Resource-Efficient Invariant Networks: Exponential Gains by Unrolled Optimization

Sam Buchanan*†‡ Jingkai Yan†‡ Ellie Haber§ John Wright†‡¶

March 11, 2022

Abstract

Achieving invariance to nuisance transformations is a fundamental challenge in the construction of robust and reliable vision systems. Existing approaches to invariance scale exponentially with the dimension of the family of transformations, making them unable to cope with natural variabilities in visual data such as changes in pose and perspective. We identify a common limitation of these approaches—they rely on sampling to traverse the high-dimensional space of transformations—and propose a new computational primitive for building invariant networks based instead on optimization, which in many scenarios provides a provably more efficient method for high-dimensional exploration than sampling. We provide empirical and theoretical corroboration of the efficiency gains and soundness of our proposed method, and demonstrate its utility in constructing an efficient invariant network for a simple hierarchical object detection task when combined with unrolled optimization. Code for our networks and experiments is available at https://github.com/sdbuch/refine.

1 Introduction

In computing with any kind of realistic visual data, one must contend with a dizzying array of complex variabilities: statistical variations due to appearance and shape, geometric variations due to pose and perspective, photometric variations due to illumination and cast shadows, and more. Practical systems cope with these variations by a data-driven approach, with deep neural network architectures trained on massive datasets. This approach is especially successful at coping with variations in texture and appearance. For invariance to geometric transformations of the input (e.g., translations, rotations, and scaling, as in Figure 1(a-d)), the predominant approach in practice is also data-driven: the 'standard pipeline' is to deploy an architecture that is structurally invariant to translations (say, by virtue of convolution and pooling), and improve its stability with respect to other types of transformations by data augmentation. Data augmentation generates large numbers of synthetic training samples by applying various transformations to the available training data, and demonstrably contributes to the performance of state-of-the-art systems [CZMV+19; CKNH20; HMCZ+20]. However, it runs into a basic resource efficiency barrier associated with the dimensionality of the set of nuisances: *learning over a d-dimensional group of transformations requires both data and architectural resources that are* exponential *in d* [BJ17; CJLZ19; Sch19; NI20; CK21]. This is a major obstacle to achieving invariance to large, structured deformations such as 3D rigid body motion ($d = 6$), homography ($d = 8$), and linked rigid body motion [KZFM19] and even nonrigid deformations [ZMH15] ($d \gg 8$). It is no surprise, then, that systems trained in this fashion remain vulnerable to adversarial transformations of domain [FF15; KMF18; XZLH+18; AAG19; ALGW+19; AW19; ETTS+19]—it simply is not possible to generate enough artificial training data to learn transformation manifolds of even moderate dimension. Moreover, this approach is fundamentally wasteful: learning nuisances known to be present in

*Corresponding author: s.buchanan@columbia.edu
†Department of Electrical Engineering, Columbia University
‡Data Science Institute, Columbia University
§NYU Tandon School of Engineering
¶Department of Applied Physics and Applied Mathematics, Columbia University

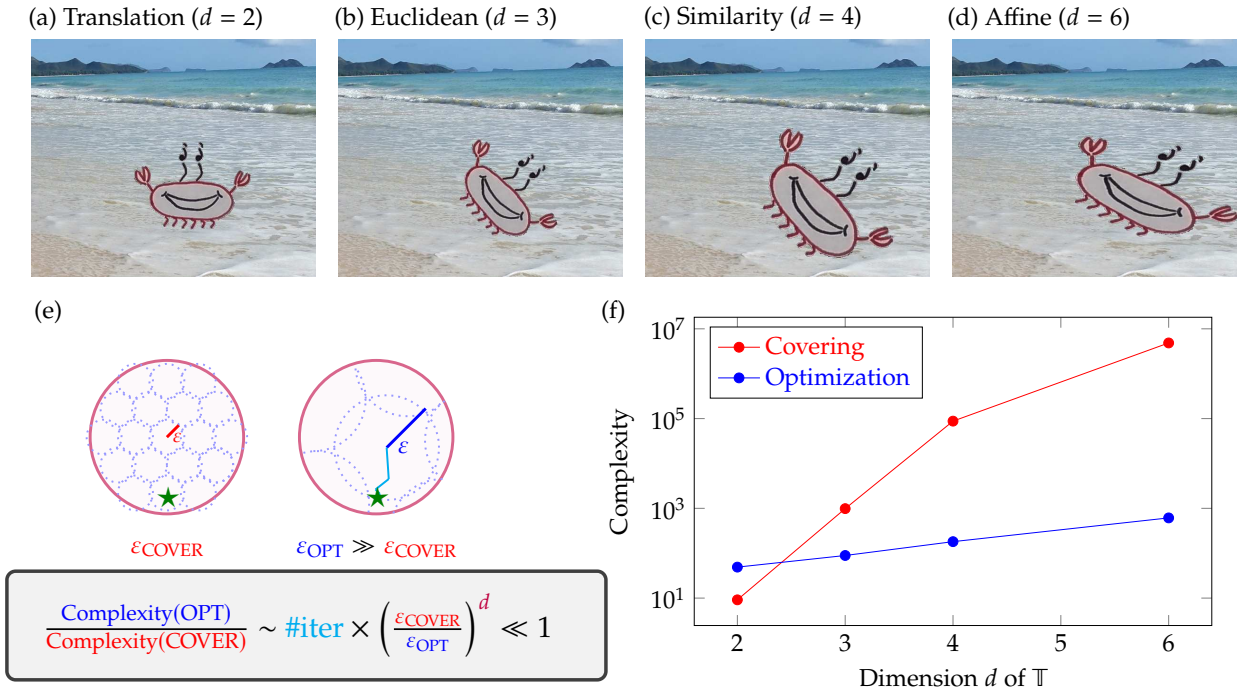

Figure 1: Comparing the complexity of covering-based and optimization-based methods for invariant recognition of a template embedded in visual clutter. **(a-d)**: We consider four different classes of deformations that generate the observation of the template, ranging across shifts, rotations, scale, and skew. The dimension d of the family of transformations increases from left to right. **(e)**: A geometric illustration of the covering and optimization approaches to global invariance: in certifying that a query (labeled with a star) is a transformed instance of the template (at the base point of the solid red/blue lines), optimization can be vastly more efficient than covering, because it effectively covers the space at the *scale of the basin of attraction* of the optimization problem, which is always larger than the template's associated $\varepsilon_{\text{COVER}}$. **(f)**: Plotting the average number of convolution-like operations necessary to reach a zero-normalized cross-correlation (ZNCC) of 0.9 between the template and a randomly-transformed query across the different deformation classes. Optimization leads to an efficiency gain of several orders of magnitude as the dimensionality of the family of transformations grows. Precise experimental details are recorded in Appendix A.3.

the input data wastes architectural capacity that would be better spent coping with statistical variability in the input, or learning to perform complex tasks.

These limitations of the standard pipeline are well-established, and they have inspired a range of alternative *architectural* approaches to achieving invariance, where each layer of the network incorporates computational operations that reflect the variabilities present in the data. Nevertheless, as we will survey in detail in Section 2, all known approaches are subject to some form of exponential complexity barrier: the computational primitives demand either a filter count that grows as $\exp(d)$ or integration over a d-dimensional space, again incurring complexity exponential in d. Like data augmentation, these approaches can be seen as obtaining invariance by exhaustively sampling transformations from the d-dimensional space of nuisances, which seems fundamentally inefficient: in many concrete high-dimensional signal recovery problems, *optimization* provides a significant advantage over naive grid searching when exploring a high-dimensional space [SER17; GBW19; CDHS21], as in Figure 1(e). This motivates us to ask:

Can we break the barrier between resource-efficiency and invariance using *optimization* as the architectural primitive, rather than sampling?

In Figure 1, we conduct a simple experiment that suggests a promising avenue to answer this question in the affirmative. Given a known synthetic textured motif subject to an unknown structured transformation and embedded in a background, we calculate the number of computations (convolutions and interpolations) required to certify with high confidence that the motif appears in the image. Our baseline approach is

template matching, which enumerates as many transformations of the input as are necessary to certify the motif's occurrence (analogous to existing architectural approaches with sampling/integration as the computational primitive)—each enumeration requires one interpolation and one convolution. We compare to a gradient-based optimization approach that attempts to match the appearance of the test image to the motif, which uses three interpolations and several convolutions per iteration (and on the order of 10^2 iterations). As the dimensionality of the space of transformations grows, the optimization-based approach demonstrates an increasingly-significant efficiency advantage over brute-force enumeration of templates—at affine transformations, for which $d = 6$, it becomes challenging to even obtain a suitable transformation of the template by sampling.

To build from the promising optimization-based approach to local invariance in this experiment to a full invariant neural network architecture capable of computing with realistic visual data, one needs a general method to incorporate prior information about the specific visual data, observable transformations, and target task into the design of the network. We take the first steps towards realizing this goal: inspired by classical methods for image registration in the computer vision literature, we propose an optimization formulation for seeking a structured transformation of an input image that matches previously-observed images, and we show how combining this formulation with *unrolled optimization* [GL10; OJMB+20; CCCH+21], which converts an iterative solver for an optimization problem into a neural network, implies resource-efficient and principled invariant neural architectural primitives. In addition to providing network architectures incorporating 'invariance by design', this is a principled approach that leads to networks amenable to theoretical analysis, and in particular we provide convergence guarantees for specific instances of our optimization formulations that transfer to the corresponding unrolled networks. On the practical side, we illustrate how these architectural primitives can be combined into a task-specific neural network by designing and evaluating an invariant network architecture for an idealized single-template hierarchical object detection task, and present an experimental corroboration of the soundness of the formulation for invariant visual motif recognition used in the experiment in Figure 1. Taken altogether, these results demonstrate a promising new direction to obtain theoretically-principled, resource-efficient neural networks that achieve guaranteed invariance to structured deformations of image data.

The remainder of the paper is organized as follows: Section 2 surveys the broad range of architectural approaches to invariance that have appeared in the literature; Section 3 describes our proposed optimization formulations and the unrolling approach to network design; Section 4 describes the hierarchical invariant object detection task and a corresponding invariant network architecture; Section 5 establishes convergence guarantees for our optimization approach under a data model inspired by the hierarchical invariant object detection task; and Section 6 provides a more detailed look at the invariance capabilities of the formulation used in Figure 1.

2 Related Work

Augmentation-based invariance approaches in deep learning. The 'standard pipeline' to achieving invariance in deep learning described in Section 1 occupies, in a certain sense, a minimal point on the tradeoff curve between a purely data-driven approach and incorporating prior knowledge about the data into the architecture: by using a convolutional neural network with pooling, invariance to translations of the input image (a two-dimensional group of nuisances) is (in principle) conferred, and invariance to more complex transformations is left up to a combination of learning from large datasets and data augmentation. A number of architectural proposals in the literature build from a similar perspective, but occupy different points on this tradeoff curve. Parallel channel networks [CMS12] generate all possible transformations of the input and process them in parallel, and have been applied for invariance to rotations [DWD15; LSBP16] and scale [JL21]. Other architectures confer invariance by pooling over transformations at the feature level [SL12], similarly for rotation [WGTB17] and scale [KSJ14]. Evidently these approaches become impracticable for moderate-dimensional families of transformations, as they suffer from the same sampling-based bottleneck as the standard pipeline.

To avoid explicitly covering the space of transformations, one can instead incorporate learned deformation offsets into the network, as in deformable CNNs [DQXL+17] and spatial transformer networks [JSZK15]. At a further level of generality, capsule networks [HKW11; SFH17; Hin21; STDS+21] allow more

flexible deformations among distinct parts of an object to be modeled. The improved empirical performance observed with these architectures in certain tasks illustrates the value of explicitly modeling deformations in the network architecture. At the same time, when it comes to guaranteed invariance to specific families of structured deformations, they suffer from the same exponential inefficiencies as the aforementioned approaches.

Invariance-by-construction architectures in deep learning. The fundamental efficiency bottleneck encountered by the preceding approaches has motivated the development of alternate networks that are invariant simply by virtue of their constituent computational building blocks. Scattering networks [BM13] are an especially principled and elegant approach: they repeatedly iterate layers that convolve an input signal with wavelet filters, take the modulus, and pool spatially. These networks provably obtain translation invariance in the limit of large depth, with feature representations that are Lipschitz-stable to general deformations [Mal12]; moreover, the construction and provable invariance/stability guarantees generalize to feature extractors beyond wavelet scattering networks [WB18]. Nevertheless, these networks suffer from a similar exponential resource inefficiency to those that plague the augmentation-based approaches: each layer takes a wavelet transform of *every* feature map at the previous layer, resulting in a network of width growing exponentially with depth. Numerous mitigation strategies have been proposed for this limitation [BM13; ZTAM20; ZGM21], and combinations of relatively-shallow scattering networks with standard learning machines have demonstrated competitive empirical performance on certain benchmark datasets [OBZ17]. However, the resulting hybrid networks still suffer from an inability to handle large, structured transformations of domain such as pose and perspective changes.

Group scattering networks attempt to remedy this deficiency by replacing the spatial convolution operation with a group convolution $w, x \mapsto [w * x](g) = \int_{\mathbb{G}} x(g')w(g^{-1}g')\,\mathrm{d}\mu(g')$ [Mal12; CW16; KT18; BBCV21]. In this formula, $\mathbb{G}$ is a group with sufficient topological structure, μ is Haar measure on $\mathbb{G}$, and w and x are the filter and signal (resp.), defined on $\mathbb{G}$ (or a homogeneous space for $\mathbb{G}$, as in spherical CNNs [CGKW18]). Spatial convolution of natural images coincides with the special case $\mathbb{G} = \mathbb{Z}^2$ in this construction; for more general groups such as 3D rotation, networks constructed by iterated group convolutions yield feature representations equivariant to the group action, and intermixing pooling operations yields invariance, just as with 2D convolutional neural networks. At a conceptual level, this basic construction implies invariant network architectures for an extremely broad class of groups and spaces admitting group actions [WFVW21], and has been especially successful in graph-structured tasks such as molecular prediction where there is an advantage to enforcing symmetries [BBCV21]. However, its application to visual data has been hindered by exponential inefficiencies in computing the group convolution—integration over a d-dimensional group $\mathbb{G}$ costs resources exponential in d—and more fundamentally by the fact that discrete images are defined on the image plane $\mathbb{Z}^2$, whereas group convolutions require the signal to be defined over the group $\mathbb{G}$ one seeks invariance to. In this sense, the 'reflexivity' of spatial convolution and discrete images seems to be the exception rather than the rule, and there remains a need for resource-efficient architectural primitives for invariance with visual data.

"Unrolling" iterative optimization algorithms. First introduced by Gregor and LeCun in the context of sparse coding [GL10], unrolled optimization provides a general method to convert an iterative algorithm for solving an optimization problem into a neural network (we will provide a concrete demonstration in the present context in Section 3), offering the possibility to combine the statistical learning capabilities of modern neural networks with very specific prior information about the problem at hand [CCCH+21]. It has found broad use in scientific imaging and engineering applications [KKHP17; BHKW18; KBCW19; KBRW19; OJMB+20], and most state-of-the-art methods for learned MRI reconstruction are based on this approach [MRRK+21]. In many cases, the resulting networks are amenable to theoretical analysis [CLWY18; LCWY19], leading to a mathematically-principled neural network construction.

