# Learning Covariant Feature Detectors

Karel Lenc and Andrea Vedaldi

Department of Engineering Science, University of Oxford
✉ karel@robots.ox.ac.uk vedaldi@robots.ox.ac.uk

**Abstract.** Local covariant feature detection, namely the problem of extracting viewpoint invariant features from images, has so far largely resisted the application of machine learning techniques. In this paper, we propose the first fully general formulation for learning local covariant feature detectors. We propose to cast detection as a regression problem, enabling the use of powerful regressors such as deep neural networks. We then derive a *covariance constraint* that can be used to automatically learn which visual structures provide stable anchors for local feature detection. We support these ideas theoretically, proposing a novel analysis of local features in term of geometric transformations, and we show that all common and many uncommon detectors can be derived in this framework. Finally, we present empirical results on translation and rotation covariant detectors on standard feature benchmarks, showing the power and flexibility of the framework.

## 1 Introduction

Image matching, i.e. the problem of establishing point correspondences between two images of the same scene, is central to computer vision. In the past two decades, this problem stimulated the creation of numerous *viewpoint invariant local feature detectors*. These were also adopted in problems such as large scale image retrieval and object category recognition, as a general-purpose image representations. More recently, however, deep learning has replaced local features as the preferred method to construct image representations; in fact, the most recent works on local feature descriptors are now based on deep learning [10,46].

Differently from *descriptors*, the problem of constructing local feature *detectors* has so far largely resisted machine learning. The goal of a detector is to extract stable local features from images, which is an essential step in any matching algorithm based on sparse features. It may be surprising that machine learning has not been very successful at this task given that it has proved very useful in many other detection problems. We believe that the reason is the difficulty of devising a learning formulation for viewpoint invariant features.

To clarify this difficulty, note that the fundamental aim of a local feature detector is to extract the same features from images regardless of effects such as viewpoint changes. In computer vision, this behavior is more formally called *covariant detection*. Handcrafted detectors achieve it by anchoring features to image structures, such as corners or blobs, that are preserved under a viewpoint change. However, there is no *a–priori* list of what visual structures constitute useful anchors. Thus, an algorithm

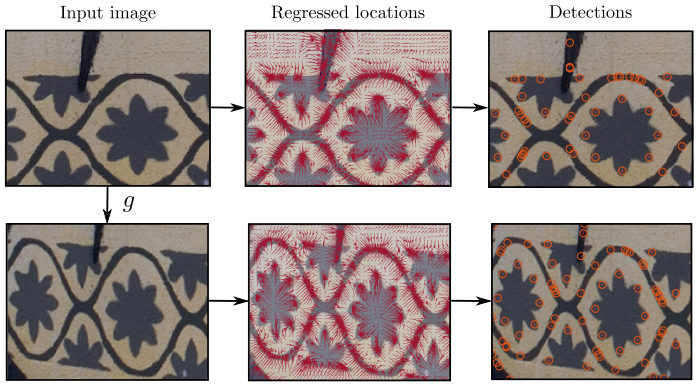



Fig. 1: *Detection by regression.* We train a neural network $\phi$ that, given a patch $\mathbf{x}|_p$ around each pixel $p$ in an image, produces a displacement vector $h_p = \phi(\mathbf{x}|_p)$ pointing to the nearest feature location (middle column). Displacements from nearby pixels are then pooled to detect features (right column). The neural network is trained in order to be covariant with transformations $g$ of the image (bottom row). Best viewed on screen. Image data from [4].

must not only learn the appearance of the anchors, but needs to determine what anchors are in the first place. In other words, the challenge is to learn simultaneously a detector together with the detection targets.

In this paper we propose a method to address this challenge. Our first contribution is to introduce a novel *learning formulation for covariant detectors* (Sect. 2). This is based on two ideas: i) defining an objective function in term of a *covariance constraint* which is anchor-agnostic (Sect. 2.1) and ii) formulating detection as a *regression problem*, which allows to use powerful regressors such as deep networks for this task (Fig. 1).

Our second contribution is to support this approach theoretically. We show how covariant feature detectors are best understood and manipulated in term of image transformations (Sect. 2.2). Then, we show that, geometrically, different detector types can be characterized by which transformations they are covariant with and, among those, which ones they fix and which they leave undetermined (Sect. 2.3). We then show that this formulation encompasses all common and many uncommon detector types and allows to derive a covariance constraint for each one of them (Sect. 2.4).

Our last contribution is to validate this approach empirically. We do so by first discussing several important implementation details (Sect. 3), and then by training and assessing two different detector types, comparing them to off-the-shelf detectors (Sect. 4). Finally, we discuss future extensions (Sect. 5).

### 1.1 Related work

Covariant detectors differ by the type of features that they extract: points [11,16,36,6], circles [17,19,23], or ellipses [18,42,2,35,22,24]. In turn, the type of feature determines which class of transformations that they can handle: Euclidean transformations, similarities, and affinities.

Another differentiating factor is the type of visual structures used as anchors. For instance, early approaches used corners extracted from an analysis of image edglets [29,8,34]. These were soon surpassed by methods that extracted corners and other anchors using operators of the image intensity such as the *Hessian of Gaussian* [3] or the *structure tensor* [7,11,47] and its generalizations [40]. In order to handle transformations more complex than translations and rotations, scale selection methods using the *Laplacian/Difference of Gaussian* operator (L/DoG) were introduced [19,23], and further extended with *affine adaptation* [2,24] to handle full affine transformations. While these are probably the best known detectors, several other approaches were explored as well, including parametric feature models [9,28] and using self-dissimilarity [38,13].

All detectors discussed so far are *handcrafted*. Learning has been mostly limited to the case in which detection anchors are defined *a-priori*, either by manual labelling [14] or as the output of a pre-existing handcrafted detetctor [5,31,39,12], with the goal of accelerating detection. Closer to our aim, [32] use simulated annealing to optimise the parameters of their FAST detector for repeatability. To the best of our knowledge, the only line of work that attempted to learn repeatable anchors from scratch is the one of [41,25], who did so using genetic programming; however, their approach is much more limited than ours, focusing only on the repeatability of corner points.

