathbf{H}^{-1}$ is the rectifying transformation. In case of a non-low-rank region, the algorithm will force to find an $\mathbf{R}_0$ with the lowest rank possible at the price of an increased error making $\mathbf{S}$ less sparse.

In order to construct the likelihood map, we first run TILT[1] over a set of sliding windows of different sizes[2] $l \times l$, with $l \in \{50, 100, 150\}$, and a step size of $l/2$ between neighboring windows.

The algorithm provides us the following quantities for each window w_i^l: (1) $\mathbf{A}_i^l$ – low-rank matrix, (2) $\mathbf{S}_i^l$ – sparse matrix, (3) $\mathbf{H}_i^l$ – rectifying homography and (4) f_i^l – the residual of the factorization in (1). Using these quantities, we can characterize the "*low-rankness*" of the window w_i^l via a strictly non-negative error (or energy)

$$e_i^l = r_i^l + s_i^l + f_i^l, \tag{3}$$

where r_i^l is the rank of $\mathbf{A}_i^l$ divided by l and s_i^l is the L1 norm of the sparse matrix $\mathbf{S}_i^l$. The energy defined in (3)

[2] We considered fixed window sizes over images that were resized to fit in 800×1200 or 1200×800.

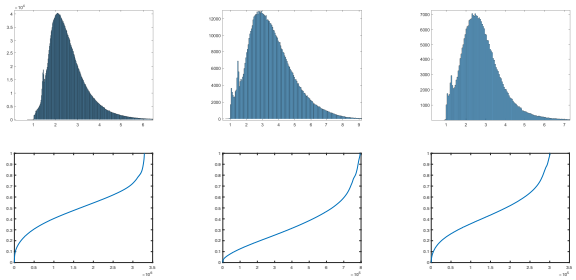

Figure 4. Top: Histogram of e_i^l (3) for window sizes of 50, 100, and 150. Bottom: Probability values constructed as (6) using (5) for layers 50, 100, and 150.

is then used to define a standard exponential distribution $P_i^l = \exp(-e_i^l)$. Note that a homogeneous region is low-rank (providing a low e_i^l value) but because of the lack of intensity patterns they are useless to estimate a well-defined rectifying homography.

Therefore, we want to impose $P_i^l = 0$ for homogeneous regions. To detect if the window corresponds to a homogeneous region we consider the binary edge map $\mathbf{E}_i^l$ of the window w_i^l and we check if

$$h_i^l = \frac{1}{l^2}\|\mathbf{E}_{i,j}^l\|_1 \tag{4}$$

is greater than a homogeneity threshold τ (set to 0.04).

Considering all e_i^l values that are obtained from all windows in our dataset processed with TILT and not rejected by the homogeneity constraint, we build the histogram corresponding to various window sizes. On these histograms, shown in Figure 4 (top), we can identify two modes: one narrow mode with smaller $e_{i,j}^l$ values (corresponding to *low-rank* region candidates) and a much wider one with larger values. As we want to stretch the first mode in order to construct a well calibrated P_i^l which is suitable to infer *low-rank* regions, we modify the low-rankness score in (3) as

$$e_i^l = \max(0, \alpha(r_i^l + s_i^l + f_i^l - 1)), \tag{5}$$

where we set α to 0.75. Then, p_i^l combined with the homogeneity constraint becomes

$$p_i^l = \exp(-e_i^l)\delta(h_i^l > \tau), \tag{6}$$

where $\delta()$ is the Kronecker delta. Figure 4 (bottom) shows the sorted probability values obtained over all windows.

In order to define the probability map over the entire image, we use wKDE with Gaussian kernels to propagate these values as

$$\mathbf{P}^l = \frac{1}{N^l}\sum_{i^l=N^l} p_i^l \mathcal{G}(w_i^l, \sigma^l), \tag{7}$$

Algorithm 3 The TILT likelihood map generation.