3 Invariant Architecture Primitives: Optimization and Unrolling

Notation. We write $\mathbb{R}$ for the reals, $\mathbb{Z}$ for the integers, and $\mathbb{N}$ for the positive integers. For positive integers m, n, and c, we let $\mathbb{R}^m$, $\mathbb{R}^{m\times n}$, and $\mathbb{R}^{m\times n\times c}$ denote the spaces of real-valued m-dimensional vectors, m-by-n

matrices, and c-channel m-by-n images (resp.). We write $\boldsymbol{e}_i$, $\boldsymbol{e}_{ij}$, etc. to denote the elements of the canonical basis of these spaces, and $\mathbf{1}_m$ and $\mathbf{0}_{m,n}$ (etc.) to denote their all-ones and all-zeros elements (resp.). We write $\langle \cdot, \cdot \rangle$ and $\| \cdot \|_F$ to denote the euclidean inner product and associated norm of these spaces. We identify m by n images $\boldsymbol{x}$ with functions on the integer grid $\{0, 1, \ldots, m-1\} \times \{0, 1, \ldots, n-1\}$, and therefore index images starting from 0; when applying operations such as filtering, we will assume that an implementation takes the necessary zero padding, shifting, and truncation steps to avoid boundary effects. For a subset $\Omega \subset \mathbb{Z}^2$, we write $\mathcal{P}_\Omega$ for the orthogonal projection onto the space of images with support Ω.

Given a deformation vector field $\boldsymbol{\tau} \in \mathbb{R}^{m' \times n' \times 2}$ and an image $\boldsymbol{x} \in \mathbb{R}^{m \times n \times c}$, we define the transformed image $\boldsymbol{x} \circ \boldsymbol{\tau}$ by $(\boldsymbol{x} \circ \boldsymbol{\tau})_{ij} = \sum_{(k,l) \in \mathbb{Z}^2} \boldsymbol{x}_{kl} \phi(\tau_{ij0} - k) \phi(\tau_{ij1} - l)$, where $\phi : \mathbb{R} \to \mathbb{R}$ is the cubic convolution interpolation kernel [Key81].[1] For parametric transformations of the image plane, we write $\boldsymbol{\tau}_{\boldsymbol{A},\boldsymbol{b}}$ to denote the vector field representation of the transformation parameterized by $(\boldsymbol{A}, \boldsymbol{b})$, where $\boldsymbol{A} \in \mathbb{R}^{2 \times 2}$ is nonsingular and $\boldsymbol{b} \in \mathbb{R}^2$ (see Appendix A.1 for specific 'implementation' details). For two grayscale images $\boldsymbol{x} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{u} \in \mathbb{R}^{m' \times n'}$, we write their linear convolution as $(\boldsymbol{x} * \boldsymbol{u})_{ij} = \sum_{(k,l) \in \mathbb{Z}^2} x_{kl} u_{i-k,j-l}$. We write $\boldsymbol{g}_{\sigma^2} \in \mathbb{R}^{\mathbb{Z} \times \mathbb{Z}}$ to denote a (sampled) gaussian with zero mean and variance σ^2. When $\boldsymbol{x} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{u} \in \mathbb{R}^c$, we write $\boldsymbol{x} \otimes \boldsymbol{u} \in \mathbb{R}^{m \times n \times c}$ to denote the 'tensor product' of these elements, with $(\boldsymbol{x} \otimes \boldsymbol{u})_{ijk} = x_{ij} u_k$. We use $\boldsymbol{x} \odot \boldsymbol{u}$ to denote elementwise multiplication of images.

3.1 Conceptual Framework

Given an input image $\boldsymbol{y} \in \mathbb{R}^{m \times n \times c}$ (e.g., $c = 3$ for RGB images), we consider the following general optimization formulation for seeking a structured transformation of the input that explains it in terms of prior observations:

$$\min_{\boldsymbol{\tau}} \varphi(\boldsymbol{y} \circ \boldsymbol{\tau}) + \lambda \rho(\boldsymbol{\tau}). \tag{1}$$

Here, $\boldsymbol{\tau} \in \mathbb{R}^{m' \times n' \times 2}$ gives a vector field representation of transformations of the image plane, and $\lambda > 0$ is a regularization tradeoff parameter. Minimization of the function φ encourages the transformed input image $\boldsymbol{y} \circ \boldsymbol{\tau}$ to be similar to previously-observed images, whereas minimization of ρ regularizes the complexity of the learned transformation $\boldsymbol{\tau}$. Both terms allow to incorporate significant prior information about the visual data and task at hand, and an optimal solution $\boldsymbol{\tau}$ to (1) furnishes an invariant representation of the input $\boldsymbol{y}$.

3.2 Computational Primitive: Optimization for Domain Transformations

We illustrate the flexibility of the general formulation (1) by instantiating it for a variety of classes of visual data. In the most basic setting, we may consider the registration of the input image $\boldsymbol{y}$ to a known motif $\boldsymbol{x}_o$ assumed to be present in the image, and constrain the transformation $\boldsymbol{\tau}$ to lie in a parametric family of transformations $\mathbb{T}$, which yields the optimization formulation

$$\min_{\boldsymbol{\tau}} \frac{1}{2} \left\| \mathcal{P}_\Omega \left[\boldsymbol{g}_{\sigma^2} * (\boldsymbol{y} \circ \boldsymbol{\tau} - \boldsymbol{x}_o) \right] \right\|_F^2 + \chi_{\mathbb{T}}(\boldsymbol{\tau}). \tag{2}$$

Here, Ω denotes a subset of the image plane corresponding to the pixels on which the motif $\boldsymbol{x}_o$ is supported, $\boldsymbol{g}_{\sigma^2}$ is a gaussian filter with variance σ^2 applied individually to each channel, and $\chi_{\mathbb{T}}(\boldsymbol{\tau})$ denotes the characteristic function for the set $\mathbb{T}$ (zero if $\boldsymbol{\tau} \in \mathbb{T}$, $+\infty$ otherwise). The parameters in (2) are illustrated in Figure 2(a-d). We do not directly implement the basic formulation (2) in our experiments, but as a simple model for the more elaborate instantiations of (1) that follow later it furnishes several useful intuitions. For instance, although (2) is a nonconvex optimization problem with a 'rough' global landscape, well-known results suggest that under idealized conditions (e.g., when $\boldsymbol{y} = \boldsymbol{x}_o \circ \boldsymbol{\tau}_o$ for some $\boldsymbol{\tau}_o \in \mathbb{T}$), multiscale solvers that repeatedly solve (2) with a smoothing level σ_k^2 then re-solve initialized at the previous optimal solution with a finer level of smoothing $\sigma_{k+1}^2 < \sigma_k^2$ converge in a neighborhood of the true transformation [LC01; MZM12; KF14; VF14]. This basic fact underpins many classical computer vision methods for image registration and stitching [Bro92; MV98; BM04; Sze07], active appearance models for objects [CET98], and optical flow estimation [HS81; LK81; Ana89], and suggests that (2) is a suitable base for constructing invariant networks.

For our experiments on textured visual data in Figure 1 and Section 4, we will need two elaborations of (2). The first arises due to the problem of obtaining invariant representations for images containing motifs

[1] The function ϕ is compactly supported on the interval $[-2, 2]$, and differentiable with absolutely continuous derivative.

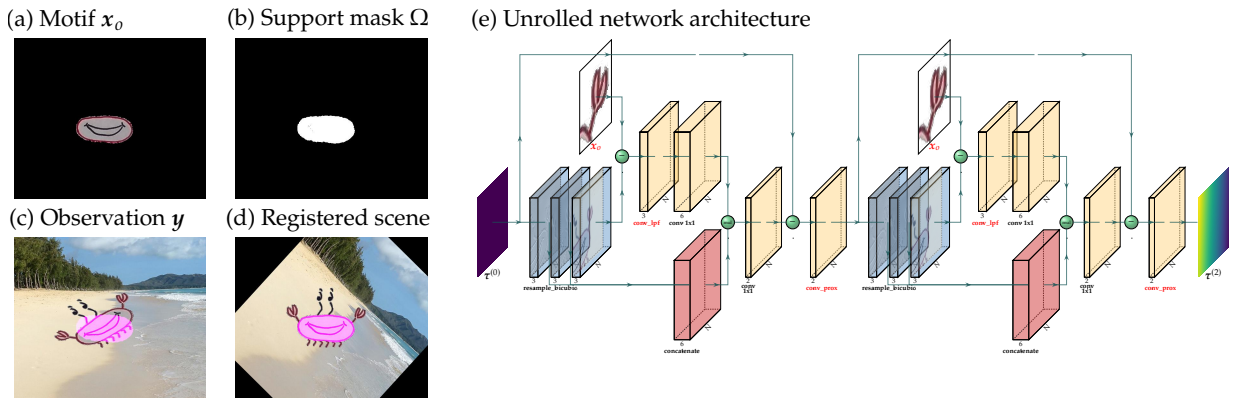

Figure 2: Motif registration with the formulation (1), and an unrolled solver. **(a-d)**: Visualization of components of a registration problem, such as (2). We model observations y as comprising an object involving the motif of interest (here, the body of the crab template we experiment with in Section 4) on a black background, as in (a), embedded in visual clutter (here, the beach background) and subject to a deformation, which leads to a novel pose. A mask Ω for the nonzero pixels of the motif, as in (b), is used to avoid having pixels corresponding to clutter enter the registration cost. After solving this optimization problem, we obtain a transformation τ that registers the observation to the motif, as in (d). In (c-d), we set the red and blue pixels corresponding to the mask Ω to 1 in order to visualize the relative location of the motif. **(e)**: Optimization formulations imply network architectures, via unrolled optimization. Here we show two iterations of an unrolled solver for (2), as we detail in Section 3.3; parameters that could be learned from data, à la unrolled optimization, are highlighted with red text. The operations comprising this unrolled network consist of linear maps, convolutions, and interpolations, leading to efficient implementation on standard hardware accelerators.

x_o appearing in general backgrounds: in such a scenario, the input image y may contain the motif x_o in a completely novel scene (as in Figure 2(c-d)), which makes it inappropriate to smooth the entire motif with the filter g_{σ^2}. In these scenarios, we consider instead a *cost-smoothed* formulation of (2):

$$\min_{\tau} \frac{1}{2} \sum_{\Delta \in \mathbb{Z} \times \mathbb{Z}} (g_{\sigma^2})_\Delta \left\| \mathcal{P}_\Omega \left[y \circ (\tau + \tau_{0,\Delta}) - x_o \right] \right\|_F^2 + \chi_{\mathbb{T}}(\tau). \tag{3}$$

In practice, we take the sum over a finite subset of shifts Δ on which most of the mass of the gaussian filter lies. This formulation is inspired by more general cost-smoothing registration proposals in the literature [MZM12], and it guarantees that pixels of $y \circ \tau$ corresponding to the background Ω^c are never compared to pixels of x_o while incorporating the basin-expanding benefits of smoothing. Second, we consider a more general formulation which also incorporates a low-frequency model for the image background:

$$\min_{\tau, \beta} \frac{1}{2} \left\| \mathcal{P}_{\widetilde{\Omega}} \left[g_{\sigma^2} * (y \circ \tau - x_o - \mathcal{P}_{\Omega^c}[g_{C\sigma^2} * \beta]) \right] \right\|_F^2 + \chi_{\mathbb{T}}(\tau). \tag{4}$$

Here, $\beta \in \mathbb{R}^{m \times n \times c}$ acts as a learnable model for the image background, and $C > 1$ is a fixed constant that guarantees that the background model is at a coarser scale than the motif and image content. The set $\widetilde{\Omega}$ represents a dilation by σ of the motif support Ω, and penalizing pixels in this dilated support ensures that an optimal τ accounts for both foreground and background agreement. We find background modeling essential in computing with scale-changing transformations, such as affine transforms in Figure 1.

3.3 Invariant Networks from Unrolled Optimization

The technique of unrolled optimization allows us to obtain principled network architectures from the optimization formulations developed in Section 3.2. We describe the basic approach using the abstract formulation (1). For a broad class of regularizers ρ, the proximal gradient method [PB14] can be used to attempt to solve the nonconvex problem (1): it defines a sequence of iterates

$$\tau^{(t+1)} = \text{prox}_{\lambda \nu_t \rho} \left(\tau^{(t)} - \nu_t \nabla_\tau \varphi(y \circ \tau^{(t)}) \right) \tag{5}$$

from a fixed initialization $\boldsymbol{\tau}^{(0)}$, where $\nu_t > 0$ is a step size sequence and $\operatorname{prox}_\rho(\boldsymbol{\tau}) = \arg\min_{\boldsymbol{\tau}'} \frac{1}{2}\|\boldsymbol{\tau} - \boldsymbol{\tau}'\|_F^2 + \rho(\boldsymbol{\tau}')$ is well-defined if ρ is a proper convex function. Unrolled optimization suggests to truncate this iteration after T steps, and treat the iterate $\boldsymbol{\tau}^{(T)}$ at that iteration as the output of a neural network. One can then learn certain parameters of the neural network from datasets, as a principled approach to combining the structural priors of the original optimization problem with the benefits of a data-driven approach.

In Figure 2(e), we show an architectural diagram for a neural network unrolled from a proximal gradient descent solver for the registration formulation (2). We always initialize our networks with $\boldsymbol{\tau}^{(0)}$ as the identity transformation field, and in this context we have $\operatorname{prox}_{\lambda\nu_t\rho}(\boldsymbol{\tau}) = \operatorname{proj}_{\mathbb{T}}(\boldsymbol{\tau})$ as the nearest point in $\mathbb{T}$ to $\boldsymbol{\tau}$, which can be computed efficiently (computational details are provided in Appendix A.1). The cost (2) is differentiable; calculating its gradient as in Appendix A.2, (5) becomes

$$\boldsymbol{\tau}^{(t+1)} = \operatorname{proj}_{\mathbb{T}}\left(\boldsymbol{\tau}^{(t)} - \nu_t \sum_{k=0}^{c-1} \left(\boldsymbol{g}_{\sigma^2} * \mathcal{P}_\Omega\left[\boldsymbol{g}_{\sigma^2} * \left(\boldsymbol{y} \circ \boldsymbol{\tau}^{(t)} - \boldsymbol{x}_o\right)_k\right] \otimes \mathbf{1}_2\right) \odot \left(\mathrm{d}\boldsymbol{y}_k \circ \boldsymbol{\tau}^{(t)}\right)\right), \tag{6}$$

as we represent visually in Figure 2(e), where a subscript of k denotes the k-th channel of the image and $\mathrm{d}\boldsymbol{y} \in \mathbb{R}^{m\times n\times c\times 2}$ is the Jacobian matrix of $\boldsymbol{y}$. The constituent operations in this network are convolutions, pointwise nonlinearities and linear maps, which lend themselves ideally to implementation in standard deep learning software packages and on hardware accelerators; and because the cubic convolution interpolation kernel ϕ is twice continuously differentiable except at four points of $\mathbb{R}$, these networks are end-to-end differentiable and can be backpropagated through efficiently. The calculations necessary to instantiate unrolled network architectures for other optimization formulations used in our experiments are deferred to Appendix A.2. A further advantage of the unrolled approach to network design is that hyperparameter selection becomes directly connected to convergence properties of the optimization formulation (1): we demonstrate how theory influences these selections in Section 5, and provide practical guidance for registration and detection problems through our experiments in Sections 4, 5.2 and 6.