More recently, [44] learns to estimate the orientation of feature points using deep learning. Contrary to our approach, the loss function is defined on top of the local image feature descriptors and is limited to estimating the rotation of keypoints. The work of [45,27,37] also use Siamese deep learning architectures for local features, but for local image feature *description*, whereas we use them for feature *detection*.

## 2 Method

We first introduce our method in a special case, namely in learning a basic corner detector (Sect. 2.1), and then we extend it to general covariant features (Sect. 2.2 and 2.3). Finally, we show how the theory applies to concrete examples of detectors (Sect. 2.4).

### 2.1 The covariance constraint

Let $\mathbf{x}$ be an image and let $T\mathbf{x}$ be its version translated by $T \in \mathbb{R}^2$ pixels. A corner detector extracts from $\mathbf{x}$ a (small) collection of points $\mathbf{f} \in \mathbb{R}^2$. The detector is said to be covariant if, when applied to the translated image $T\mathbf{x}$, it returns the translated points $\mathbf{f} + T$. Most covariant detectors work by anchoring features to image structures that, such as corners, are preserved under transformation. A challenge in defining anchors is that these must be general enough to be found in most images and at the same time sufficiently distinctive to achieve covariance.

Anchor extraction is usually formulated as a *selection problem* by finding the features that maximize a handcrafted figure of merit such as Harris' cornerness, the Laplacian of Gaussian, or the Hessian of Gaussian. This indirect construction makes learning anchors difficult. As a solution, we propose to regard feature detection not as a selection problem but as a *regression one*. Thus the goal is to learn a function $\psi : \mathbf{x} \mapsto \mathbf{f}$ that

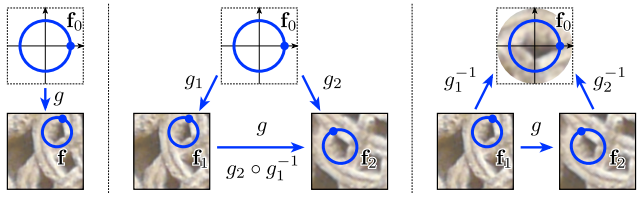



Fig. 2: **Left**: an oriented circular frame $\mathbf{f} = g\mathbf{f}_0$ is obtained as a unique similarity transformation $g \in G$ of the canonical frame $\mathbf{f}_0$, where the orientation is represented by the dot. Concretely, this could be the output of the SIFT detector after orientation assignment. **Middle**: the detector finds feature frames $\mathbf{f}_i = g_i\mathbf{f}_0, g_i = \phi(\mathbf{x}_i)$ in images $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively due to covariance, matching the features allows to recover the underlying image transformation $\mathbf{x}_2 = g\mathbf{x}_1$ as $g = g_2 \circ g_1^{-1}$. **Right**: equivalently, then inverse transformations $g_i^{-1}$ normalize the images, resulting in the same canonical view.

directly maps an image (patch[1]) $\mathbf{x}$ to a corner $\mathbf{f}$. The key advantage is that this function can be implemented by any regression method, including a deep neural network.

This leaves the problem of defining a learning objective. This would be easy if we had example anchors annotated in the data; however, our aim is to discover useful anchors automatically. Thus, we propose to *use covariance itself as a learning objective*. This is formally captured by the *covariance constraint* $\psi(T\mathbf{x}) = T + \psi(\mathbf{x})$. A corresponding learning objective can be formulated as follows:

$$\min_{\psi} \frac{1}{n} \sum_{i=1}^{n} \|\psi(T_i\mathbf{x}_i) - \psi(\mathbf{x}_i) - T_i\|^2 \tag{1}$$

where $(\mathbf{x}_i, T_i)$ are example patches and transformations and the optimization is over the parameters of the regressor $\psi$ (e.g. the filter weights in a deep neural network).

### 2.2 Beyond corners

This section provides a first generalization of the construction above. While simple detectors such as Harris extract 2D points $\mathbf{f}$ in correspondence of corners, others such as SIFT extract circles in correspondence of blobs, and others again extract even more complex features such as oriented circles (e.g. SIFT with orientation assignment), ellipses (e.g. Harris-Affine), oriented ellipses (e.g. Harris-Affine with orientation assignment), etc. In general, due to their role in fixing image transformations, we will call the extracted shapes $\mathbf{f} \in \mathcal{F}$ *feature frames*.

The detector is thus a function $\psi : \mathcal{X} \to \mathcal{F},\ \mathbf{x} \mapsto \mathbf{f}$ mapping an image patch $\mathbf{x}$ to a corresponding feature frame $\mathbf{f}$. We say that the detector is *covariant with a group of*

[1] As the function $\psi$ needs to be location invariant it can be applied in a sliding window manner. Therefore $x$ can be a single patch which represents its perception field.

*transformations*[2] $g \in G$ (e.g. similarity or affine) when

$$\forall \mathbf{x} \in \mathcal{X}, g \in G : \quad \psi(g\mathbf{x}) = g\psi(\mathbf{x}) \tag{2}$$

where $g\mathbf{f}$ is the transformed frame and $g\mathbf{x}$ is the warped image.[3]

Working with feature frames is intuitive, but cumbersome and not very flexible. A much better approach is to drop frames altogether and replace them with corresponding transformations. For instance, in SIFT with orientation assignment all possible oriented circles $\mathbf{f}$ can be expressed uniquely as a similarity $g\mathbf{f}_0$ of a fixed oriented circle $\mathbf{f}_0$ (Fig. 2.left). Hence, instead of talking about oriented circles $\mathbf{f}$, we can equivalently talk about similarities $g$. Likewise, in the case of the Harris' corner detector, all possible 2D points $\mathbf{f}$ can be expressed as translations $T + \mathbf{f}_0$ of the origin $\mathbf{f}_0$, and so we can talk about translations $T$ instead of points $\mathbf{f}$.