Input: The image I and the set of window sizes $l \in \{50, 100, 150\}$.
Pre-process: Convert the image to grayscale and resize it to get $min(width, height) = 800$.
Pre-process: Build edge map $\mathbf{E}$ using Canny edge detector.
Initialize: The likelihood map $\mathbf{P}$ and $\mathbf{P}^l$ with zeros.
for all sliding window w_i^l sampled with step $l/2$. **do**
  Run Algorithm 1 to get $\mathbf{R}_0^*$, $\mathbf{S}^*$ and $\mathbf{H}^*$
  Compute e_i^l using (5).
  Compute the homogeneity h_i^l for the window using (4).
  Compute p_i^l with (6).
  Add $p_i^l \mathcal{G}(w_i^l, \sigma^l)$ to $\mathbf{P}^l$ as described in (7).
end for
Normalize $\mathbf{P}^l$ for each $l \in \{50, 100, 150\}$.
Compute $\mathbf{P}$ with (8).
Output: The likelihood map $\mathbf{P}$ obtained for I.

where N^l denotes the number of windows in an image at level l, w_i^l are the sliding windows and σ^l is function of the window size l. In order to ensure that the values of p^l are taken from the probability distribution over *all images* at level l, we divide $\mathbf{P}^l$ by its maximum value and then multiply it by the maximum p_i^l obtained in the whole dataset. Finally, the probability maps, obtained at different levels $l \in \{50, 100, 150\}$, are averaged as

$$\mathbf{P} = \frac{1}{\sum_l \frac{1}{l}}\sum_l \frac{1}{l}\mathbf{P}^l. \tag{8}$$

The above steps are summarized in Algorithm 3. Note that we used a stronger weight for $\mathbf{P}^l$ obtained with a smaller window size for two reasons. On the one hand we have much more sliding windows that contributed to generate $\mathbf{P}^{50}$ than $\mathbf{P}^{100}$ or $\mathbf{P}^{150}$. On the other hand, due to the choice of our σ that depends on the window size, the Gaussians corresponding to the window size 50 are more localized than the Gaussians corresponding to bigger windows yielding much smoother $\mathbf{P}^l$ maps.

4. Deep low-rank region detection network

Obtaining the probability map as described in Section 3.2 is extremely costly[3] because we have to run two incorporated iterative optimizations in TILT and within the ALM algorithm for every sliding window and at several scales. Therefore, we propose to train a deep neural network that learns to directly predict such maps from a given image, trained with a set of likelihood maps generated with the method from Section 3.2.

While these maps are only approximated probability distributions and hence cannot be considered as perfect ground

[3]Running the MATLAB version of TILT for all sliding windows with step $l/2$ and 3 scales takes about 25 mins for an image.

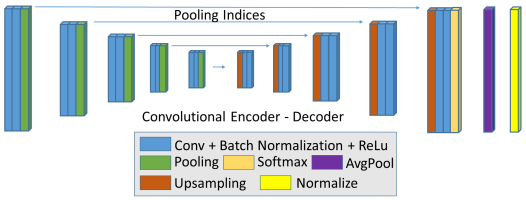

Figure 5. Modified Segnet [2].

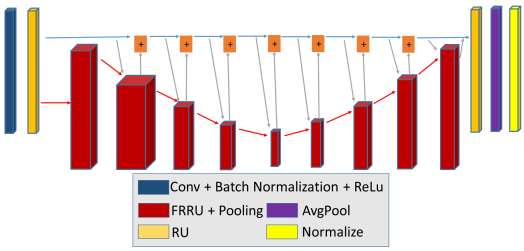

Figure 6. Modified Full-Resolution Residual Networks (FRRN) [13].

truth, we hope that if sufficient training samples are shown to the network, it is able to not only learn to reproduce the map estimated with TILT, but also to generalize between the training samples and learn to recognize implicit low-rank non-homogeneous structures.

Willing to get a pixel level output, we considered models used for image segmentation as network architectures, which include both down- and upscaling to obtain an output feature map of the same resolution as the input image. We experimented with two different architectures: Segnet [2] (see Figure 5) and Full-Resolution Residual Networks (FRRN) [13] (see Figure 6).

In both cases, we modified the network as follows: Since color is not relevant for detecting low-rank regions, we first convert the input images to single-channel grayscale. Similarly, since we only consider a single feature map output, we smooth the map with an average pooling layer and finally we normalize the values in the map to be between 0 and 1. Then, instead of using the cross-entropy loss as the objective function for training the network, we use a loss based on Kullback-Leibler (KL) divergence.