4 Invariant Networks for Hierarchical Object Detection

The unrolled networks in Section 3 represent architectural primitives for building deformation-invariant neural networks: they are effective at producing invariant representations for input images containing local motifs. In this section, we illustrate how these local modules can be combined into a network that performs invariant processing of nonlocally-structured visual data, via an invariant hierarchical object detection task with a fixed template. For simplicity, in this section we will focus on the setting where $\mathbb{T}$ is the set of rigid motions of the image plane (i.e., translations and rotations), which we will write as SE(2).

4.1 Data Model and Problem Formulation

We consider an object detection task, where the objective is to locate a fixed template with independently-articulating parts (according to a SE(2) motion model) in visual clutter. More precisely, we assume the template is decomposable into a hierarchy of deformable parts, as in Figure 3(a): at the top level of the hierarchy is the template itself, with concrete visual motifs at the lowest levels that correspond to specific pixel subsets of the template, which constitute independent parts. Because these constituent parts deform independently of one another, detecting this template *efficiently* demands an approach to detection that captures the specific hierarchical structure of the template.[2] Compared to existing methods for parts-based object detection that are formulated to work with objects subject to complicated variations in appearance [FMR08; FGMR10; GIDM15; PVGR15], focusing on the simpler setting of template detection allows us to develop a network that guarantees invariant detection under the motion model, and can incorporate large-scale statistical learning techniques by virtue of its unrolled construction (although we leave this latter

[2]Reasoning as in Section 1, the effective dimension of the space of all observable transformations of the object is the product of the dimension of the motion model and the number of articulating parts. A detector that exploits the hierarchical structure of the object effectively reduces the dimensionality to dim(motion model) + log(number of parts), yielding a serious advantage for moderate-dimensional families of deformations.

direction for future work). We note that other approaches are possible, such as hierarchical sparse modeling [BS10; JMOB10] or learning a graphical model [SM12].

More formally, we write $\boldsymbol{y}_o \in \mathbb{R}^{m_o \times n_o \times 3}$ for the RGB image corresponding to a canonized view of the template to be detected (e.g., the crab at the top of the hierarchy in Figure 3(a) left) embedded on a black background. For a K-motif object (e.g., $K = 4$ for the crab template), we let $\boldsymbol{x}_k \in \mathbb{R}^{m_k \times n_k \times 3}$ denote the k distinct (canonized, black-background-embedded) transforming motifs in the object, each with non-overlapping occurrence coordinates $(i_k, j_k) \in \{0, \dots, m_o\} \times \{0, \dots, n_o\}$. The template $\boldsymbol{y}_o$ decomposes as

$$\boldsymbol{y}_o = \underbrace{\sum_{k=1}^{K} \boldsymbol{x}_k * \boldsymbol{e}_{i_k j_k}}_{\text{transforming motifs}} + \underbrace{\boldsymbol{y}_o - \sum_{k=1}^{K} \boldsymbol{x}_k * \boldsymbol{e}_{i_k j_k}}_{\text{static body}}. \tag{7}$$

For example, the four transforming motifs for the crab template in Figure 3(a) are the two claws and two eyes. In our experiments with the crab template, we will consider detection of transformed templates $\boldsymbol{y}_{\text{obs}}$ of the following form:

$$\boldsymbol{y}_{\text{obs}} = \left[\sum_{k=1}^{K} (\boldsymbol{x}_k * \boldsymbol{e}_{i_k j_k}) \circ \boldsymbol{\tau}_k + \left(\boldsymbol{y}_o - \sum_{k=1}^{K} \boldsymbol{x}_k * \boldsymbol{e}_{i_k j_k} \right) \right] \circ \boldsymbol{\tau}_0, \tag{8}$$

where $\boldsymbol{\tau}_0 \in \text{SE}(2)$, and $\boldsymbol{\tau}_k \in \text{SO}(2)$ is sufficiently close to the identity transformation (which represents the physical constraints of the template). The detection task is then to decide, given an input scene $\boldsymbol{y} \in \mathbb{R}^{m \times n \times 3}$ containing visual clutter (and, in practice, $m \gg m_o$ and $n \gg n_o$), whether or not a transformed instance $\boldsymbol{y}_{\text{obs}}$ appears in $\boldsymbol{y}$ or not, and to output estimates of its transformation parameters $\boldsymbol{\tau}_k$.

Although our experiments will pertain to the observation model (8), as it agrees with our decomposition of the crab template in Figure 3(a), the networks we construct in Section 4.3 will be amenable to more complex observation models where parts at intermediate levels of the hierarchy also transform.[3] To this end, we introduce additional notation that captures the hierarchical structure of the template $\boldsymbol{y}_o$. Concretely, we identify a hierarchically-structured template with a directed rooted tree $G = (V, E)$, with 0 denoting the root node, and $1, \dots, K$ denoting the K leaf nodes. Our networks will treat observations of the form

$$\boldsymbol{y}_{\text{obs}} = \sum_{k=1}^{K} ((\cdots (((\boldsymbol{x}_k * \boldsymbol{e}_{i_k j_k}) \circ \boldsymbol{\tau}_k) \circ \boldsymbol{\tau}_{v_{d(k)-1}}) \circ \cdots) \circ \boldsymbol{\tau}_{v_1}) \circ \boldsymbol{\tau}_0 + \left(\boldsymbol{y}_o - \sum_{k=1}^{K} \boldsymbol{x}_k * \boldsymbol{e}_{i_k j_k} \right) \circ \boldsymbol{\tau}_0, \tag{9}$$

where $d(k)$ is the depth of node k, and $v_1, \dots, v_{d(k)-1} \in V$ with $0 \to v_1 \to \cdots \to v_{d(k)-1} \to k$ specifying the path from the root node to node k in G. To motivate the observation model (9), consider the crab example of Figure 3(a), where in addition we imagine the coordinate frame of the eye pair motif transforms independently with a transformation $\boldsymbol{\tau}_5$: in this case, the observation model (9) can be written in an equivalent 'hierarchical' form

$$\boldsymbol{y}_{\text{obs}} = \left[(\boldsymbol{x}_1 * \boldsymbol{e}_{i_1 j_1}) \circ \boldsymbol{\tau}_1 + (\boldsymbol{x}_2 * \boldsymbol{e}_{i_2 j_2}) \circ \boldsymbol{\tau}_2 + \left[(\boldsymbol{x}_3 * \boldsymbol{e}_{i_3 j_3}) \circ \boldsymbol{\tau}_3 + (\boldsymbol{x}_4 * \boldsymbol{e}_{i_4 j_4}) \circ \boldsymbol{\tau}_4 \right] \circ \boldsymbol{\tau}_5 \right] \circ \boldsymbol{\tau}_0 + \boldsymbol{y}_{\text{body}} \circ \boldsymbol{\tau}_0,$$

by linearity of the interpolation operation $\boldsymbol{x} \mapsto \boldsymbol{x} \circ \boldsymbol{\tau}$ (with $\boldsymbol{y}_{\text{body}} = \boldsymbol{y}_o - \sum_k \boldsymbol{x}_k * \boldsymbol{e}_{i_k j_k}$).

4.2 Aside: Optimization Formulations for Registration of "Spiky" Motifs

To efficiently perform hierarchical invariant detection of templates following the model (9), the networks we design will build from the following basic paradigm, given an input scene $\boldsymbol{y}$:

1. **Visual motif detection**: First, perform detection of all of the lowest-level motifs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_K$ in $\boldsymbol{y}$. The output of this process is an *occurrence map* for each of the K transforming motifs, i.e. an $m \times n$ image taking (ideally) value 1 at the coordinates where detections occur and 0 elsewhere.

[3] For example, consider a simple extension of the crab template in Figure 3(a), where the left and right claw motifs are further decomposed into two pairs of pincers plus the left and right arms, with opening and closing motions for the pincers, and the same SO(2) articulation model for the arms (which naturally moves the pincers in accordance with the rotational motion).

2. **Spiky motif detection for hierarchical motifs**: Detect intermediate-level abstractions using the occurrence maps in $\boldsymbol{y}$ obtained in the previous step. For example, if $k = 3$ and $k = 4$ index the left and right eye motifs in the crab template of Figure 3(a), detection of the eye pair motif corresponds to registration of the canonized eye pair's occurrence map against the two-channel image corresponding to the concatenation of $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$'s occurrence maps in $\boldsymbol{y}$.

3. **Continue until the top of the hierarchy**: This occurrence map detection process is iterated until the top level of the hierarchy. For example, in Figure 3(a), a detection of the crab template occurs when the multichannel image corresponding to the occurrence maps for the left and right claws and the eye pair motif is matched.

To instantiate this paradigm, we find it necessary to develop a separate registration formulation for registration of occurrence maps, beyond the formulations we have introduced in Section 3. Indeed, occurrence maps contain no texture information and are maximally localized, motivating a formulation that spreads out gradient information and avoids interpolation artifacts—and although there is still a need to cope with clutter in general, the fact that the occurrence maps are generated on a black background obviates the need for extensive background modeling, as in (4). We therefore consider the following "complementary smoothing" formulation for spike registration: for a c-channel occurrence map $\boldsymbol{y}$ and canonized occurrence map $\boldsymbol{x}_o$, we optimize over the affine group $\mathrm{Aff}(2) = \mathrm{GL}(2) \rtimes \mathbb{R}^2$ as

$$\min_{\boldsymbol{A},\boldsymbol{b}} \frac{1}{2c} \left\| \boldsymbol{g}_{\sigma^2\boldsymbol{I}-\sigma_0^2\boldsymbol{A}\boldsymbol{A}^*} * \left(\det{}^{-1/2}(\boldsymbol{A}\boldsymbol{A}^*) \left(\boldsymbol{g}_{\sigma_0^2\boldsymbol{I}} * \boldsymbol{y} \right) \circ \boldsymbol{\tau}_{\boldsymbol{A}^{-1},-\boldsymbol{A}^{-1}\boldsymbol{b}} \right) - \boldsymbol{g}_{\sigma^2\boldsymbol{I}} * \boldsymbol{x}_o \right\|_F^2 + \chi_{\mathrm{Aff}(2)}(\boldsymbol{A},\boldsymbol{b}), \tag{10}$$

where $\boldsymbol{g}_{\boldsymbol{M}}$ denotes a single-channel centered gaussian filter with positive definite covariance matrix $\boldsymbol{M} \succ 0$, and correlations are broadcast across channels. Here, $\sigma > 0$ is the main smoothing parameter to propagate gradient information, and $\sigma_0 > 0$ is an additional smoothing hyperparameter to mitigate interpolation artifacts.

In essence, the key modifications in (10) that make it amenable to registration of occurrence maps are the compensatory effects for scaling that it introduces: transformations that scale the image correspondingly reduce the amplitude of the (smoothed) spikes, which is essential given the discrete, single-pixel spike images we will register. Of course, since we consider only euclidean transformations in our experiments in this section, we always have $\boldsymbol{A}\boldsymbol{A}^* = \boldsymbol{I}$, and the problem (10) can be implemented in a simpler form. However, these modifications lead the problem (10) to work excellently for scale-changing transformations as well: we explore the reasons behind this from both theoretical and practical perspectives in Section 5.

4.3 Invariant Network Architecture

The networks we design to detect under the observation model (9) consist of a configuration of unrolled motif registration networks, as in Figure 2(e), arranged in a 'bottom-up' hierarchical fashion, following the hierarchical structure in the example shown in Figure 3(a). The configuration for each motif registration subnetwork is a 'GLOM-style' [Hin21] collection of the networks sketched in Figure 2(e), oriented at different pixel locations in the input scene $\boldsymbol{y}$; the transformation parameters predicted of each of these configurations are aggregated across the image, weighted by the final optimization cost (as a measure of quality of the final solution), in order to determine detections. These detections are then used as feature maps for the next level of occurrence motifs, which in turn undergo the same registration-detection process until reaching the top-level object's occurrence map, which we use as a solution to the detection problem. A suitable unrolled implementation of the registration and detection process leads to a network that is end-to-end differentiable and amenable to implementation on standard hardware acceleration platforms (see Section 4.4).

We now describe this construction formally, following notation introduced in Section 4.1. The network input is an RGB image $\boldsymbol{y} \in \mathbb{R}^{m \times n \times 3}$. We shall assume that the canonized template $\boldsymbol{y}_o$ is given, as are as the canonized visual motifs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K$ and their masks $\Omega_1, \ldots, \Omega_K$; we also assume that for every $v \in V$ with $v \notin \{1, \ldots, K\}$, we are given the canonized occurrence map $\boldsymbol{x}_v \in \mathbb{R}^{m_v \times n_v \times c_v}$ of the hierarchical feature v in $\boldsymbol{y}_o$. In practice, one obtains these occurrence maps through a process of "extraction", using $\boldsymbol{y}_o$ as an input to the network, which we describe in Appendix A.4. The network construction can be separated into three distinct steps:

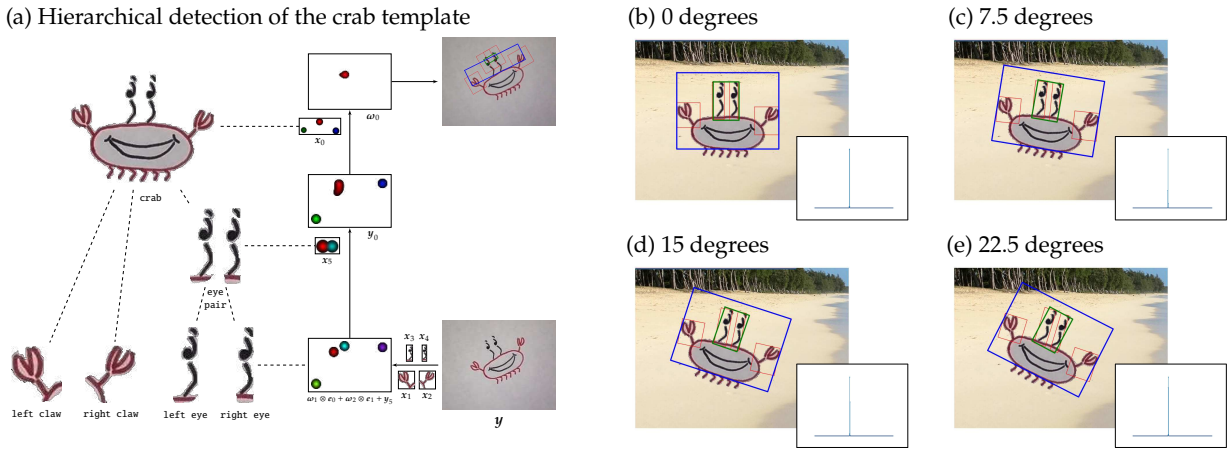

Figure 3: An example of a hierarchically-structured template, and the results of an implementation of our detection network. **(a)**: Structure of the crab template, described in Section 4.1, and its interaction with our network architecture for detection, described in Section 4.3. *Left: top-down decomposition of the template into motifs.* A template of interest $\boldsymbol{y}_o$ (here, the crab at top left) is decomposed into a hierarchy of abstractions. The hierarchical structure is captured by a tree $G = (V, E)$: nodes represent parts or aggregations of parts, and edges represent their relationships. *Right: bottom-up detection of the template in a novel scene.* To detect the template in a novel scene and pose, the network described in Section 4.3 first localizes each of the lowest-level visual motifs at left and their transformation parameters in the input scene $\boldsymbol{y}$ (bottom right). Motifs and the derived occurrence maps are labeled in agreement with the notation we introduce in Section 4.3. The output of each round of optimization is an occurrence map $\boldsymbol{\omega}_v$ for nodes $v \in V$; these occurrence maps then become the inputs for detection of the next level of concepts, following the connectivity structure of G, until the top-level template is reached (top right). **(b-e)**: Concrete results for the hierarchical invariant object detection network implemented in Section 4.4: the learned transformation at the minimum-error stride for each motif is used to draw the motifs' transformed bounding boxes. Insets at the bottom right corner of each result panel visualize the quality of the final detection trace $\boldsymbol{\omega}_0$ for the template, with a value of 1 at the top of the inset.