To generalize this idea, we say that that a class of frames $\mathcal{F}$ *resolves* a group of transformations $G$ when, given a fixed *canonical frame* $\mathbf{f}_0 \in \mathcal{F}$, all frames are *uniquely generated* from it by the action of $G$:

$$\mathcal{F} = G\mathbf{f}_0 = \{g\mathbf{f}_0 : g \in G\} \quad \text{and} \quad \forall g, h \in G : \; g\mathbf{f}_0 = h\mathbf{f}_0 \;\Rightarrow\; g = h \text{ (uniqueness).}$$

This bijective correspondence allows to "rename" frames with transformations. Using this renaming, the detector $\psi$ can be rewritten as a function $\phi$ that outputs directly a transformation $\psi(\mathbf{x}) = \phi(\mathbf{x})\mathbf{f}_0$ instead of a frame.

With this substitution, the covariance constraint (2) becomes

$$\boxed{\phi(g\mathbf{x}) \circ \phi(\mathbf{x})^{-1} \circ g^{-1} = 1}. \tag{3}$$

Note that, for the group of translations $G = T(2)$, this constraint corresponds directly to the objective function (1). Fig. 2 provides two intuitive visualizations of this constraint.

It is also useful to extend the learning objective (1) as follows. As training data, we consider $n$ triplets $(\mathbf{x}_i, \hat{\mathbf{x}}_i, g_i), i = 1, \dots, n$ comprising an image (patch) $\mathbf{x}_i$, a transformation $g_i$, and the transformed and distorted image $\hat{\mathbf{x}}_i = g\mathbf{x}_i + \eta$. Here $\eta$ represents additive noise or some other useful distortion such as a random rescaling of the intensity which allows to train a more robust detector. The learning problem is then given by:

$$\min_{\phi} \frac{1}{n} \sum_{i=1}^{n} d(r_i, 1)^2, \qquad r_i = \phi(\hat{\mathbf{x}}_i) \circ \phi(\mathbf{x}_i)^{-1} \circ g_i^{-1} \tag{4}$$

where $d(r_i, 1)^2$ is the "distance" of the residual transformation $r_i$ from the identity.

### 2.3 General covariant feature extraction

The theory presented so far is insufficient to fully account for the properties of many common detectors. For this, we need to remove the assumptions that feature frames

[2] Here, a group of transformation $(G, \circ)$ is a set of functions $g, h : \mathbb{R}^2 \to \mathbb{R}^2$ together with composition $g \circ h \in G$ as group operation. Composition is associative; furthermore, $G$ contains the identity transformation 1 and the inverse $g^{-1}$ of each of its elements $g \in G$.

[3] The action $g\mathbf{x}$ of the transformation $g$ on the image $\mathbf{x}$ is to warp it: $(g\mathbf{x})(u, v) = \mathbf{x}(g^{-1}(u, v))$

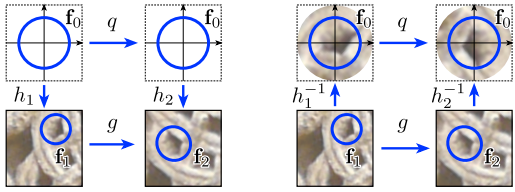



Fig. 3: **Left**: a (unoriented) circle identifies the translation and scale component of a similarity transformation $g \in G$, but leaves a residual rotation $q \in Q$ undetermined. Concretely, this could be the output of the SIFT detector prior orientation assignment. **Right:** normalization is achieved up to the residual transformation $q$.

resolve (i.e. fix) completely the group of transformations $G$. Most detectors are in fact *covariant with transformation groups larger than the ones that they can resolve*. For example, the Harris's detector is covariant with rotation and translation (in the sense that the same corners are extracted after the image is roto-translated), but, by detecting 2D points, it only resolves translations. Likewise, SIFT without orientation assignment is covariant to full similarity transformations but, by detecting circles, only resolves dilations (i.e. rotations remains undetermined; Fig. 3).

Next, we explain how eq. (3) must be modified to deal with detectors that (i) are covariant with a transformation group $G$ but (ii) resolve only a subgroup $H \subset G$. In this case, the detector function $\phi(\mathbf{x}) \in H$ returns a transformation in the smaller group $H$, and the covariance constraint (3) is satisfied up to a complementary transformation $q \in Q$ that makes up for the part not resolved by the detector:

$$\exists q \in Q : \quad \phi(g\mathbf{x}) \circ q \circ \phi(\mathbf{x})^{-1} \circ g^{-1} = 1. \tag{5}$$

This situation is illustrated graphically in Fig. 3.

For this construction to work, given $H \subset G$, the group $Q \subset G$ must be chosen appropriately. In eq. (5), and following Fig. 3, call $h_1 = \phi(\mathbf{x})$ and $h_2 = \phi(g\mathbf{x})$. Rearranging the terms, we get that $h_2 q = h_1 g$, where $h_2 \in H, q \in Q$ and $h_1 g \in G$. This means that any element in $G$ must be expressible as a composition $hq$, i.e. $G = HQ = \{hq : h \in H, q \in Q\}$. Formally (proofs in appendix):

**Proposition 1.** *If the group $G = HQ$ is the product of the subgroups $H$ and $Q$, then, for any choice of $g \in G$ and $h_1 \in H$, there is always a decomposition*

$$h_2 q h_1^{-1} g^{-1} = 1, \quad \textit{such that} \quad h_2 \in H,\ q \in Q. \tag{6}$$

In practice, given $G$ and $H$, $Q$ is usually easily found as the "missing transformation"; however, compared to (2), the transformation $q$ in constraint (5) is an extra degree of freedom that complicates optimization. Fortunately, in many cases the following proposition shows that there is only one possible $q$:

**Proposition 2.** *If $H \triangleleft G$ is* normal *in $G$ (i.e. $\forall g \in G, h \in H : g^{-1} h g \in H$) and $H \cap Q = \{1\}$, then, given $g \in G$, the choice of $q$ in the decomposition (5) is unique.*

The next section works through several concrete examples to illustrate these concepts.