Indeed, we constructed $\mathbf{P}$ as a wKDE estimate of a probability distribution of low-rank regions defined over the image plane via the method outlined in Section 3.2, *i.e.* it assigns a likelihood $P(i,j)$ to each pixel (i,j) which characterizes the probability of this pixel being part of a *low-rank* region. Our aim is to obtain the output feature map $\mathbf{F}$ of the network to be similar to $\mathbf{P}$ in a probabilistic closeness sense. Therefore, as loss we use the Kullback-Leibler (KL) divergence from the output feature map $\mathbf{F}$ to the input likelihood map $\mathbf{P}$, which measures the difference between the target probability distribution $\mathbf{F}$ and the reference probability distribution $\mathbf{P}$ as

$$D(\mathbf{P}||\mathbf{F}) = \sum_{(i,j)\in I} \widehat{\mathbf{P}}(i,j) \log \frac{\widehat{\mathbf{P}}(i,j)}{\widehat{\mathbf{F}}(i,j)}, \qquad (9)$$

where $\widehat{\mathbf{P}}$ and $\widehat{\mathbf{F}}$ are normalized likelihood maps such that the sum of all values is 1, thus making the maps equivalent to probability distributions conditioned on the given image. Note that KL divergence is not a metric as it is anti-symmetric and does not obey the triangle inequality. However, it has important properties which makes it appropriate to measure the difference between probability distributions. Furthermore, it is always non-negative: $D(P||F) \geq 0$, being 0 only when $P \equiv F$.

5. Experimental results

Training. To train our model, we used the Aachen Day-Night dataset presented in [19, 21]. Originally proposed for visual localization, this dataset consists of a training set (with ground truth camera poses) and a test set (without ground truth camera poses). We randomly split the Aachen training set into three groups of images: 500 for validation, 500 for testing, and 3328 remain for training. In addition to the test images from the split above, we used the images *Day (milestone)* and *Night (nexus5x)* from the official Aachen test set for testing. In order to keep the network's memory consumption and training time reasonable, we reduced the image size to 800×640 by randomly alternating between re-scaling with zero-padding and random cropping. Even if this strategy already introduced variability in the training set, we additionally applied various data augmentation methods such as flip, rotation, gamma, brightness, contrast, and saturation change.

We built our models modifying the PyTorch implementation provided by [25]. We used the SGD optimizer with a learning rate between 1.0e-7 and 1.0e-12, weight decay equal to 0.0005, and a momentum of 0.99. We used small batch sizes 2 or 4 and considered 200000 iterations evaluating the model every 500 iteration on the validation set. We kept only the model that performed best on the validation set, which often was obtained after around 150000 iterations (if the model converged). The results shown in the paper were obtained with a model trained with batch sizes 4 and learning rate equal to 1.0e-9.

Low-rank region detection. Following a standard procedure, we used the 3328 images of our training set to train the model, we used the validation set to select the parameters, and finally we used our test set (including the official Aachen Day-Night test images) to evaluate the model. Note that in all our experiments, we consider the maps obtained

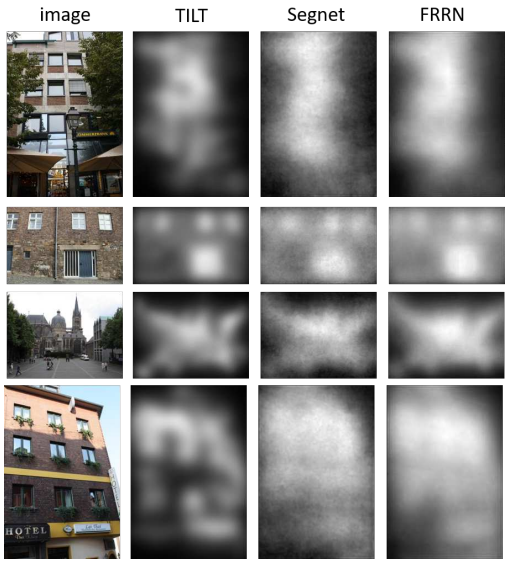

Figure 7. Examples of predicted low-rank likelihood maps for test images obtained with TILT+wKDE described in Section 3.2 (second column), with SegNet-based deep network (third column), and with the FRNN-based deep network (fourth column).

with TILT+wKDE as ground truth and compare the maps obtained with the network to these maps using KL divergence.