Traversal. The network topology is determined by a simple traversal of the graph G. For each $v \in V$, let $d(v)$ denote the shortest-path distance from 0 to v, with unit weights for edges in E (the "depth" of v in G). We will process motifs in a deepest-first order for convenience, although this is not strictly necessary in all cases (e.g. for efficiency, it might be preferable to process all leaf nodes $1, \ldots, K$ first). Let $\mathrm{diam}(G) = \max_{v \in V} d(v)$, and for an integer ℓ no larger than $\mathrm{diam}(G)$, we let $D(\ell) \in \mathbb{N}$ denote the number of nodes in V that are at depth ℓ.

Motif detection at one depth. Take any integer $0 \leq \ell \leq \mathrm{diam}(G)$, and let $v_1, \ldots, v_{D(\ell)}$ denote the nodes in G at depth ℓ, enumerated in increasing order (say). For each such vertex v_k, perform the following steps:

1. Is this a leaf? If the neighborhood $\{v' \mid (v_k, v') \in E\}$ is empty, this node is a leaf; otherwise it is not. Subsequent steps depend on this distinction. We phrase the condition more generally, although we have defined $1, \ldots, K$ as the leaf vertices here, to facilitate some implementation-independence.

2. Occurrence map aggregation for non-leaves: If v_k is not a leaf, construct its detection feature map from lower-level detections: concretely, let

$$\boldsymbol{y}_{v_k} = \sum_{v' : (v_k, v') \in E} \boldsymbol{\omega}_{v'} \otimes \boldsymbol{e}_{\pi_{v_k}(v')}, \tag{11}$$

where $\pi_{v_k}(v')$ denotes a vertex-increasing-order enumeration of the set $\{v' : (v_k, v') \in E\}$ starting from 0. By construction (see the fourth step below), $\boldsymbol{y}_{v_k}$ has the same width and height as the input scene $\boldsymbol{y}$, but $c_{v_k} = |\{v' : (v_k, v') \in E\}|$ channels (one for each child node) instead of 3 RGB channels.

3. Perform strided registration: Because the motif $\boldsymbol{x}_{v_k}$ is in general much smaller in size than the scene $\boldsymbol{y}_{v_k}$, and because the optimization formulation (1) is generally nonconvex with a finite-radius basin of attraction around the true transformation parameters in the model (9), the detection process consists of a search for $\boldsymbol{x}_{v_k}$ anchored at a grid of points in $\boldsymbol{y}_{v_k}$. Concretely, let

$$\Lambda_{v_k} = \{(i\Delta_{H,v_k}, j\Delta_{W,v_k}) \mid (i,j) \in \{0,\dots,m-1\} \times \{0,\dots,n-1\}\} \cap (\{0,\dots,m-1\} \times \{0,\dots,n-1\})$$

denote the grid for the v_k-th motif; here Δ_{H,v_k} and Δ_{W,v_k} define the vertical and horizontal stride lengths of the grid (we discuss choices of these and other hyperparameters introduced below in Appendix A.4). When v_k is a leaf, for each $\boldsymbol{\lambda} \in \Lambda_{v_k}$, we let $(\boldsymbol{U}(v_k,\boldsymbol{\lambda}), \boldsymbol{b}(v_k,\boldsymbol{\lambda})) \in \mathrm{SE}(2)$ denote the parameters obtained after running an unrolled solver for the cost-smoothed visual motif registration problem

$$\min_{\boldsymbol{\tau}} \frac{1}{2} \sum_{\boldsymbol{\Delta} \in \mathbb{Z} \times \mathbb{Z}} (\boldsymbol{g}_{\sigma^2_{v_k}})_{\boldsymbol{\Delta}} \left\| \mathcal{P}_{\Omega_{v_k}} \left[\left(\boldsymbol{g}_{\sigma^2_{\mathrm{in}}} * \boldsymbol{y} \right) \circ (\boldsymbol{\tau} + \boldsymbol{\tau}_{\mathbf{0},\boldsymbol{\Delta}+\boldsymbol{\lambda}}) - \boldsymbol{x}_{v_k} \right] \right\|_F^2 + \chi_{\mathrm{SE}(2)}(\boldsymbol{\tau}), \tag{12}$$

for T_{v_k} iterations, with step size ν_{v_k}. In addition, we employ a two-step multiscale smoothing strategy, which involves initializing an unrolled solver for (12) with a much smaller smoothing parameter $(\sigma'_{v_k})^2$ at $(\boldsymbol{U}(v_k,\boldsymbol{\lambda}), \boldsymbol{b}(v_k,\boldsymbol{\lambda}))$ and running it for an additional fixed number of iterations; we let $\mathrm{loss}(v_k,\boldsymbol{\lambda})$ denote the final objective function value after this multiscale process, and abusing notation, we let $(\boldsymbol{U}(v_k,\boldsymbol{\lambda}), \boldsymbol{b}(v_k,\boldsymbol{\lambda}))$ denote the updated final parameters . Precise implementation details are discussed in Appendix A.4. When v_k is not a leaf, we instead define the same fields on the grid Λ_{v_k} via a solver for the spike registration problem

$$\min_{\boldsymbol{\tau}} \frac{1}{2c_{v_k}} \left\| \mathcal{P}_{\Omega_{v_k}} \left[\boldsymbol{g}_{\sigma^2_{v_k} - \sigma^2_{0,v_k}} * \left(\boldsymbol{y}_{v_k} \circ (\boldsymbol{\tau} + \boldsymbol{\tau}_{\mathbf{0},\boldsymbol{\lambda}}) - \boldsymbol{x}_{v_k} \right) \right] \right\|_F^2 + \chi_{\mathrm{SE}(2)}(\boldsymbol{\tau}), \tag{13}$$

with Ω_{v_k} denoting a dilated bounding box for $\boldsymbol{x}_{v_k}$, and otherwise the same notation and hyperparameters. We do not use multiscale smoothing for non-leaf motifs.

4. Aggregate registration outputs into detections (occurrence maps): We convert the registration fields into detection maps, by computing

$$\boldsymbol{\omega}_{v_k} = \sum_{\boldsymbol{\lambda} \in \Lambda_{v_k}} \left(\boldsymbol{g}_{\sigma^2_{0,v_k}} * \boldsymbol{e}_{\boldsymbol{\lambda}+\boldsymbol{b}(v_k,\boldsymbol{\lambda})} \right) \exp\left(-\alpha_{v_k} \max\{0, \mathrm{loss}(v_k,\boldsymbol{\lambda}) - \gamma_{v_k}\} \right), \tag{14}$$

where each summand $\boldsymbol{g}_{\sigma^2_{0,v_k}} * \boldsymbol{e}_{\boldsymbol{\lambda}+\boldsymbol{b}(v_k,\boldsymbol{\lambda})}$ is truncated to be size $m \times n$.[4] The scale and threshold parameters α_{v_k} and γ_{v_k} appearing in this formula are calibrated to achieve a specified level of performance under an assumed maximum level of visual clutter and transformation for the observations (9), as discussed in Appendix A.4.

We prefer to embed detections as occurrence maps and use these as inputs for higher-level detections using optimization, rather than a possible alternate approach (e.g. extracting landmarks and processing these using group synchronization), in order to have each occurrence map $\boldsymbol{\omega}_v$ for $v \in V$ be differentiable with respect to the various filters and hyperparameters.

Template detection. To perform detection given an input $\boldsymbol{y}$, we repeat the four steps in the previous section for each motif depth, starting from depth $\ell = \mathrm{diam}(G)$, and each motif at each depth. After processing depth $\ell = 0$, the output occurrence map $\boldsymbol{\omega}_0$ can be thresholded to achieve a desired level of detection performance for observations of the form (9). The detection process is summarized as Algorithm 1.

By construction, this output $\boldsymbol{\omega}_0$ can be differentiated with respect to each node $v \in V$'s hyperparameters or filters, and the unrolled structure of the sub-networks and G's topology can be used to efficiently calculate such gradients via backpropagation. In addition, although we do not use the full transformation parameters $\boldsymbol{U}(\boldsymbol{\lambda}, v)$ calculated in the registration operations (12) and (13), these can be leveraged for various purposes (e.g. drawing detection bounding boxes, as in our experimental evaluations in Section 4.4).

[4] This convolutional notation is of course an abuse of notation, to avoid having to define a gaussian filter with a general mean parameter. In practice, this latter technique is both more efficient to implement and leads to a stably-differentiable occurrence map.

Algorithm 1 Invariant Hierarchical Motif Detection Network, Summarizing Section 4.3

input scene $\boldsymbol{y}$, graph $G = (V, E)$, motifs $(\boldsymbol{x}_v, \Omega_v)_{v \in V}$, hyperparameters $(\nu_v, T_v, \Delta_{H,v}, \Delta_{W,v}, \sigma_v^2, \sigma_{0,v}^2, \alpha_v, \gamma_v)_{v \in V}$
  set $\operatorname{diam}(G)$ and node enumerations by depth-first traversal of G
  for all depths $\ell = \operatorname{diam}(G), \operatorname{diam}(G) - 1, \ldots, 0$ **do**
    for all nodes v at depth ℓ **do**
      set $N_v = \{v' \mid (v, v') \in E\}$ and $c_v = |N_v|$
      if $c_v > 0$ **then**
        concatenate occurrence maps into $\boldsymbol{y}_v = \sum_{v' \in N_v} \boldsymbol{\omega}_{v'} \otimes \boldsymbol{e}_{\pi_v(v')}$
      for all $\boldsymbol{\lambda} \in \Lambda_v(\Delta_{H,v}, \Delta_{W,v})$ **do**
        if $c_v > 0$ **then**
          set $\boldsymbol{U}(v, \boldsymbol{\lambda}), \boldsymbol{b}(v, \boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\tau}} \frac{1}{2c_v} \|\boldsymbol{g}_{\sigma_v^2 - \sigma_{0,v}^2} * (\boldsymbol{y}_v \circ (\boldsymbol{\tau} + \boldsymbol{\tau}_{\mathbf{0},\boldsymbol{\lambda}}) - \boldsymbol{x}_v)\|_F^2 + \chi_{\mathrm{SE}(2)}(\boldsymbol{\tau})$
          set $\operatorname{loss}(v, \boldsymbol{\lambda}) = \min_{\boldsymbol{\tau}} \frac{1}{2c_v} \|\boldsymbol{g}_{\sigma_v^2 - \sigma_{0,v}^2} * (\boldsymbol{y}_v \circ (\boldsymbol{\tau} + \boldsymbol{\tau}_{\mathbf{0},\boldsymbol{\lambda}}) - \boldsymbol{x}_v)\|_F^2 + \chi_{\mathrm{SE}(2)}(\boldsymbol{\tau})$
          (both with a T_v-layer unrolled solver)
        else
          set $\boldsymbol{U}(v, \boldsymbol{\lambda}), \boldsymbol{b}(v, \boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\tau}} \frac{1}{2} \sum_{\Delta} (\boldsymbol{g}_{\sigma_v^2})_{\Delta} \|\mathcal{P}_{\Omega_v}[(\boldsymbol{g}_{\sigma_{\mathrm{in}}^2} * \boldsymbol{y}) \circ (\boldsymbol{\tau} + \boldsymbol{\tau}_{\mathbf{0},\Delta+\boldsymbol{\lambda}}) - \boldsymbol{x}_v]\|_F^2 + \chi_{\mathrm{SE}(2)}(\boldsymbol{\tau})$
          set $\operatorname{loss}(v, \boldsymbol{\lambda}) = \min_{\boldsymbol{\tau}} \frac{1}{2} \sum_{\Delta} (\boldsymbol{g}_{\sigma_v^2})_{\Delta} \|\mathcal{P}_{\Omega_v}[(\boldsymbol{g}_{\sigma_{\mathrm{in}}^2} * \boldsymbol{y}) \circ (\boldsymbol{\tau} + \boldsymbol{\tau}_{\mathbf{0},\Delta+\boldsymbol{\lambda}}) - \boldsymbol{x}_v]\|_F^2 + \chi_{\mathrm{SE}(2)}(\boldsymbol{\tau})$
          (both with a T_v-layer unrolled solver, with two-round multiscale smoothing)
      construct the occurrence map $\boldsymbol{\omega}_v = \sum_{\boldsymbol{\lambda} \in \Lambda_v} (\boldsymbol{g}_{\sigma_{0,v}^2} * \boldsymbol{e}_{\boldsymbol{\lambda} + \boldsymbol{b}(v,\boldsymbol{\lambda})}) \exp(-\alpha_v \max\{0, \operatorname{loss}(v, \boldsymbol{\lambda}) - \gamma_v\})$
output template occurrence map $\boldsymbol{\omega}_0$

4.4 Implementation and Evaluation

We implement the hierarchical invariant object detection network described in Section 4.3 in PyTorch [PGML+19], and test it for detection of the crab template from Figure 3(a) subject to a global rotation (i.e., $\boldsymbol{\tau}_0$ in the model (9)) of varying size (Figure 3(b-e)). In 512×384 pixel scenes on a "beach" background, a calibrated detector perfectly detects the crab from its constituent parts up to rotations of $\pi/8$ radians—at rotations around $\pi/6$, a multiple-instance issue due to similarity between the two eye motifs begins to hinder the detection performance. Traces in each panel of Figure 3, right demonstrate the precise localization of the template.

For hyperparameters, we set $T_v = 1024$ and $\Delta_{H,v} = \Delta_{W,v} = 20$ for all $v \in V$, and calibrate detection parameters as described in Appendix A.4; for visual motifs, we calibrate the remaining registration hyperparameters as described in Appendix A.4 on a per-motif basis, and for spike motifs, we find the prescriptions for σ_v^2 and the step sizes ν_v implied by theory (Section 5) to work excellently without any fine-tuning. We also implement selective filtering of strides for spiky motif alignment that are unlikely to succeed: due to the common background, this type of screening is particularly effective here. The strided registration formulations (12) and (13) afford efficient batched implementation on a hardware accelerator, given that the motifs $\boldsymbol{x}_v$ for $v \in V$ are significantly smaller than the full input scene $\boldsymbol{y}$, and the costs only depend on pixels near to the motifs $\boldsymbol{x}_v$. On a single NVIDIA TITAN X Pascal GPU accelerator (12 GB memory), it takes approximately five minutes to complete a full detection. We expect throughput to be further improvable without sacrificing detection performance by decreasing the maximum iterations for each unrolled network T_v even further—the setting of $T_v = 1024$ is conservative, with convergence typically much more rapid. Our implementation is available at `https://github.com/sdbuch/refine`.