### 2.4 A taxonomy of detectors

This section applies the theory developed above to standard detectors. Concretely, we limit ourselves to transformations up to affine, and write:

$$h_i = \begin{bmatrix} M_i & P_i \\ 0 & 1 \end{bmatrix}, \; q = \begin{bmatrix} L & 0 \\ 0 & 1 \end{bmatrix}, \; g = \begin{bmatrix} A & T \\ 0 & 1 \end{bmatrix}.$$

Here $P_i$ can be interpreted as the centre of the feature in image $\mathbf{x}_i$ and $M_i$ as its affine shape, $(A, T)$ as the parameters of the image transformation, and $L$ as the parameter of the complementary transformation not fixed by the detector. The covariance constraint (5) can be written, after a short calculation, as

$$M_2 L M_1^{-1} = A, \qquad P_2 - A P_1 = T. \tag{7}$$

As a first example, consider a basic corner detector that resolves translations $H = G = T(2)$ with no (non-trivial) complementary transformation $Q = \{1\}$. Hence $M_1 = M_2 = L = A = I$ and (5) becomes:

$$P_2 - P_1 = T. \tag{8}$$

This is the same expression found in the simple example of Sect. 2.1 and requires the detected features to have the correct relative shift $T$.

The Harris corner detector is similar, but is covariant with rotations too. Formally, $H = T(2) \subset G = SE(2)$ (Euclidean transforms) and $Q = SO(2)$ (rotations). Since $T(2) \triangleleft SE(2)$ is a normal subgroup, we expect to find a unique choice for $q$. In fact, it must be $M_i = I$, $A = L = R$, and the constraint reduces to:

$$P_2 - R P_1 = T. \tag{9}$$

In SIFT, $G = S(2)$ is the group of similarities, so that $A = sR$ is the composition of a rotation $R \in SO(2)$ and an isotropic scaling $s \in \mathbb{R}_+$. SIFT prior to orientation assignment resolves the subgroup $H$ of dilations (scaling and translation), so that $M_i = \sigma_i I$ (scaling) and the complement is a rotation $L \in SO(2)$. Once again $H \triangleleft G$, so the choice of $q$ is unique, and in particular $L = R$. The constraint reduces to:

$$P_2 - sRP_1 = T, \qquad \sigma_2/\sigma_1 = s. \tag{10}$$

When orientation assignment is added to SIFT, the similarities are completely resolved $H = G = S(2)$, $M_i = \sigma_i R_i$ is a rotation and scaling, and the constraint becomes:

$$P_2 - sRP_1 = T, \qquad \sigma_2/\sigma_1 = s, \qquad R_2 R_1^\top = R. \tag{11}$$

Affine detectors such as Harris-Affine (without orientation assignment) are more complex. In this case $G = A(2)$ are affinities and $H = UA(2)$ are *upright affinities*, i.e. affinities where the linear map $M_i \in LT_+(2)$ is a lower-triangular matrix with positive diagonal (these affinities, which still form a group, leave the "up" direction unchanged). The residual $Q = SO(2)$ are rotations and $HQ = G$ is still satisfied. However, $UA(2)$

is *not* normal in $A(2)$, Prop. 2 does not apply, and the choice of $Q$ is *not* unique.[4] The constraint has the form:

$$P_2 - AP_1 = T, \qquad M_2^{-1} A M_1 \in SO(2). \tag{12}$$

For affine detectors with orientation assignment, $H = G = A(2)$ and the constraint is:

$$P_2 - AP_1 = T, \qquad M_2 M_1^{-1} = A. \tag{13}$$

The generality of our formulation allows learning many new types of detectors. For example, by setting $H = T(2)$ and $G = A(2)$ it is possible to train a corner detector such as Harris which is covariant to full affine transformations. Furthermore, a benefit of working with transformations instead of feature frames is that we can train detectors that would be difficult to express in terms of geometric primitives. For instance, by setting $H = SO(2)$ and $G = SE(2)$, we can train a *orientation detector* which is covariant with *rotation and translation*. As for affine upright features, in this case $H$ is not normal in $G$ so the complementary translation $q = (I, T') \in Q$ is not uniquely fixed by $g = (R, T) \in G$; nevertheless, a short calculation shows that the only part of (5) that matters in this case is

$$R_2^\top R_1 = R \tag{14}$$

where $h_i = (R_i, 0)$ are the rotations estimated by the regressor.

## 3 Implementation

This section discusses several implementation details of our method: the parametrization of transformations, example CNN architectures, multiple features detection, efficient dense detection, and preparing the training data.

*Transformations: parametrization and loss.* Implementing (4) requires parametrizing the transformation $\phi(\mathbf{x}) \in H$ predicted by the regressor. In the most general case of interest here, $H = A(2)$ are affine transformations and the simplest approach is to output the corresponding matrix of coefficients:

$$\phi(\mathbf{x}) = \begin{bmatrix} \mathbf{a} & \mathbf{b} & \mathbf{p} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_u & b_u & p_u \\ a_v & b_v & p_v \\ 0 & 0 & 1 \end{bmatrix}.$$

Here $\mathbf{p}$ can be interpreted as the feature center and $\mathbf{a}$ and $\mathbf{b}$ as the feature affine shape. By rearranging the terms in (2), the loss function in (4) takes the form

$$d^2(r, 1) = \min_{q \in Q} \|g\phi(\mathbf{x}) - \phi(g\mathbf{x})q\|_F^2, \tag{15}$$

where $\|\cdot\|_F$ is the Frobenius norm. As seen before, the complementary transformation $q$ is often uniquely determined given $g$ and the minimization can be removed by substituting this fixed value for $q$. In practice, $g$ and $q$ are also represented by matrices, as described in Sect. 2.4.

[4] Concretely, from $M_2 L = A M_1$ the complement matrix $L$ is given by the QR decomposition of the r.h.s. which is a function of $M_1$, i.e. not unique.