We show the average KL divergence over the validation and test sets in Table 1 and on the *Day (milestone)* and *Night (nexus5x)* sets in Table 2.

Table 1. Average KL divergence values on our train, val and test splits of the Aachen Day-Night dataset.

model	train	val	test
SegNet	0.0321	0.0364	0.0337
FRNN	0.0280	0.0315	0.0312

Table 2. Average KL divergence values on the *Day (milestone)* and *Night (nexus5x)* set from the official Aachen-Day-Night dataset.

model	Day (milestone)	Night (nexus5x)
SegNet	0.09276	0.0617
FRNN	0.07752	0.0415

Furthermore, in Figures 7 and 8, we show low-rank likelihood maps obtained with TILT+wKDE-based likelihood map generation described in Section 3.2, and likelihood maps predicted by Segnet and FRNN-based deep networks.

As can be seen in Tables 1 and 2 and Figures 7 and 8, the FRNN-based architecture provides smoother output maps which in addition have a lower KL divergence when comparing them to the likelihood maps obtained with TILT+wKDE map generation.

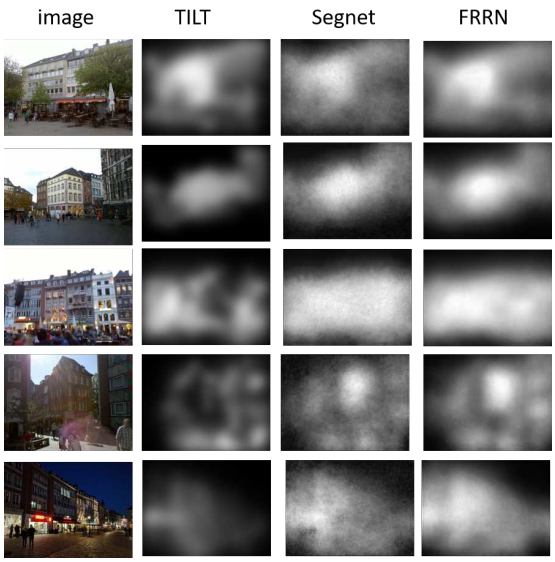

Figure 8. Examples of predicted low-rank likelihood maps for images from *Day (milestone)* and *Night (nexus5x)* obtained with TILT+wKDE (second column), Segnet-based (third column) and FRNN-based (fourth column) deep network.

Table 3. Average KL divergence values on sequences taken from the CambridgeLandmarks dataset.

model	GreatCourt (seq2)	OldHospital (seq1)
SegNet	1.3846	0.8868
FRNN	1.0910	0.8240

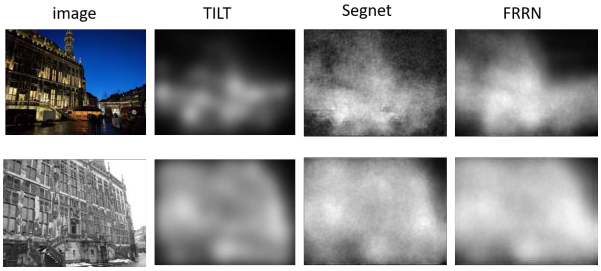

Figure 9. Example image pair from Aachen, where the same scene is visible day (bottom) and night (top) together with their TILT+wKDE, SegNet and FRRN likelihood maps.

Generalization to other datasets. In order to test how the model generalizes to other datasets, we consider sequences from the CambridgeLandmarks dataset [9]. We show results for *GreatCourt (seq2)* and *OldHospital (seq1)* in Table 3. Comparing these numbers with the ones from Table 2, we see larger KL divergence values in Table 3. However, while looking at some representative results in Figure 10, both Segnet and FRRN provide relevant, consistent, and better maps than the reference (TILT+wKDE). In order to understand this, let us have a closer look at what is ex-