5 Guaranteed, Efficient Detection of Occurrence Maps

In Section 4, we described how invariant processing of hierarchically-structured visual data naturally leads to problems of registering 'spiky' occurrence maps, and we introduced the formulation (10) for this purpose. In this section, we provide a theoretical analysis of a continuum model for the proximal gradient descent method applied to the optimization formula (10). A byproduct of our analysis is a concrete prescription for the step size and rate of smoothing—in Section 5.2, we demonstrate experimentally that these prescriptions work excellently for the discrete formulation (10), leading to rapid registration of the input scene.

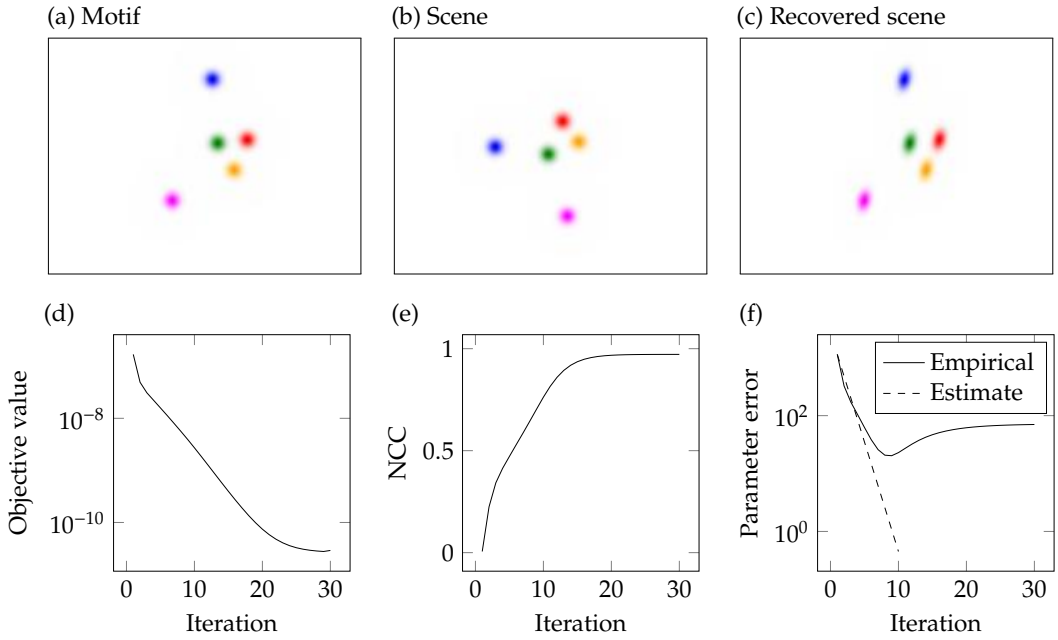

Figure 4: Numerical verification of Theorem 5.1. **(a):** A multichannel spike motif containing 5 spikes. **(b):** A scene generated by applying a random affine transformation to the motif. **(c):** The solution to (10) with these data. The skewing apparent here is undone by the compensated external gaussian filter, which enables accurate localization in spite of these artifacts. **(d):** Change in objective value of (10) across iterations of proximal gradient descent. Convergence occurs in tens of iterations. **(e):** Change in normalized cross correlation across iterations (see Appendix A.3). We observe that the method successfully registers the multichannel spike scene. **(f):** Comparison between the left and right-hand side of equation (18) with gradient descent iterates from (10) (labeled as φ here). After an initial faster-than-predicted linear rate, the discretized solver saturates at a sub-optimal level. This is because because accurate estimation of the transformation parameters $(\boldsymbol{A}, \boldsymbol{b})$ requires subpixel-level preciseness, which is affected by discretization and interpolation artifacts. It does not hinder correct localization of the scene, as (e) shows.

5.1 Multichannel Spike Model

We consider continuous signals defined on $\mathbb{R}^2$ in this section, as an 'infinite resolution' idealization of discrete images, free of interpolation artifacts. We refer to Appendix B for full technical details. Consider a target signal

$$\boldsymbol{X}_o = \sum_{i=1}^{c} \boldsymbol{\delta}_{\boldsymbol{v}_i} \otimes \boldsymbol{e}_i,$$

where $\boldsymbol{\delta}_{\boldsymbol{v}_i}$ is a Dirac distribution centered at the point $\boldsymbol{v}_i$, and an observation

$$\boldsymbol{X} = \sum_{i=1}^{c} \boldsymbol{\delta}_{\boldsymbol{u}_i} \otimes \boldsymbol{e}_i,$$

satisfying

$$\boldsymbol{v}_i = \boldsymbol{A}_\star \boldsymbol{u}_i + \boldsymbol{b}_\star.$$

In words, the observed signal is an affine transformation of the spike signal $\boldsymbol{X}_o$, as in Figure 4(a, b). This model is directly motivated by the occurrence maps (11) arising in our hierarchical detection networks. Following (10), consider the objective function

$$\varphi_{L^2,\sigma}(\boldsymbol{A}, \boldsymbol{b}) \equiv \frac{1}{2c} \sum_{i=1}^{c} \left\| \boldsymbol{g}_{\sigma^2 \boldsymbol{I} - \sigma_0^2 (\boldsymbol{A}^* \boldsymbol{A})^{-1}} * \left(\det{}^{1/2}(\boldsymbol{A}^* \boldsymbol{A}) \left(\boldsymbol{g}_{\sigma_0^2 \boldsymbol{I}} * \boldsymbol{X}_i \right) \circ \boldsymbol{\tau}_{\boldsymbol{A},\boldsymbol{b}} \right) - \boldsymbol{g}_{\boldsymbol{0},\sigma^2 \boldsymbol{I}} * (\boldsymbol{X}_o)_i \right\|_{L^2}^2 .$$

We study the following "inverse parameterization" of this function:

$$\varphi^{\mathrm{inv}}_{L^2,\sigma}(\boldsymbol{A},\boldsymbol{b}) \equiv \varphi_{L^2,\sigma}(\boldsymbol{A}^{-1},-\boldsymbol{A}^{-1}\boldsymbol{b}). \tag{15}$$

We analyze the performance of gradient descent for solving the optimization problem

$$\min_{\boldsymbol{A},\boldsymbol{b}} \varphi^{\mathrm{inv}}_{L^2,\sigma}(\boldsymbol{A},\boldsymbol{b}).$$

Under mild conditions, local minimizers of this problem are global. Moreover, if σ is set appropriately, the method exhibits linear convergence to the truth:

Theorem 5.1 (Multichannel Spike Model, Affine Transforms, L^2)**.** *Consider an instance of the multichannel spike model, with $\boldsymbol{U} = [\boldsymbol{u}_1,\dots,\boldsymbol{u}_c] \in \mathbb{R}^{2\times c}$. Assume that the spikes $\boldsymbol{U}$ are centered and nondegenerate, so that $\boldsymbol{U}\boldsymbol{1} = \boldsymbol{0}$ and* $\mathrm{rank}(\boldsymbol{U}) = 2$*. Then gradient descent*

$$\begin{aligned}\boldsymbol{A}_{k+1} &= \boldsymbol{A}_k - t_{\boldsymbol{A}}\nabla_{\boldsymbol{A}}\varphi^{\mathrm{inv}}_{L^2,\sigma}(\boldsymbol{A}_k,\boldsymbol{b}_k),\\ \boldsymbol{b}_{k+1} &= \boldsymbol{b}_k - t_{\boldsymbol{b}}\nabla_{\boldsymbol{b}}\varphi^{\mathrm{inv}}_{L^2,\sigma}(\boldsymbol{A}_k,\boldsymbol{b}_k)\end{aligned}$$

with smoothing

$$\sigma^2 \geq 2\frac{\max_i \|\boldsymbol{u}_i\|_2^2}{s_{\min}(\boldsymbol{U})^2}\left(s_{\max}(\boldsymbol{U})^2\|\boldsymbol{A}_\star - \boldsymbol{I}\|_F^2 + c\|\boldsymbol{b}_\star\|_2^2\right) \tag{16}$$

and step sizes

$$\begin{aligned}t_{\boldsymbol{A}} &= \frac{8\pi c\sigma^4}{s_{\max}(\boldsymbol{U})^2},\\ t_{\boldsymbol{b}} &= 8\pi\sigma^4,\end{aligned} \tag{17}$$

from initialization $\boldsymbol{A}_0 = \boldsymbol{I}, \boldsymbol{b}_0 = \boldsymbol{0}$ satisfies

$$\frac{8\pi\sigma^4}{t_{\boldsymbol{A}}}\|\boldsymbol{A}_k - \boldsymbol{A}_\star\|_F^2 + \|\boldsymbol{b}_k - \boldsymbol{b}_\star\|_2^2 \leq \left(1 - \frac{1}{2\kappa}\right)^{2k}\left(\frac{8\pi\sigma^4}{t_{\boldsymbol{A}}}\|\boldsymbol{I} - \boldsymbol{A}_\star\|_F^2 + \|\boldsymbol{b}_\star\|_2^2\right), \tag{18}$$

where

$$\kappa = \frac{s_{\max}(\boldsymbol{U})^2}{s_{\min}(\boldsymbol{U})^2},$$

with, $s_{\min}(\boldsymbol{U})$ and $s_{\max}(\boldsymbol{U})$ denoting the minimum and maximum singular values of the matrix $\boldsymbol{U}$.

Theorem 5.1 establishes a global linear rate of convergence for the continuum occurrence map registration formulation (15) in the relevant product norm, where the rate depends on the condition number of the matrix of observed spike locations $\boldsymbol{U}$. This dependence arises from the intuitive fact that *recovery* of the transformation parameters $(\boldsymbol{A}_\star, \boldsymbol{b}_\star)$ is a more challenging problem than registering the observation to the motif—in practice, we do not observe significant degradation of the ability to rapidly register the observed scene as the condition number increases. The proof of Theorem 5.1 reveals that the use of inverse parameterization in (15) dramatically improves the landscape of optimization: the problem becomes strongly convex around the true parameters when the smoothing level is set appropriately. In particular, (16) suggests a level of smoothing commensurate with the maximum distance the spikes need to travel for a successful registration, and (17) suggests larger step sizes for larger smoothing levels, with appropriate scaling of the step size on the $\boldsymbol{A}$ parameters to account for the larger motions experienced by objects further from the origin. In the proof, the 'centered locations' assumption $\boldsymbol{U}\boldsymbol{1} = \boldsymbol{0}$ allows us to obtain a global linear rate of convergence in both the $\boldsymbol{A}$ and $\boldsymbol{b}$ parameters. This is not a restrictive assumption, as in practice it is always possible to center the spike scene (e.g., by computing its center of mass and subtracting), and we also find it to accelerate convergence empirically when it is applied.

5.2 Experimental Verification

To verify the practical implications of Theorem 5.1, which is formulated in the continuum, we conduct numerical experiments on registering affine-transformed multichannel spike images using the discrete formulation (10). We implement a proximal gradient descent solver for (10), and use it to register randomly-transformed occurrence maps, as visualized in Figure 4(a-b). We set the step sizes and level of smoothing in accordance with (16) and (17), with a complementary smoothing value of $\sigma_0 = 3$. Figure 4 shows representative results taken from one such run: the objective value rapidly converges to near working precision, and the normalized cross-correlation between the transformed scene and the motif rapidly reaches a value of 0.972. This rapid convergence implies the formulation (10) is a suitable base for an unrolled architecture with mild depth, and is a direct consequence of the robust step size prescription offered by Theorem 5.1. Figure 4(f) plots the left-hand and right-hand sides of the parameter error bound (18) to evaluate its applicability to the discretized formulation: we observe an initial faster-than-predicted linear rate, followed by saturation at a suboptimal value. This gap is due to the difference between the continuum theory of Theorem 5.1 and practice: in the discretized setting, interpolation errors and finite-resolution artifacts prevent subpixel-perfect registration of the parameters, and hence exact recovery of the transformation $(\boldsymbol{A}_\star, \boldsymbol{b}_\star)$. In practice, successful registration of the spike scene, as demonstrated by Figure 4(e), is sufficient for applications, as in the networks we develop for hierarchical detection in Section 4.

6 Basin of Attraction for Textured Motif Registration with (4)

The theory and experiments we have presented in Section 5 justify the use of local optimization for alignment of spiky motifs. In this section, we provide additional corroboration beyond the experiment of Figure 1 of the efficacy of our textured motif registration formulation (4), under euclidean and similarity motion models. To this end, in Figure 5 we empirically evaluate the basin of attraction of a suitably-configured solver for registration of the crab body motif from Figure 2 with this formulation. Two-dimensional search grids are generated for each of the two setups as shown in the figure. For each given pair of transformation parameters, a similar multi-scale scheme over σ as in the above complexity experiment is used, starting at $\sigma = 10$ and step size 0.05, and halved every 50 iterations. The process terminates after a total of 250 iterations. The final ZNCC calculated over the motif support is reported, and the figure plots the average over 10 independent runs, where the background image is randomly generated for each pair of parameters in each run. The ZNCC ranges from 0 to 1, with a value of 1 implying equality of the channel-mean-subtracted motif and transformed image content over the corresponding support (up to scale).

Panels (a) and (b) of Figure 5 show that the optimization method tends to succeed unconditionally up to moderate amounts of transformation. For larger sets of transformations, it is important to first appropriately center the image, which will significantly improve the optimization performance. In practice, one may use a combination of optimization and a small number of covering, so that the entire transformation space is covered by the union of the basins of attractions. We note that irregularity near the edges, especially in panel (a), can be attributed in part due to the randomness in the background embedding, and in this sense the size of the basin in these results conveys a level of performance across a range of simulated operating conditions. In general, these basins are also motif-dependent: we would expect these results to change if we were testing with the eye motif from Figure 3(a), for example. A notable phenomenon in Figure 5(b), where translation is varied against scale, is the lack of a clear-cut boundary of the basin at small scales. This is due to the effect illustrated in Figure 5(c-d), where interpolation artifacts corrupt the motif when it is 'zoomed out' by optimization over deformations, and hence registration can never achieve a ZNCC close to 1. For applications where perfect reconstruction is not required, such as the hierarchical detection task studied in Section 4, these interpolation artifacts will not hinder the ability to localize the motif in the scene at intermediate scales, and if the basin were generated with a success metric other than ZNCC, a better-defined boundary to the basin would emerge.