Table 1: *Network architectures.* The DetNet-S and DetNet-L CNN architectures used which consist of a small number of convolutional layers applied densely and with no padding. The filter sizes and number is specified in the top part of each cell. Filters are followed by ReLU layers and, where indicated, by $2 \times 2$ max pooling and/or LRN.

| Model | Conv1 | Conv2 | Conv3 | Conv4 | Conv5 | Conv6 | Conv7 |
|---|---|---|---|---|---|---|---|
| DetNet-S | $5 \times 5 \times 40$ Pool $\downarrow 2$ | $5 \times 5 \times 100$ Pool $\downarrow 2$ | $4 \times 4 \times 300$ | $1 \times 1 \times 500$ | $1 \times 1 \times 500$ | $1 \times 1 \times 2$ | |
| DetNet-L | $5 \times 5 \times 60$ Pool $\downarrow 2$ | $5 \times 5 \times 150$ Pool $\downarrow 2$ + LRN | $4 \times 4 \times 450$ | $1 \times 1 \times 600$ | $1 \times 1 \times 600$ | $1 \times 1 \times 600$ | $1 \times 1 \times 2$ |

When the resolved transformations $H$ are less general than affinities, the parametrization can be adjusted accordingly. For instance, for the basic detector of Sect. 2.1, where $H = T(2)$, on can fix $\mathbf{a} = (1, 0)$, $\mathbf{b} = (0, 1)$, $q = I$ and $g = (I, T)$, which reduces to eq. (1). If, on the other hand, $H = SO(2)$ are rotation matrices as for the orientation detector (14),

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{a_u^2 + a_v^2}} \begin{bmatrix} a_u & -a_v & 0 \\ a_v & a_u & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{16}$$

*Network architectures.* One of the benefits of our approach is that it allows to use deep neural networks in order to implement the feature regressor $\phi(\mathbf{x})$. Here we experiment with two such architectures, DetNet-S and DetNet-L, summarized in Tab. 1. For fast detection, these resemble the compact LeNet model of [15]. The main difference between the two is the number of layers and filters. The loss (15) is differentiable and easily implemented in a network loss layer. Note that the loss requires evaluating the network $\phi$ twice, once applied to image $\mathbf{x}$ and once to image $g\mathbf{x}$. Like in siamese architectures, these can be thought of as two networks with shared weights.

When implemented in a standard CNN toolbox (in our case in MatConvNet [43]), multiple patch pairs are processed in parallel by a single CNN execution in what is known as a minibatch. In practice, the operations in (15) can be implemented using off-the-shelf CNN components. For example, the multiplication by the affine transformation $g$ in (15), which depends on which pair of images in the batch is considered, can be implemented by using convolution routines, $1 \times 1$ filters, and so called "filter groups".

*From local regression to global detection.* The formulation (4) learns a function $\psi$ that maps an image patch $\mathbf{x}$ to a single detected feature $\mathbf{f} = \psi(\mathbf{x})$. In order to detect multiple features in a larger image, the function $\psi$ is simply applied convolutionally at all image locations (Fig. 1). Then, due to covariance, partially overlapping patches $\mathbf{x}$ that contain the same feature are mapped by $\psi$ to the same detection $\mathbf{f}$. Such duplicate detections are collapsed and their number, which reflects the stability of the feature, is used as detection confidence.

For point features ($G = T(2)$), this voting process is implemented efficiently by accumulating votes in a map containing one bin for each pixel in the input image. Votes are accumulated using bilinear interpolation, after which non-maxima suppression is applied with a radius of two pixels. This scheme can be easily extended to more complex

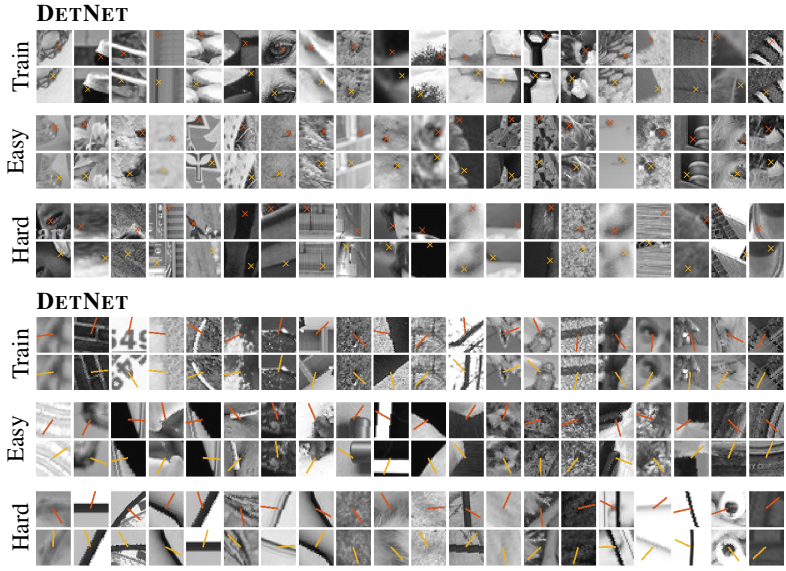



Fig. 4: *Training and validation patches.* Example of training triplets $(\mathbf{x}_1, \mathbf{x}_2, g)$ ($\mathbf{x}_1$ above and $\mathbf{x}_2 = g\mathbf{x}_1$ below) for different detectors. The figure also shows "easy" and "hard" patch pairs, extracted from the validation set based on the value of the loss (16). The crosses and bars represent respectively the detected translation and orientation, as learned by DETNET-L and ROTNET-L.

features, particularly under the reasonable assumption that only one feature is detected at each image location.

Note that some patches may in practice contain two or more clearly visible feature anchors. The detector $\psi$ must then decide which one to select. This is not a significant limitation at test time (as the missed anchors would likely be selected by a translated application of $\psi$). Its effect at training time is discussed later.

*Efficient dense evaluation.* As most CNNs, architectures DetNet-S and DetNet-L rapidly downsample their input for efficiency. In order to perform dense feature detection, the easiest approach is to reapply the CNNs to slightly shifted versions of the image, filling the "holes" left in the downsampled output. An equivalent but much more efficient method, which reuses significant computations in the denser early layers of the network, is the *à trous* algorithm [21,26].