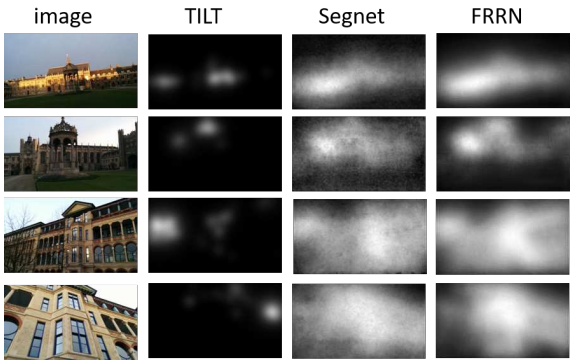

Figure 10. Examples of predicted low-rank likelihood maps for images from *GreatCourt (seq2)* (top two rows) and *OldHospital (seq1)* (bottom two rows) and obtained with TILT+wKDE (second column), Segnet-based (third column) and FRNN-based (fourth column) deep network..

actly captured by our likelihood map outlined in Section 4.

First, the images used for TILT were *not* pre-processed for exposure or contrast enhancement. Hence, when the image is too dark and/or blurry, the edge detector provides only a few edges yielding low h_i^l in (6) and considering such windows to be homogeneous (*i.e.* $p_i^l = 0$) which prevents them from contributing to the constructed likelihood map. On the contrary, thanks to our data augmentation which varies gamma, luminosity, and contrast, the deep model is less affected by this and yields a better likelihood map than TILT+wKDE (see Figure 10). As a result, the same structures under drastically different lighting conditions (see Figure 9 for a day/night example) are correctly detected as low-rank, in spite of the extreme light changes.

Second, based on an experimental analysis of various window sizes and resolution levels, we only considered fixed-size sliding windows at three scales which captured most of the low-rank properties while keeping the computational cost at a reasonable scale. Therefore, the output of TILT+wKDE depends on the considered scales. Our deep models are less sensitive to this, thanks to the fact that we incorporated geometric transformations including scale change in our data augmentation (see two bottom rows in Figure 10). Both observations clearly show the strength of the learned estimation over the hand-crafted computation.

Potential applications. The focus of the paper is the unsolved problem of detection of low-rank regions by introducing the estimation of a novel likelihood map. When such a region is correctly detected, one can easily and efficiently get the rectifying homography. Figure 11 shows an example of a low-rank region extracted at a local maximum of our FRRN map together with its rectification using TILT. As we can see, while TILT is unable to *detect* low-rank regions, it can efficiently estimate a rectifying homography of the bounding boxes around the local maxima of our predicted likelihood maps. Such homographies have important applications, *e.g.* camera pose estimation, matching, and 3D reconstruction [7, 14, 16, 30]. An example of camera pose estimation w.r.t. a 3D plane is shown in Figure 12.

Figure 11. Low-rank region extracted at a local maximum of our predicted FRRN map together with its rectification using TILT (runtime: 0.8145s in Matlab).

Figure 12. Purple: detected bounding box used for relative pose estimation w.r.t. the 3D plane of the low-rank region. Green: GT camera. Blue: camera factorized from the rectifying homography. Rotation error: 2.4°, translation error: 1.5° (angle w.r.t. the GT translation because the absolute length of the translation cannot be obtained from an homography).

6. Conclusion

We have shown that low-rank regions can be robustly detected using a deep neural network which estimates a probability distribution. The network can be trained in a self-supervised manner using likelihood maps computed with TILT on an image grid at multiple scales. This is significantly faster and easier to use in practice, since the low-rank regions are directly estimated from the image without solving complex optimization problems. Furthermore, evaluation on two real-world datasets have shown that we can achieve very similar results to our baseline method (TILT+wKDE). More importantly, the results show that the learned estimation can even handle more challenging cases where the hand-crafted computation fails. We strongly believe that this method will enable the usage of low-rank regions in many applications since to date the full potential of, *e.g.* camera pose estimation without depending on pixel-wise correspondences, is not yet exploited. Furthermore, our proposed detection network can further be extended to directly return the rectifying homography.