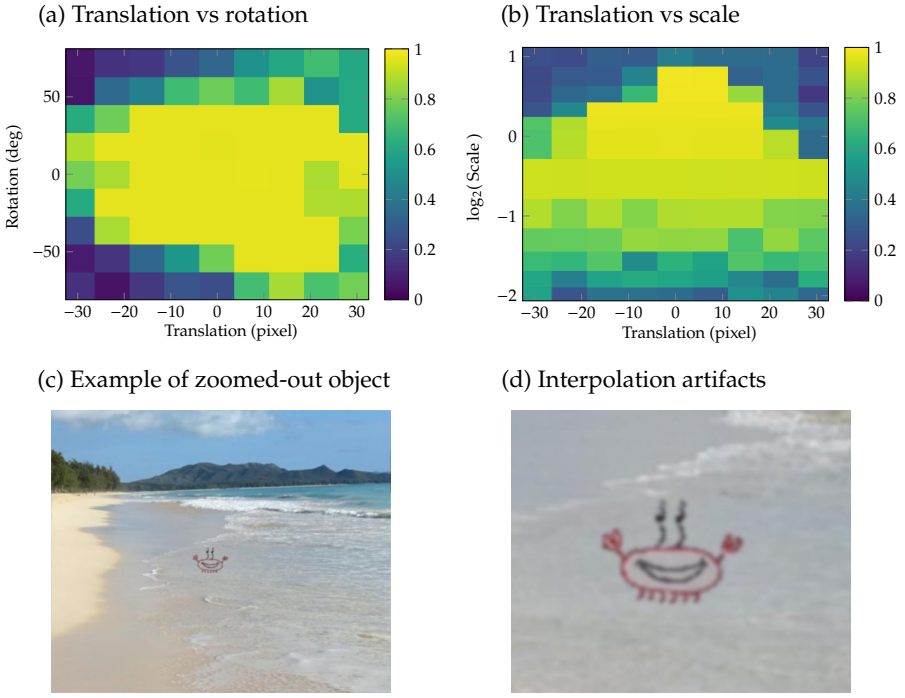

Figure 5: Plotting a basin of attraction for the textured motif registration formulation (4). **(a):** Heatmap of the ZNCC at convergence (see Appendix A.3), for translation versus rotation. Optimization conducted with SE(2) motion model. **(b):** Heatmap of the ZNCC at convergence, for translation versus scale. Optimization conducted in 'similarity mode', a SE(2) motion model with an extra global scale parameter. In both experiments, each reported data point is averaged over 10 independent runs. **(c-d):** Notably, when the registration target $\boldsymbol{y}$ is zoomed out relative to the motif $\boldsymbol{x}_o$, resolution is lost in the detection target, so recovering it will cause interpolation artifacts and blur the image. This prevents the ZNCC value from converging to 1 despite correct alignment with the motif, and accounts for the results shown in (b) at small scales.

7 Discussion

In this paper, we have taken initial steps towards realizing the potential of optimization over transformations of domain as an approach to achieve resource-efficient invariance with visual data. Below, we discuss several important future directions for the basic framework we have developed.

Statistical variability and complex tasks. To build invariant networks for complex visual tasks and real-world data beyond matching against fixed templates $\boldsymbol{x}_o$, it will be necessary to incorporate more refined appearance models for objects, such as a sparse dictionary model or a deep generative model [BDS19; DTLW+21; SSKK+21], and train the resulting hybrid networks in an end-to-end fashion. The invariant architectures we have designed in this work naturally plug into such a framework, and will allow for investigations similar to what we have developed in Section 4 into challenging tasks with additional structure (e.g., temporal or 3D data). Coping with the more complex motion models in these applications will demand regularizers ρ for our general optimization formulation (1) that go beyond parametric constraints.

Theory for registration of textured motifs in visual clutter. Our experiments in Section 5 have demonstrated the value that theoretical studies of optimization formulations have with respect to the design of the corresponding unrolled networks. Extending our theory for spiky motif registration to more general textured motifs will enable similar insights into the roles played by the various problem parameters in a formulation like (4) with respect to texture and shape properties of visual data and the clutter present, and

allow for similarly resource-efficient architectures to be derived in applications like the hierarchical template detection task we have developed in Section 4.3.

Hierarchical detection networks in real-time. The above directions will enable the networks we have demoed for hierarchical detection in Section 4 to scale to more general data models. At the same time, there are promising directions to improve the efficiency of the networks we design for a task like this one at the modeling level. For example, the networks we design in Section 4.3 essentially operate in a 'sliding window' fashion, without sharing information across the strides $\boldsymbol{\lambda}$, and they perform registration and detection separately. An architecture developed around an integrated approach to registration and detection, possibly building off advances in convolutional sparse modeling [QLZ19; KZLW20; LQKZ+20], may lead to further efficiency gains and push the resulting network closer to real-time operation capability.

Acknowledgments

This work was supported by the National Science Foundation through grants NSF 1733857, NSF 1838061, NSF 1740833, and NSF 2112085, and by a fellowship award (SB) through the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program, sponsored by the Air Force Research Laboratory (AFRL), the Office of Naval Research (ONR) and the Army Research Office (ARO). The authors thank Mariam Avagyan, Ben Haeffele, and Yi Ma for helpful discussions and feedback.

References

[SW71] Elias M Stein and Guido Weiss. *Introduction to Fourier Analysis on Euclidean Spaces*. Princeton University Press, 1971.

[HS81] Berthold K P Horn and Brian G Schunck. "Determining optical flow". *Artif. Intell.* 17.1 (Aug. 1981), pp. 185–203.

[Key81] R Keys. "Cubic convolution interpolation for digital image processing". *IEEE Trans. Acoust.* 29.6 (Dec. 1981), pp. 1153–1160.

[LK81] Bruce D Lucas and Takeo Kanade. "An iterative image registration technique with an application to stereo vision". *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*. IJCAI'81. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., Aug. 1981, pp. 674–679.

[Ana89] P Anandan. "A computational framework and an algorithm for the measurement of visual motion". *Int. J. Comput. Vis.* 2.3 (Jan. 1989), pp. 283–310.

[Bro92] Lisa Gottesfeld Brown. "A survey of image registration techniques". *ACM Comput. Surv.* 24.4 (Dec. 1992), pp. 325–376.

[CET98] T F Cootes, G J Edwards, and C J Taylor. "Active appearance models". *Computer Vision — ECCV'98*. Springer Berlin Heidelberg, 1998, pp. 484–498.

[MV98] J B Antoine Maintz and Max A Viergever. "A survey of medical image registration". *Med. Image Anal.* 2.1 (Mar. 1998), pp. 1–36.

[LC01] Martin Lefébure and Laurent D Cohen. "Image Registration, Optical Flow and Local Rigidity". *J. Math. Imaging Vis.* 14.2 (Mar. 2001), pp. 131–147.

[BM04] Simon Baker and Iain Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework". *Int. J. Comput. Vis.* 56.3 (Feb. 2004), pp. 221–255.

[Sze07] Richard Szeliski. "Image Alignment and Stitching: A Tutorial". *Foundations and Trends® in Computer Graphics and Vision* 2.1 (2007), pp. 1–104.

[FMR08] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. "A discriminatively trained, multiscale, deformable part model". *2008 IEEE Conference on Computer Vision and Pattern Recognition*. June 2008, pp. 1–8.

[HSS08] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA, 2008, pp. 11–15.

[BS10] Leah Bar and Guillermo Sapiro. "Hierarchical dictionary learning for invariant classification". *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. ieeexplore.ieee.org, Mar. 2010, pp. 3578–3581.

[FGMR10] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. "Object detection with discriminatively trained part-based models". *IEEE Trans. Pattern Anal. Mach. Intell.* 32.9 (Sept. 2010), pp. 1627–1645.

[GL10] Karol Gregor and Yann LeCun. "Learning fast approximations of sparse coding". *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 399–406.

[JMOB10] Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, and Francis R Bach. "Proximal Methods for Sparse Hierarchical Dictionary Learning". *ICML*. 2010, pp. 487–494.

[HKW11] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. "Transforming Auto-Encoders". *Artificial Neural Networks and Machine Learning – ICANN 2011*. Springer Berlin Heidelberg, 2011, pp. 44–51.

[CMS12] Dan Cireşan, Ueli Meier, and Juergen Schmidhuber. "Multi-column deep neural networks for image classification". *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2012, pp. 3642–3649.

[Mal12] Stéphane Mallat. "Group Invariant Scattering". *Commun. Pure Appl. Math.* 65.10 (Oct. 2012), pp. 1331–1398.

[MZM12] Hossein Mobahi, C Lawrence Zitnick, and Yi Ma. "Seeing through the blur". *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 1736–1743.

[SL12] Kihyuk Sohn and Honglak Lee. "Learning invariant representations with local transformations". *Proceedings of the 29th International Coference on International Conference on Machine Learning*. ICML'12. Edinburgh, Scotland: Omnipress, June 2012, pp. 1339–1346.

[SM12] Charles Sutton and Andrew McCallum. "An Introduction to Conditional Random Fields". *Foundations and Trends® in Machine Learning* 4.4 (2012), pp. 267–373.

[BM13] Joan Bruna and Stéphane Mallat. "Invariant scattering convolution networks". *IEEE Trans. Pattern Anal. Mach. Intell.* 35.8 (Aug. 2013), pp. 1872–1886.

[KSJ14] Angjoo Kanazawa, Abhishek Sharma, and David Jacobs. "Locally Scale-Invariant Convolutional Neural Networks" (Dec. 2014). arXiv: 1412.5104 [cs.CV].

[KF14] Sofia Karygianni and Pascal Frossard. "Tangent-based manifold approximation with locally linear models". *Signal Processing* 104 (2014), pp. 232–247.

[PB14] Neal Parikh and Stephen Boyd. "Proximal Algorithms". *Foundations and Trends® in Optimization* 1.3 (2014), pp. 127–239.

[VF14] Elif Vural and Pascal Frossard. "Analysis of Image Registration with Tangent Distance". *SIAM J. Imaging Sci.* 7.4 (Jan. 2014), pp. 2860–2915.

[DWD15] Sander Dieleman, Kyle W. Willett, and Joni Dambre. "Rotation-invariant convolutional neural networks for galaxy morphology prediction". *Monthly Notices of the Royal Astronomical Society* 450.2 (Apr. 2015), pp. 1441–1459.

[FF15] Alhussein Fawzi and Pascal Frossard. "Manitest: Are classifiers really invariant?" *British Machine Vision Conference (BVMC)*. 2015.

[GIDM15] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. "Deformable part models are convolutional neural networks". *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. openaccess.thecvf.com, 2015, pp. 437–446.

[JSZK15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. "Spatial Transformer Networks". *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 2017–2025.

[PVGR15] Marco Pedersoli, Andrea Vedaldi, Jordi Gonzàlez, and Xavier Roca. "A coarse-to-fine approach for fast deformable object detection". *Pattern Recognition* 48.5 (May 2015), pp. 1844–1853.

[ZMH15] Maxim Zaitsev, Julian Maclaren, and Michael Herbst. "Motion artifacts in MRI: A complex problem with many partial solutions". *J. Magn. Reson. Imaging* 42.4 (Oct. 2015), pp. 887–901.

[CW16] Taco Cohen and Max Welling. "Group Equivariant Convolutional Networks". *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 2990–2999.

[LSBP16] Dmitry Laptev, Nikolay Savinov, Joachim M. Buhmann, and Marc Pollefeys. "TI-Pooling: Transformation-Invariant Pooling for Feature Learning in Convolutional Neural Networks". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[BJ17] Ronen Basri and David W Jacobs. "Efficient Representation of Low-Dimensional Manifolds using Deep Networks". *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[DQXL+17] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. "Deformable convolutional networks". *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 764–773.

[KKHP17] Erich Kobler, Teresa Klatzer, Kerstin Hammernik, and Thomas Pock. "Variational Networks: Connecting Variational Methods and Deep Learning". *Pattern Recognition*. Lecture Notes in Computer Science. Springer, Cham, Sept. 2017, pp. 281–293.

[OBZ17] Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. "Scaling the Scattering Transform: Deep Hybrid Networks". *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[SFH17] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic Routing Between Capsules". *NIPS*. 2017, pp. 3859–3869.

[SER17] Max Simchowitz, Ahmed El Alaoui, and Benjamin Recht. "On the Gap Between Strict-Saddles and True Convexity: An Omega(log d) Lower Bound for Eigenvector Approximation" (Apr. 2017). arXiv: 1704.04548 [cs.LG].

[WGTB17] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. "Harmonic Networks: Deep Translation and Rotation Equivariance". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[BHKW18] S Buchanan, T Haque, P Kinget, and J Wright. "Efficient Model-Free Learning to Overcome Hardware Nonidealities in Analog-to-Information Converters". *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Apr. 2018, pp. 3574–3578.

[CLWY18] Xiaohan Chen, Jialin Liu, Zhangyang Wang, and Wotao Yin. "Theoretical Linear Convergence of Unfolded ISTA and Its Practical Weights and Thresholds". *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada, 2018, pp. 9079–9089.

[CGKW18] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. "Spherical CNNs". *International Conference on Learning Representations*. 2018.

[KMF18] C Kanbak, S Moosavi-Dezfooli, and P Frossard. "Geometric Robustness of Deep Networks: Analysis and Improvement". *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2018, pp. 4441–4449.

[KT18] Risi Kondor and Shubhendu Trivedi. "On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups". *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 2747–2755.

[WB18] Thomas Wiatowski and Helmut Bölcskei. "A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction". *IEEE Trans. Inf. Theory* 64.3 (Mar. 2018), pp. 1845–1866.

[XZLH+18] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. "Spatially Transformed Adversarial Examples". *International Conference on Learning Representations*. 2018.

[AAG19] Rima Alaifari, Giovanni S Alberti, and Tandri Gauksson. "ADef: an Iterative Algorithm to Construct Adversarial Deformations". *International Conference on Learning Representations*. 2019.

[ALGW+19] Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. "Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects". *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019.

[AW19] Aharon Azulay and Yair Weiss. "Why do deep convolutional networks generalize so poorly to small image transformations?" *Journal of Machine Learning Research* 20.184 (2019), pp. 1–25.

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". *International Conference on Learning Representations*. 2019.

[CJLZ19] Minshuo Chen, Haoming Jiang, Wenjing Liao, and Tuo Zhao. "Nonparametric Regression on Low-Dimensional Manifolds using Deep ReLU Networks" (Aug. 2019). arXiv: 1908.01842 [cs.LG].

[CZMV+19] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. "AutoAugment: Learning Augmentation Strategies From Data". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[ETTS+19] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. "Exploring the Landscape of Spatial Robustness". *International Conference on Machine Learning*. 2019, pp. 1802–1811.

[GBW19] Dar Gilboa, Sam Buchanan, and John Wright. "Efficient Dictionary Learning with Gradient Descent". *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, 2019, pp. 2252–2259.

[KZFM19] Angjoo Kanazawa, Jason Y. Zhang, Panna Felsen, and Jitendra Malik. "Learning 3D Human Dynamics From Video". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[KBCW19] Michael Kellman, Emrah Bostan, Michael Chen, and Laura Waller. "Data-Driven Design for Fourier Ptychographic Microscopy". *2019 IEEE International Conference on Computational Photography (ICCP)*. 2019, pp. 1–8.

[KBRW19] Michael R. Kellman, Emrah Bostan, Nicole A. Repina, and Laura Waller. "Physics-Based Learned Design: Optimized Coded-Illumination for Quantitative Phase Imaging". *IEEE Transactions on Computational Imaging* 5.3 (2019), pp. 344–353.

[LCWY19] Jialin Liu, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. "ALISTA: Analytic Weights Are As Good As Learned Weights in LISTA". *International Conference on Learning Representations*. 2019.

[PGML+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". *Advances in Neural Information Processing Systems 32*. 2019, pp. 8024–8035.

[QLZ19] Qing Qu, Xiao Li, and Zhihui Zhu. "A Nonconvex Approach for Exact and Efficient Multichannel Sparse Blind Deconvolution". *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[Sch19] Johannes Schmidt-Hieber. "Deep ReLU network approximation of functions on a manifold" (Aug. 2019). arXiv: 1908.00695 [stat.ML].