We propose here an algorithm equivalent to *à trous* which is just as efficient and more easily implemented. Given a CNN layer $\mathbf{x}_l = \phi_l(\mathbf{x}_{l-1})$ that downsamples the input tensor $\mathbf{x}_{l-1}$ by a factor of, say, two, the downsampling factor is changed to one, and the now larger output $\mathbf{x}_l$ is split into four parts $\mathbf{x}_l^{(k)}, k = 1, \ldots, 4$. Each part is obtained by downsampling $\mathbf{x}_l$ after shifting it by zero or one pixels in the horizontal and vertical directions (for a total of four combinations). Then the deeper layers of the networks are computed as usual on the four parts independently. The construction is repeated whenever downsampling needs to be performed.

Detection speed can be improved with evaluating the regressor with stride 2 (at every second pixel). We refer to these detector as DETNET*S2*. Source code and the DETNETmodels are freely available[5].

*Training data.* Training images are obtained from the ImageNet ILSVRC 2012 training data [33], extracting twenty random $57 \times 57$ crops per image, for up to $6M$ crops. Uniform crops are discarded since they clearly cannot contain any useful anchor. To do so, the absolute response of a LoG filter of variance $\sigma = 2.5$ is averaged and the crop is retained if the response is greater than 1.5 (image intensities are in the range $[0, 255]$). Note that, combined with random selection, this operation *does not* center crops on blobs or any other pre-defined anchors, but simply discards uniform or very low contrast crops.

Recall that the formulation Sect. 2.2 requires triplets $(\mathbf{x}_1, \mathbf{x}_2, g)$. A triplet is generated by randomly picking a crop and then by extracting $28 \times 28$ patches $\mathbf{x}_1$ and $\mathbf{x}_2$ within 20 pixels of the crop center (Fig. 4). This samples two patches related by translation, corresponding to the translation sampled in $g$, while guaranteeing that patches overlap by least $27\%$. Then the linear part of $g$ is sampled at random and used to warp $\mathbf{x}_2$ around its center. In order too achieve better robustness to photometric transformations, additive ($\pm 8\%$ of the intensity range) and multiplicative ($\pm 40\%$ of a pixel intensity) is added to the pixels .

Training uses batches of 64 patch pairs. An epoch contains $40 \cdot 10^3$ pairs, and the data is resampled after each epoch completes. The learning rate is set to $\lambda = 0.01$ and decreased tenfold when the validation error stops decreasing. Usually, training converges after 60 epochs, which, due to the small size of the network and input patches, takes no more than a couple of minutes on a GPU.

## 4 Experiments

We apply our framework to learn two complementary types of detectors in order to illustrate the flexibility of the approach: a corner detector (Sect. 4.1) and an orientation detector (Sect. 4.2).

*Evaluation benchmark and metrics.* We compare the learned detectors to standard ones: FAST [31,30] (using OpenCV's implementation[6]), the Difference of Gaussian detector (DoG) or SIFT [20], the Harris corner point detector [11] and Hessian point detector [24] (all using VLFeat's implementation[7]). All experiments are performed at a single scale, but all detectors can be applied to a scale space pyramid if needed.

For evaluation of the corner detector, we use the standard VGG-Affine benchmark dataset [24], using both the *repeatability* and *matching score* criteria. For matching score, SIFT descriptors are extracted from a fixed region of $41 \times 41$ pixels around each corner. A second limitation in the original protocol of [24] is that repeatability can be made arbitrarily large simply by detecting enough features. Thus, in order to control

[5] https://github.com/lenck/ddet
[6] opencv.org
[7] www.vlfeat.org

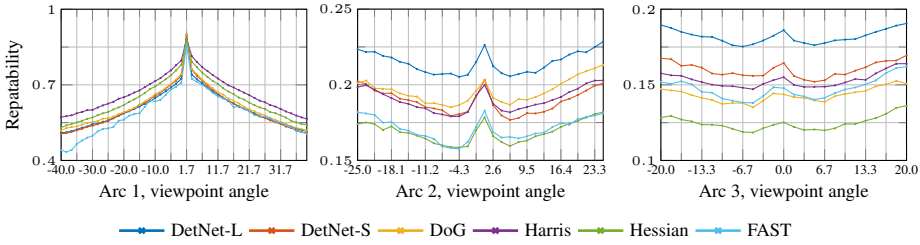



Fig. 5: *Repeatability on the DTU Dataset* averaged over all 60 scenes, divided by arc. Repeatability is computed over the top 600 detections for each detector.

for the number of features detected, we compute repeatability and matching score as the feature detection threshold is increased; we then plot the metrics as functions of the number of feature selected in the first image.

VGG-Affine contains scenes related by homography. We also consider the more recent DTU-Robots dataset [1] that contains 3D objects under changing viewpoint. Matches in DTU dataset are estimated using the known 3D shape of the objects and position of the camera. The data is divided in three "arcs", corresponding to three swipes of the robotic camera at different distances from the scene (0.5, 0.65, and 0.8m respectively). Due to the large number of images in this dataset, only aggregated results for $n = 600$ are reported.

### 4.1 Corner or translation detector

In this section we train a "corner detector" network DETNET. Using the formalism of Sect. 2, this is a detector which is covariant with translations $G = T(2)$, corresponding to the covariance constraint of eq. (1). Fig. 4 provides a few examples of the patches used for training, as well as of the anchors discovered by learning.

Fig. 5 reports the performance of the two versions of DETNET, small and large, on the DTU data. As noted in [1], the Harris corner detector performs very well on the first arc; however, on the other two arcs DETNET-L clearly outperforms the other methods, whereas DETNET-S is on par or a little better than standard detectors.

Fig. 6 evaluates the method on the VGG-Affine dataset. Here the learned networks perform generally well too, outperforming existing detectors in some scenarios and performing less well on others. Note that our current implementation is the simplest possible and the very first of its kind; in the future more refined setups may improve the learned detectors across the board (Sect. 5).