References

[1] Clemens Arth, Christian Pirchheim, Jonathan Ventura, Dieter Schmalstieg, and Vincent Lepetit. Instant outdoor localization and slam initialization from 2.5d maps. *Transactions on Visualization and Computer Graphics (TVCG)*, 21(11):1309–1318, 2015. 2

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 39(12):2481–2495, 2017. 6

[3] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[4] Mayank Bansal and Kostas Daniilidis. Geometric urban geo-localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3978–3985, 2014. 2

[5] David M. Chen, Georges Baatz, Kevin Köser, Sam S. Tsai, Ramakrishna Vedantham, Timo Pylvänäinen, Kimmo Roimela, Xin Chen, Jeff Bach, Marc Pollefeys, Bernd Girod, and Radek Grzeszczuk. City-scale landmark identification on mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 2

[6] Gabriela Csurka, Christopher R. Dance, and Martin Humenberger. From handcrafted to deep local features. *arXiv preprint arXiv:1807.10254*, 2018. 1

[7] Robert Frohlich and Zoltan Kato. Simultaneous multi-view relative pose estimation and 3D reconstruction from planar regions. In Gustavo Carneiro and Shaodi You, editors, *ACCV Workshop on Advanced Machine Vision for Real-life and Industrially Relevant Applications*, volume 11367 of *Lecture Notes in Computer Science*, pages 467–483, Perth, Australia, Dec. 2018. Springer. 1, 2, 8

[8] James Hays, Marius Leordeanu, Alexei A Efros, and Yanxi Liu. Discovering texture regularity as a higher-order correspondence problem. In *European Conference on Computer Vision (ECCV)*, 2006. 2

[9] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *International Conference on Computer Vision (ICCV)*, 2015. 7

[10] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *International Conference on Computer Vision (ICCV)*, 2013. 1

[11] Minwoo Park, Kyle Brocklehurst, Robert T. Collins, and Yanxi Liu. Deformed lattice detection in real-world images using mean-shift belief propagation. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(10):804–1816, 2009. 2

[12] Yigang Peng, Arvind Ganesh, John Wright, Wenli Xu, and Yi Ma. RASL: Robust Alignment by Sparse and Low-rank decomposition for linearly correlated images. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(11):2233–2246, 2012. 1, 2

[13] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6

[14] James Pritts, Zuzana Kukelova, Victor Larsson, and Ondřej Chum. Rectification from radially-distorted scales. In *Asian Conference on Computer Vision (ACCV)*, 2018. 1, 2, 8

[15] James Pritts, Jiří Matas, and Ondřej Chum. Detection, rectification and segmentation of coplanar repeated patterns. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[16] Zsolt Santa and Zoltan Kato. Pose estimation of ad-hoc mobile camera networks. In *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, Hobart, Tasmania, Australia, Nov. 2013. IEEE. 1, 2, 8

[17] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[18] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 39(9):1744–1756, 2016. 1

[19] Torsten Sattler, Will Maddern, Carl Toft, Torii Akihiko, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomáš Pajdla. Benchmarking 6dof outdoor visual localization in changing conditions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 6

[20] Torsten Sattler, Akihiko Torii, Josef Sivic, Marc Pollefeys, Hajime Taira, Masatoshi Okutomi, and Tomas Pajdla. Are large-scale 3D models really necessary for accurate visual localization? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[21] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image Retrieval for Image-Based Localization Revisited. In *BMVA British Machine Vision Conference (BMVC)*, 2012. 3, 6

[22] Grant Schindler, Panchapagesan Krishnamurthy, Roberto Lublinerman, Yanxi Liu, and Frank Dellaert. Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 2

[23] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[24] Johannes L. Schönberger, Hans Hardmeier, Torsten Sattler, and Marc Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1

[25] Meet P. Shah. Semantic segmentation architectures implemented in PyTorch. https://github.com/meetshah1995/pytorch-semseg, 2017. 6

[26] Pablo Sprechmann, Alex M. Bronstein, and Guillermo Sapiro. Learning efficient sparse and low rank models. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 37(9):1821–1833, 2015. 1

[27] Akihiko Torii, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. Visual place recognition with repetitive structures. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 37(11):2346–2359, 2015. 2

[28] Changchang Wu, Jan-Michael Frahm, and Marc Pollefeys. Repetition-based dense single-view reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 2

[29] Zhengdong Zhang, Arvind Ganesh, Xiao Liang, and Yi Ma. TILT: Transform Invariant Low-rank Textures. *International Journal of Computer Vision*, 99(1):1–24, 2012. 1, 2, 3, 4

[30] Zhengdong Zhang, Yasuyuki Matsushita, and Yi Ma. Camera calibration with lens distortion from low-rank textures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 2, 8