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1597–1607.

[HMCZ+20] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. "Augmix: A simple method to improve robustness and uncertainty under data shift". *International conference on learning representations*. Vol. 1. 2020, p. 6.

[KZLW20] Han-Wen Kuo, Yuqian Zhang, Yenson Lau, and John Wright. "Geometry and Symmetry in Short-and-Sparse Deconvolution". *SIAM Journal on Mathematics of Data Science* 2.1 (Jan. 2020), pp. 216–245.

[LQKZ+20] Yenson Lau, Qing Qu, Han-Wen Kuo, Pengcheng Zhou, Yuqian Zhang, and John Wright. "Short and Sparse Deconvolution — A Geometric Approach". *International Conference on Learning Representations*. 2020.

[NI20] Ryumei Nakada and Masaaki Imaizumi. "Adaptive Approximation and Generalization of Deep Neural Network with Intrinsic Dimensionality". *J. Mach. Learn. Res.* 21.174 (2020), pp. 1–38.

[OJMB+20] Gregory Ongie, Ajil Jalal, Christopher A Metzler, Richard G Baraniuk, Alexandros G Dimakis, and Rebecca Willett. "Deep Learning Techniques for Inverse Problems in Imaging". *IEEE Journal on Selected Areas in Information Theory* 1.1 (May 2020), pp. 39–56.

[ZTAM20] John Zarka, Louis Thiry, Tomas Angles, and Stephane Mallat. "Deep Network Classification by Scattering and Homotopy Dictionary Learning". *International Conference on Learning Representations*. 2020.

[BBCV21] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges" (Apr. 2021). arXiv: 2104.13478 [cs.LG].

[CDHS21] Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. "Lower bounds for finding stationary points II: first-order methods". *Math. Program.* 185.1 (Jan. 2021), pp. 315–355.

[CCCH+21] Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. "Learning to Optimize: A Primer and A Benchmark" (Mar. 2021). arXiv: 2103.12828 [math.OC].

[CK21] Alexander Cloninger and Timo Klock. "A deep network construction that adapts to intrinsic dimensionality beyond the domain". *Neural Networks* 141 (2021), pp. 404–419.

[DTLW+21] Xili Dai, Shengbang Tong, Mingyang Li, Ziyang Wu, Kwan Ho Ryan Chan, Pengyuan Zhai, Yaodong Yu, Michael Psenka, Xiaojun Yuan, Heung Yeung Shum, and Yi Ma. "Closed-Loop Data Transcription to an LDR via Minimaxing Rate Reduction" (Nov. 2021). arXiv: 2111.06636 [cs.CV].

[Hin21] Geoffrey Hinton. "How to represent part-whole hierarchies in a neural network" (Feb. 2021). arXiv: 2102.12627 [cs.CV].

[JL21] Ylva Jansson and Tony Lindeberg. "Scale-invariant scale-channel networks: Deep networks that generalise to previously unseen scales". *CoRR* abs/2106.06418 (2021). arXiv: 2106.06418.

[MRRK+21] Matthew J. Muckley, Bruno Riemenschneider, Alireza Radmanesh, Sunwoo Kim, Geunu Jeong, Jingyu Ko, Yohan Jun, Hyungseob Shin, Dosik Hwang, Mahmoud Mostapha, Simon Arberet, Dominik Nickel, Zaccharie Ramzi, Philippe Ciuciu, Jean-Luc Starck, Jonas Teuwen, Dimitrios Karkalousos, Chaoping Zhang, Anuroop Sriram, Zhengnan Huang, Nafissa Yakubova, Yvonne W. Lui, and Florian Knoll. "Results of the 2020 fastMRI Challenge for Machine Learning MR Image Reconstruction". *IEEE Transactions on Medical Imaging* 40.9 (2021), pp. 2306–2317.

[SSKK+21] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-Based Generative Modeling through Stochastic Differential Equations". *International Conference on Learning Representations*. 2021.

[STDS+21] Weiwei Sun, Andrea Tagliasacchi, Boyang Deng, Sara Sabour, Soroosh Yazdani, Geoffrey E Hinton, and Kwang Moo Yi. "Canonical Capsules: Self-Supervised Capsules in Canonical Pose". *Advances in Neural Information Processing Systems* 34 (2021).

[WFVW21] Maurice Weiler, Patrick Forré, Erik Verlinde, and Max Welling. "Coordinate Independent Convolutional Networks – Isometry and Gauge Equivariant Convolutions on Riemannian Manifolds" (June 2021). arXiv: 2106.06020 [cs.LG].

[ZGM21] John Zarka, Florentin Guth, and Stéphane Mallat. "Separation and Concentration in Deep Networks". *International Conference on Learning Representations*. 2021.

A Implementation and Experimental Details

A.1 Implementation Details for Parametric Transformations of the Image Plane

Our implementation of parametric image deformations revolves around the specific definition of interpolation we have made:

$$\boldsymbol{y} \circ \boldsymbol{\tau} = \sum_{(k,l)\in\mathbb{Z}^2} y_{kl}\phi(\boldsymbol{\tau}_0 - k\mathbf{1}) \odot \phi(\boldsymbol{\tau}_1 - l\mathbf{1}),$$

and the identification of the image $\boldsymbol{y} \in \mathbb{R}^{m\times n}$ with a function on $\mathbb{Z}^2$ with support in $\{0,\dots,m-1\}\times\{0,\dots,n-1\}$.[5] Although we use the notation $\circ$ for interpolation in analogy with the usual notation for composition of functions, this operation is significantly less well-structured: although we *can* define interpolation of motion fields $\boldsymbol{\tau}_0 \circ \boldsymbol{\tau}_1$, it is impossible in general to even have associativity of $\circ$ (let alone inverses), so that in general $(\boldsymbol{x} \circ \boldsymbol{\tau}_0) \circ \boldsymbol{\tau}_1 \neq \boldsymbol{x} \circ (\boldsymbol{\tau}_0 \circ \boldsymbol{\tau}_1)$. This failure is intimately linked to the existence of parasitic interpolation artifacts when computing and optimizing with interpolated images, which we go to great lengths to avoid in our experiments. On the other hand, there does exist a well-defined identity vector field: from our definitions, we can read off the canonical definition of the identity transformation, from which definitions for other parametric transformations we consider here follow. Defining (with a slight abuse of notation)

$$\boldsymbol{m} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ m-1 \end{bmatrix}; \quad \boldsymbol{n} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ n-1 \end{bmatrix},$$

we have from the definition of the cubic convolution interpolation kernel ϕ that

$$\boldsymbol{y} \circ (\boldsymbol{m}\mathbf{1}^* \otimes \boldsymbol{e}_0 + \mathbf{1}\boldsymbol{n}^* \otimes \boldsymbol{e}_1) = \boldsymbol{y}.$$

One can then check that the following linear embedding of the affine transformations, which we will write as $\mathrm{Aff}(2) = \mathrm{GL}(2)\times\mathbb{R}^2$, leads to the natural vector field analogue of affine transformations on the continuum $\mathbb{R}^2$ (c.f. Appendix B):

$$\mathrm{Aff}(2) \cong \mathrm{span}\{\boldsymbol{m}\mathbf{1}^* \otimes \boldsymbol{e}_0, \mathbf{1}\boldsymbol{n}^* \otimes \boldsymbol{e}_0, \boldsymbol{m}\mathbf{1}^* \otimes \boldsymbol{e}_1, \mathbf{1}\boldsymbol{n}^* \otimes \boldsymbol{e}_1, \mathbf{1}_{m,n} \otimes \boldsymbol{e}_0, \mathbf{1}_{m,n} \otimes \boldsymbol{e}_1\}. \tag{19}$$

Of course, these vector fields can be any size—they need not match the size of the image. As we mention in Section 3.3, we always initialize our networks with the identity transform; in the basis above, this corresponds to the vector $(1,0,0,1,0,0)$ (i.e., this is like a row-wise flattening of the affine transform's matrix $\boldsymbol{A} \in \mathrm{GL}(2)$, concatenated with $\boldsymbol{b}$).

Next we turn to computation of the proximal operator, which we need for unrolling (see Section 3.3). Given (6) and the fact that (19) is a subspace, we can compute the proximal operator for Aff(2) given an orthonormal basis for Aff(2). It is then unfortunate that the natural basis vectors that we have used in the expression (19) are not orthogonal: we have $\langle \boldsymbol{m}\mathbf{1}^*, \mathbf{1}\boldsymbol{n}^*\rangle = \langle \boldsymbol{m}, \mathbf{1}\rangle\langle \boldsymbol{n}, \mathbf{1}\rangle \gg 0$, for example. To get around this, in practice we apply a technique we refer to as *centering* of transformations. Indeed, notice that for any $\boldsymbol{c} \in \mathbb{R}^2$, we have

$$\begin{aligned}\mathrm{Aff}(2) \cong \mathrm{span}\{&(\boldsymbol{m} - c_0\mathbf{1})\mathbf{1}^* \otimes \boldsymbol{e}_0, \mathbf{1}(\boldsymbol{n} - c_1\mathbf{1})^* \otimes \boldsymbol{e}_0, (\boldsymbol{m} - c_0\mathbf{1})\mathbf{1}^* \otimes \boldsymbol{e}_1, \mathbf{1}(\boldsymbol{n} - c_1\mathbf{1})^* \otimes \boldsymbol{e}_1, \mathbf{1}_{m,n} \otimes \boldsymbol{e}_0, \mathbf{1}_{m,n} \otimes \boldsymbol{e}_1\} \\ &+ \mathbf{1}_{m,n} \otimes \boldsymbol{c}.\end{aligned} \tag{20}$$

In the continuum, applying an affine transform in this way corresponds to the mapping $\boldsymbol{x} \mapsto \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{c}) + \boldsymbol{b} + \boldsymbol{c}$, hence the name: the image plane is shifted to have its origin at $\boldsymbol{c}$ for the purposes of applying the transform. When we implement affine transforms as suggested by (20), we choose $\boldsymbol{c}$ to make the basis vectors orthogonal

[5] These conventions are not universal, although they seem most natural from a mathematical standpoint—for example, PyTorch thinks of its images as lying on a grid in the square $[-1,+1]\times[-1,+1]$ instead, with spacing and offsets depending on the image resolution and other implementation-specific options. In our released code, we handle conversion from our notation to this notation.

this necessitates that $\boldsymbol{c} = ((m-1)/2, (n-1)/2)$. Then we are able to write down a concrete expression for the projection operator in these coordinates:[6]

$$\mathrm{proj}_{\mathrm{Aff}(2)}(\boldsymbol{\tau}) = \left((\boldsymbol{m} - \tfrac{m-1}{2}\mathbf{1})^* \boldsymbol{\tau}_0 \mathbf{1}, \mathbf{1}^* \boldsymbol{\tau}_0 (\boldsymbol{n} - \tfrac{n-1}{2}\mathbf{1}), (\boldsymbol{m} - \tfrac{m-1}{2}\mathbf{1})^* \boldsymbol{\tau}_1 \mathbf{1}, \mathbf{1}^* \boldsymbol{\tau}_1 (\boldsymbol{n} - \tfrac{n-1}{2}\mathbf{1}), \mathbf{1}^* \boldsymbol{\tau}_0 \mathbf{1}, \mathbf{1}^* \boldsymbol{\tau}_1 \mathbf{1}\right). \tag{21}$$

The low-rank structure of the basis vectors implies that this transformation can be computed quite rapidly. Although it may seem we have undertaken this discussion for the sake of mathematical rigor, in our experiments we observe significant computational benefits to centering by the prescription above. For example, when computing with (10), using a non-orthogonal basis for the affine transforms (or a center that is not at the center of the region being transformed) often leads to skewing artifacts in the final transformation recovered. We also notice slower convergence.

Finally, for our experiments in Section 4 with the rigid motion model SE(2), some additional discussion is required. This is because the orthogonal transformations SO(2) are not a linear subspace, like the affine transforms (19), but a smooth manifold (diffeomorphic to a circle). For these transformations, we modify the formula (1) by differentiating in a parameterization of SE(2): concretely, we use

$$\mathrm{SO}(2) \cong \left\{ \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \middle| \, \theta \in [0, 2\pi] \right\}.$$

Writing $F : \mathbb{R} \to \mathbb{R}^{m \times n \times 2}$ for this parameterization composed with our usual vector field representation (20) for subgroups of the affine transforms, we modify the objective (1) to be $\min_\theta \varphi(\boldsymbol{y} \circ F(\theta))$. A simple calculation then shows that gradients in this parameterization are obtainable from gradients with respect to the affine parameterization as

$$\nabla_\theta [\varphi(\boldsymbol{y} \circ F)](\theta) = \left\langle \nabla_A [\varphi(\boldsymbol{y} \circ \boldsymbol{\tau}_{\cdot,b})] \left(\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \right), \begin{bmatrix} -\sin\theta & -\cos\theta \\ \cos\theta & -\sin\theta \end{bmatrix} \right\rangle.$$

This is a minor extra nonlinearity that replaces the proximal operation when we unroll networks as in (5) with this motion model. Gradients and projections with respect to the translation parameters are no different from the affine case.

A.2 Gradient Calculations for Unrolled Network Architectures

We collect in this section several computations relevant to gradients of the function φ (following the structure of (1)) in the optimization formulations (2), (3), (4) and (10).

$\boldsymbol{\tau}$ gradients. All of the costs we consider use the ℓ^2 error $\|\cdot\|_F$, so their gradient calculations with respect to $\boldsymbol{\tau}$ are very similar. We will demonstrate the gradient calculation for (2) to show how (6) is derived; the calculations for other costs follow the same type of argument. To be concise, we will write $\nabla_{\boldsymbol{\tau}} \varphi$ for the gradient with respect to $\boldsymbol{\tau}$ of the relevant costs $\varphi(\boldsymbol{y} \circ \boldsymbol{\tau})$.