The speed of the tested detectors is shown in Table 2. While our goal is not to obtain the fastest detector but rather to demonstrate the possibility of learning detectors from scratch, we note that even an unoptimised MATLAB implementation can achieve reasonable performance on a GPU, especially with stride 2 with a slightly decreased performance compared to the dense evaluation (see Figure 6).

Table 2: The detection speed (in FPS) for different image sizes of all tested detectors, computed as an average over 10 measurements. Please not that the DETNETdetectors run on a GPU, other detectors run on a CPU.

| | DETNET-L | DETNET-L S2 | DETNET-S | DETNET-S S2 | Harris | DoG | Hessian | FAST |
|---|---|---|---|---|---|---|---|---|
| 320 × 240 | 9.16 | 33.14 | 27.26 | 83.16 | 144.39 | 88.64 | 150.34 | 439.68 |
| 800 × 600 | 1.45 | 5.87 | 4.68 | 19.32 | 15.65 | 8.00 | 17.45 | 328.20 |
| 1024 × 768 | 0.39 | 1.56 | 2.78 | 11.68 | 12.21 | 6.05 | 11.17 | 206.96 |

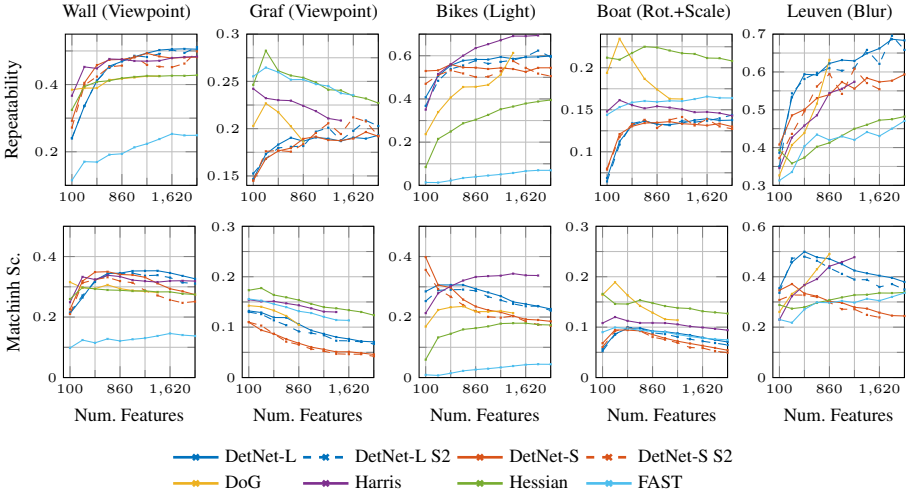
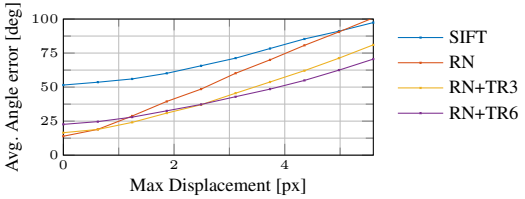



Fig. 6: *Repeatability and matching score on VGG dataset* comparing two versions of DetNet and standard detectors controlled for an increasing number of detected features. Dashed line values are for DetNet with stride 2. Scores are computed as an average over all 5 transformed images for each set (e.g. "wall").

### 4.2 Orientation detector

This section evaluates a network, ROTNET, trained for orientation detection. This detector resolves $H = SO(2)$ rotations and is covariant to Euclidean transformations $G = SE(2)$, which means that translations $Q = T(2)$ are nuisance factor that the detector should ignore. The corresponding form of the covariance constraint is given by eq. (14). Training proceeds as above, using $28 \times 28$ pixels patches and, for $g$, random $2\pi$ rotations composed with a maximum nuisance translation of 0, 3, or 6 pixels, resulting in three different versions of the network (Fig. 4).

The SIFT detector [20] contains both a blob detector as well as an orientation detector, based on determining the dominant gradient orientation in the patch. Fig. 7 compares the average angular registration error obtained by the SIFT orientation detector and different versions of ROTNET, measured from pairs of randomly-sampled image patches. We note that: 1) ROTNETis sensibly better than the SIFT orientation detector, with up to half the error rate, and that 2) while the error increases with the maximum




Matching score (%)

| | *graf* | *boat* | *bark* | *bikes* | *leuven* |
|---|---|---|---|---|---|
| **SIFT** | 68.0 | 73.7 | 75.3 | 81.8 | 85.2 |
| **RotNet-L** | 67.3 | 74.3 | 81.1 | 83.0 | 85.2 |

Fig. 7: *Orientation detector evaluation.* **Left**: versions of RotNet(RN) and the SIFT orientation detector evaluated on recovering the relative rotation of random patch pairs. **Right**: matching score on the VGG-Affine benchmark when the native SIFT orientation estimation is replaced with RotNet(percentage of correct matches using the DoG-Affine detector).

nuisance translation between patches, networks that are trained to account for such translations are sensibly better than the ones that do not. Furthermore, when applied to the output of the SIFT blob detector, the improved orientation estimation results in an improved feature matching score, as measured on the VGG-Affine benchmark.

## 5 Discussion

We have presented the first general machine learning formulation for covariant feature detectors. The latter is supported by a comprehensive theory of covariant detectors, and builds on the idea of casting detection as a regression problem. We have shown that this method can successfully learn corner and orientation detectors that outperform in several cases off-the-shelf detectors. The potential is significant; for example, the framework can be used to learn scale selection and affine adaptation CNNs. Furthermore, many significant improvements to our basic implementation are possible, including explicitly modelling detection strength/confidence, predicting multiple features in a patch, and jointly training detectors and descriptors.

***Acknowledgements*** We would like to thank ERC 677195-IDIU for supporting this research.

## A Proofs

*Proof (of Proposition 1).* Due to group closure, $gh_1 \in G$. Since $HQ = G$, then there must be $h_2 \in H, q \in Q$ such that $h_2 q = g h_1$, and so $h_2 q h_1^{-1} g^{-1} = 1$.

*Proof (of Proposition 2).* Let $h_2 q (h_1)^{-1} = h_2' q' (h_1')^{-1}$ be two such decompositions and multiply to the left by $(q)^{-1}(h_2')^{-1}$ and to the right by $h_1'$:

$$\underbrace{q^{-1}\,[(h_2')^{-1} h_2]\,q}_{\in H \text{ (due to normality)}}\ \underbrace{h_1^{-1} h_1'}_{\in H} = \underbrace{q^{-1} q'}_{\in Q}.$$

Since this quantity is simultaneously in $H$ and in $Q$, it must be in the intersection $H \cap Q$, which by hypothesis contains only the identity. Hence $q^{-1}q' = 1$ and $q = q'$.

## References


1. H. Aanæs, A. Dahl, and K. Steenstrup Pedersen. Interesting interest points. *International Journal of Computer Vision*, pages 18–35, 2012. 12
2. A. M. Baumberg. Reliable feature matching across widely separated views. In *Proc. CVPR*, pages 774–781, 2000. 2, 3
3. P. R. Beaudet. Rotationally invariant image operators. In *International Joint Conference on Pattern Recognition*, volume 579, page 583, 1978. 3
4. K. Cordes, B. Rosenhahn, and J. Ostermann. Increasing the accuracy of feature evaluation benchmarks using differential evolution. In *IEEE Symposium on Differential Evolution*, 2011. 2
5. P. Dias, A. Kassim, and V. Srinivasan. A neural network based corner detection method. In *IEEE Int. Conf. on Neural Networks*, 1995. 3
6. Y. Dufournaud, C. Schmid, and R. Horaud. Matching images with different resolutions. In *Proc. CVPR*, 1999. 2
7. W. Förstner. A feature based correspondence algorithm for image matching. *International Archives of Photogrammetry and Remote Sensing*, 26(3):150–166, 1986. 3
8. H. Freeman and L. S. Davis. A corner-finding algorithm for chain-coded curves. *IEEE Transactions on Computers*, (3):297–303, 1977. 3
9. A. Guiducci. Corner characterization by differential geometry techniques. *Pattern Recognition Letters*, 8(5):311–318, 1988. 3
10. X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proc. CVPR*, 2015. 1
11. C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. of The Fourth Alvey Vision Conference*, pages 147–151, 1988. 2, 3, 11
12. S. Holzer, J. Shotton, and P. Kohli. Learning to efficiently detect repeatable interest points in depth data. In *Proc. ECCV*, 2012. 3
13. T. Kadir and M. Brady. Saliency, scale and image description. *Int. J. Computer Vision*, 45:83–105, 2001. 3
14. W. Kienzle, F. A. Wichmann, B. Schölkopf, and M. O. Franz. Learning an interest operator from human eye movements. In *CVPR Workshop*, 2006. 3
15. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Nov 1998. 9
16. T. Lindeberg. *Scale-Space Theory in Computer Vision*. Springer, 1994. 2
17. T. Lindeberg. Feature detection with automatic scale selection. *IJCV*, 30(2):77–116, 1998. 2
18. T. Lindeberg and J. Garding. Shape-adapted smoothing in estimation of 3-D depth cues from affine distortions of local 2-D brightness structure. In *Proc. ECCV*, 1994. 2
19. D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV*, 1999. 2, 3
20. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2(60):91–110, 2004. 11, 13
21. S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 2008. 10
22. J. Matas, S. Obdrzálek, and O. Chum. Local affine frames for wide-baseline stereo. In *Intl. Conference on Pattern Recognition*, 2002. 2
23. K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proc. ICCV*, 2001. 2, 3
24. K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. ECCV*, pages 128–142. Springer-Verlag, 2002. 2, 3, 11
25. G. Olague and L. Trujillo. Evolutionary-computer-assisted design of image operators that detect interest points using genetic programming. *Image and Vision Computing*, 2011. 3

26. G. Papandreou, I. Kokkinos, and P.-A. Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. *Proc. CVPR*, 2015. 10
27. M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronin, and C. Schmid. Local convolutional features with unsupervised training for image retrieval. In *ICCV*, 2015. 3
28. K. Rohr. Recognizing corners by fitting parametric models. *IJCV*, 9(3), 1992. 3
29. A. Rosenfeld and E. Johnston. Angle detection on digital curves. *Computers, IEEE Transactions on*, 100(9):875–878, 1973. 3
30. E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *ICCV*, volume 2, 2005. 11
31. E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. ECCV*, 2006. 3, 11
32. E. Rosten, R. Porter, and T. Drummond. Faster and better: a machine learning approach to corner detection. In *PAMI*, volume 32, 2010. 3
33. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge, 2014. 11
34. P. Sankar and C. Sharma. A parallel procedure for the detection of dominant points on a digital curve. *Computer Graphics and Image Processing*, 7(3):403–412, 1978. 3
35. F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *Proc. ICCV*, 2001. 2
36. C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1997. 2
37. E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 3
38. S. M. Smith and J. M. Brady. Susan – a new approach to low level image processing. Technical report, Oxford University, 1995. 3
39. J. Sochman and J. Matas. Learning fast emulators of binary decision processes. *IJCV*, 2009. 3
40. B. Triggs. Detecting keypoints with stable position, orientation, and scale under illumination changes. In *Proc. ECCV*, 2004. 3
41. L. Trujillo and G. Olague. Synthesis of interest point detectors through genetic programming. In *Proc. GECCO*, 2006. 3
42. T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *Proc. BMVC*, pages 412–425, 2000. 2
43. A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proc. ACM Int. Conf. on Multimedia*, 2015. 9
44. K. M. Yi, Y. Verdie, P. Fua, and V. Lepetit. Learning to Assign Orientations to Feature Points. In *CVPR*, 2016. 3
45. S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015. 3
46. J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proc. CVPR*, 2015. 1
47. M. Zuliani, C. Kenney, and B. S. Manjunath. A mathematical comparison of point detectors. In *Proc. CVPR*, 2005. 3