Proposition A.1. *Let φ denote the $\|\cdot\|_F$ cost in (2). One has*

$$\nabla_{\boldsymbol{\tau}} \varphi(\boldsymbol{\tau}) = \sum_{k=0}^{c-1} \left(\boldsymbol{g}_{\sigma^2} * \mathcal{P}_\Omega \left[\boldsymbol{g}_{\sigma^2} * \left(\boldsymbol{y} \circ \boldsymbol{\tau} - \boldsymbol{x}_o \right)_k \right] \otimes \mathbf{1}_2 \right) \odot \left(\mathrm{d}\boldsymbol{y}_k \circ \boldsymbol{\tau} \right).$$

Proof. The cost separates over channels, so by linearity of the gradient it suffices to assume $c = 1$. We proceed by calculating the differential of $\varphi(\boldsymbol{y} \circ \boldsymbol{\tau})$ with respect to $\boldsymbol{\tau}$. We have for $\boldsymbol{\Delta}$ of the same shape as $\boldsymbol{\tau}$ and $t \in \mathbb{R}$

$$\left. \frac{\partial}{\partial t} \right|_{t=0} \varphi(\boldsymbol{y} \circ (\boldsymbol{\tau} + t\boldsymbol{\Delta})) = \left\langle \mathcal{P}_\Omega \left[\boldsymbol{g}_{\sigma^2} * (\boldsymbol{y} \circ \boldsymbol{\tau} - \boldsymbol{x}_o) \right] \otimes \mathbf{1}_2, \mathcal{P}_\Omega \left[\boldsymbol{g}_{\sigma^2} * \left((\mathrm{d}\boldsymbol{y} \circ \boldsymbol{\tau}) \odot \boldsymbol{\Delta} \right) \right] \right\rangle,$$

[6] In practice, our choice of step size is made to scale each element in this basis to be orthonormal (in particular, applying different steps to the matrix and translation parameters of the transformation)—strictly speaking the projection in (21) is not the orthogonal projection because this extra scaling has not been applied. We do not specify this scaling here because its optimal value often depends on the image content: for example, see the step size prescriptions in Theorem 5.1.

where $\mathrm{d}\boldsymbol{y} \in \mathbb{R}^{m\times n\times 2}$ is the Jacobian matrix of $\boldsymbol{y}$, defined as (here $\dot\phi$ denotes the derivative of the cubic convolution interpolation kernel ϕ)

$$\mathrm{d}\boldsymbol{y}_0 = \sum_{(k,l)\in\mathbb{Z}^2} y_{kl}\dot\phi(\boldsymbol{m}\mathbf{1}^* - k\mathbf{1}) \odot \phi(\mathbf{1}\boldsymbol{n}^* - l\mathbf{1}), \quad \mathrm{d}\boldsymbol{y}_1 = \sum_{(k,l)\in\mathbb{Z}^2} y_{kl}\phi(\boldsymbol{m}\mathbf{1}^* - k\mathbf{1}) \odot \dot\phi(\mathbf{1}\boldsymbol{n}^* - l\mathbf{1}),$$

and where for concision we are writing $\boldsymbol{g}_{\sigma^2} * \mathrm{d}\boldsymbol{y}$ to denote the filtering of each of the two individual channels of $\mathrm{d}\boldsymbol{y}$ by $\boldsymbol{g}_{\sigma^2}$. Using three adjoint relations ($\mathcal{P}_\Omega$ is an orthogonal projection, hence self-adjoint; the adjoint of convolution by $\boldsymbol{g}_{\sigma^2}$ is cross-correlation with $\boldsymbol{g}_{\sigma^2}$; elementwise multiplication is self-adjoint) and a property of the tensor product, the claim follows. □

Convolutional representation of cost-smoothed formulation (3). The cost-smoothed formulation (3) can be directly expressed as a certain convolution with $\boldsymbol{g}_{\sigma^2}$, leading to very fast convolution-free inner loops in gradient descent implementation. To see this, write

$$\begin{aligned}\left\|\mathcal{P}_\Omega\left[\boldsymbol{y}\circ(\boldsymbol{\tau}+\boldsymbol{\tau}_{\mathbf{0},\Delta}) - \boldsymbol{x}_o\right]\right\|_F^2 &= \left\|\mathcal{P}_\Omega\left[\boldsymbol{y}\circ(\boldsymbol{\tau}+\boldsymbol{\tau}_{\mathbf{0},\Delta})\right]\right\|_F^2 + \|\mathcal{P}_\Omega[\boldsymbol{x}_o]\|_F^2 + 2\left\langle\mathcal{P}_\Omega\left[\boldsymbol{y}\circ(\boldsymbol{\tau}+\boldsymbol{\tau}_{\mathbf{0},\Delta})\right],\mathcal{P}_\Omega[\boldsymbol{x}_o]\right\rangle\\ &= \left\langle\left[\boldsymbol{y}\circ(\boldsymbol{\tau}+\boldsymbol{\tau}_{\mathbf{0},\Delta})\right]^{\odot 2},\mathcal{P}_\Omega[\mathbf{1}]\right\rangle + \|\mathcal{P}_\Omega[\boldsymbol{x}_o]\|_F^2 + 2\left\langle\boldsymbol{y}\circ(\boldsymbol{\tau}+\boldsymbol{\tau}_{\mathbf{0},\Delta}),\mathcal{P}_\Omega[\boldsymbol{x}_o]\right\rangle,\end{aligned}$$

using self-adjointness of $\mathcal{P}_\Omega$ and the fact that it can be represented as an elementwise multiplication, and writing $[\,\cdot\,]^{\odot 2}$ for elementwise squaring. Thus, denoting the $\|\cdot\|_F$ cost in (3) by $\varphi(\boldsymbol{\tau})$, φ can be written as

$$\begin{aligned}2\varphi(\boldsymbol{\tau}) =& \left\langle\sum_\Delta(\boldsymbol{g}_{\sigma^2})_\Delta\left[\boldsymbol{y}\circ(\boldsymbol{\tau}+\boldsymbol{\tau}_{\mathbf{0},\Delta})\right]^{\odot 2},\mathcal{P}_\Omega[\mathbf{1}]\right\rangle + \left\langle\boldsymbol{g}_{\sigma^2},\mathbf{1}\right\rangle\|\mathcal{P}_\Omega[\boldsymbol{x}_o]\|_F^2\\ &+ 2\left\langle\sum_\Delta(\boldsymbol{g}_{\sigma^2})_\Delta\cdot\boldsymbol{y}\circ(\boldsymbol{\tau}+\boldsymbol{\tau}_{\mathbf{0},\Delta}),\mathcal{P}_\Omega[\boldsymbol{x}_o]\right\rangle.\end{aligned}$$

This can be expressed as a cross-correlation with $\boldsymbol{g}_{\sigma^2}$:

$$2\varphi(\boldsymbol{\tau}) = \left\langle\boldsymbol{g}_{\sigma^2} * \left[\boldsymbol{y}\circ\boldsymbol{\tau}\right]^{\odot 2},\mathcal{P}_\Omega[\mathbf{1}]\right\rangle + \left\langle\boldsymbol{g}_{\sigma^2},\mathbf{1}\right\rangle\|\mathcal{P}_\Omega[\boldsymbol{x}_o]\|_F^2 + 2\left\langle\boldsymbol{g}_{\sigma^2} * (\boldsymbol{y}\circ\boldsymbol{\tau}),\mathcal{P}_\Omega[\boldsymbol{x}_o]\right\rangle,$$

and taking adjoints gives finally

$$2\varphi(\boldsymbol{\tau}) = \left\langle\left[\boldsymbol{y}\circ\boldsymbol{\tau}\right]^{\odot 2},\boldsymbol{g}_{\sigma^2} * \mathcal{P}_\Omega[\mathbf{1}]\right\rangle + \left\langle\boldsymbol{g}_{\sigma^2},\mathbf{1}\right\rangle\|\mathcal{P}_\Omega[\boldsymbol{x}_o]\|_F^2 + 2\left\langle\boldsymbol{y}\circ\boldsymbol{\tau},\boldsymbol{g}_{\sigma^2} * \mathcal{P}_\Omega[\boldsymbol{x}_o]\right\rangle.$$

This gives a convolution-free gradient step implementation for this cost (aside from pre-computing the fixed convolutions in the cost), and also yields a useful interpretation of the cost-smoothed formulation (3), and its disadvantages relative to the background-modeled formulation (4).

Filter gradient for complementary smoothing formulation (10). Relative to the standard registration model formulation (2), the complementary smoothing spike registration formulation (10) contains an extra complicated transformation-dependent gaussian filter. We provide a key lemma below for the calculation of the gradient with respect to the parameters of the complementary smoothing cost in "standard parameterization" (see the next paragraph below). The full calculation follows the proof of Proposition A.1 with an extra "product rule" step and extra adjoint calculations.

Proposition A.2. *Given fixed $\sigma^2 > \sigma_0^2 > 0$, define $\boldsymbol{\Sigma}(\boldsymbol{A}) = \sigma^2\boldsymbol{I} - \sigma_0^2(\boldsymbol{A}^*\boldsymbol{A})^{-1}$, and define*

$$g(\boldsymbol{A}) = \sqrt{\det(\boldsymbol{A}^*\boldsymbol{A})}\,\boldsymbol{g}_{\boldsymbol{\Sigma}(\boldsymbol{A})},$$

where the filter is m by n and the domain is the open set $\{\boldsymbol{A} \mid \sigma\boldsymbol{I} - \sigma_0^2(\boldsymbol{A}^*\boldsymbol{A})^{-1} > \mathbf{0}\}$. Then for any fixed $\boldsymbol{V}\in\mathbb{R}^{m\times n}$, one has

$$\nabla_{\boldsymbol{A}}[\langle\boldsymbol{V},\boldsymbol{g}\rangle](\boldsymbol{A}) = \sigma_0^2\boldsymbol{A}^{-*}\left(\boldsymbol{\Sigma}(\boldsymbol{A})^{-1}\left(\sum_{i,j}V_{ij}g(\boldsymbol{A})_{ij}\boldsymbol{w}_{ij}\boldsymbol{w}_{ij}^*\right)\boldsymbol{\Sigma}(\boldsymbol{A})^{-1} - \langle\boldsymbol{g}(\boldsymbol{A}),\boldsymbol{V}\rangle\boldsymbol{\Sigma}(\boldsymbol{A})^{-1}\right)(\boldsymbol{A}^*\boldsymbol{A})^{-1} + \langle\boldsymbol{g}(\boldsymbol{A}),\boldsymbol{V}\rangle\boldsymbol{A}^{-*},$$

where $\boldsymbol{A}^{-*} = (\boldsymbol{A}^{-1})^*$.

Proof. For $(i,j) \in \{0,\dots,m-1\} \times \{0,\dots,n-1\}$, let $\boldsymbol{w}_{ij} = (i - \lfloor m/2 \rfloor, j - \lfloor n/2 \rfloor)$. Then we have

$$g(\boldsymbol{A}) = \frac{1}{2\pi} \sum_{i,j} \boldsymbol{e}_{ij} \exp\left(-\frac{1}{2} \boldsymbol{w}_{ij}^* \boldsymbol{\Sigma}(\boldsymbol{A})^{-1} \boldsymbol{w}_{ij} - \frac{1}{2} \log\det \Sigma(\boldsymbol{A}) + \frac{1}{2} \log\det \boldsymbol{A}^* \boldsymbol{A} \right).$$

Let $\mathrm{d}g$ denote the differential of $\boldsymbol{A} \mapsto g(\boldsymbol{A})$. By the chain rule, we have for any $\boldsymbol{\Delta} \in \mathbb{R}^{2\times 2}$

$$\left\langle \boldsymbol{V}, \mathrm{d}g_{\boldsymbol{A}}(\boldsymbol{\Delta}) \right\rangle = \frac{1}{2} \sum_{i,j} V_{ij} g(\boldsymbol{A})_{ij} \frac{\partial}{\partial t}\bigg|_{t=0} \left[-\boldsymbol{w}_{ij}^* \boldsymbol{\Sigma}(\boldsymbol{A} + t\boldsymbol{\Delta})^{-1} \boldsymbol{w}_{ij} - \log\det \Sigma(\boldsymbol{A} + t\boldsymbol{\Delta}) + \log\det(\boldsymbol{A} + t\boldsymbol{\Delta})^*(\boldsymbol{A} + t\boldsymbol{\Delta}) \right].$$

We need the differential of several mappings here. We will use repeatedly that if $\boldsymbol{X} \in \mathrm{GL}(2)$ and $\boldsymbol{W} \in \mathbb{R}^{2\times 2}$, one has

$$\mathrm{d}[\boldsymbol{X} \mapsto \left\langle \boldsymbol{W}, \boldsymbol{X}^{-1} \right\rangle]_{\boldsymbol{X}}(\boldsymbol{\Delta}) = -\langle \boldsymbol{\Delta}, \boldsymbol{X}^{-*} \boldsymbol{W} \boldsymbol{X}^{-*} \rangle. \tag{22}$$

Applying (22) and the chain rule, we get

$$\begin{aligned} \mathrm{d}[\langle \boldsymbol{W}, \boldsymbol{\Sigma} \rangle]_{\boldsymbol{A}}(\boldsymbol{\Delta}) &= \sigma_0^2 \left\langle (\boldsymbol{A}^* \boldsymbol{A})^{-1} \boldsymbol{W} (\boldsymbol{A}^* \boldsymbol{A})^{-1}, \boldsymbol{\Delta}^* \boldsymbol{A} + \boldsymbol{A}^* \boldsymbol{\Delta} \right\rangle \\ &= \sigma_0^2 \left\langle \boldsymbol{A}^{-*} (\boldsymbol{W} + \boldsymbol{W}^*) (\boldsymbol{A}^* \boldsymbol{A})^{-1}, \boldsymbol{\Delta} \right\rangle. \end{aligned} \tag{23}$$

In particular, using the chain rule and (23) and (22) gives

$$\frac{\partial}{\partial t}\bigg|_{t=0} \left[\boldsymbol{w}_{ij}^* \boldsymbol{\Sigma}(\boldsymbol{A} + t\boldsymbol{\Delta})^{-1} \boldsymbol{w}_{ij} \right] = -2\sigma_0^2 \left\langle \boldsymbol{A}^{-*} \boldsymbol{\Sigma}(\boldsymbol{A})^{-1} \boldsymbol{w}_{ij} \boldsymbol{w}_{ij}^* \boldsymbol{\Sigma}(\boldsymbol{A})^{-1} (\boldsymbol{A}^* \boldsymbol{A})^{-1}, \boldsymbol{\Delta} \right\rangle. \tag{24}$$

Next, using the Leibniz formula for the determinant, we obtain

$$\mathrm{d}[\log\det]_{\boldsymbol{X}}(\boldsymbol{\Delta}) = \langle \boldsymbol{X}^{-*}, \boldsymbol{\Delta} \rangle. \tag{25}$$

The chain rule and (23) and (25) thus give

$$\frac{\partial}{\partial t}\bigg|_{t=0} \left[\log\det \Sigma(\boldsymbol{A} + t\boldsymbol{\Delta}) \right] = 2\sigma_0^2 \left\langle \boldsymbol{A}^{-*} \boldsymbol{\Sigma}(\boldsymbol{A})^{-1} (\boldsymbol{A}^* \boldsymbol{A})^{-1}, \boldsymbol{\Delta} \right\rangle, \tag{26}$$

and similarly

$$\frac{\partial}{\partial t}\bigg|_{t=0} \left[\log\det(\boldsymbol{A} + t\boldsymbol{\Delta})^* (\boldsymbol{A} + t\boldsymbol{\Delta}) \right] = 2\langle \boldsymbol{A}^{-*}, \boldsymbol{\Delta} \rangle. \tag{27}$$

Combining (24), (26) and (27), we have

$$\left\langle \boldsymbol{V}, \mathrm{d}g_{\boldsymbol{A}}(\boldsymbol{\Delta}) \right\rangle = \sum_{i,j} V_{ij} g(\boldsymbol{A})_{ij} \left\langle \sigma_0^2 \boldsymbol{A}^{-*} \left(\boldsymbol{\Sigma}(\boldsymbol{A})^{-1} \boldsymbol{w}_{ij} \boldsymbol{w}_{ij}^* \boldsymbol{\Sigma}(\boldsymbol{A})^{-1} - \boldsymbol{\Sigma}(\boldsymbol{A})^{-1} \right) (\boldsymbol{A}^* \boldsymbol{A})^{-1} + \boldsymbol{A}^{-*}, \boldsymbol{\Delta} \right\rangle,$$

and the claim follows by distributing and reading off the gradient.[7] □

Differentiating costs in "inverse parameterization". Our theoretical study of spike alignment in Appendix B and our experiments on the discretized objective (10) in Section 5.2 suggest strongly to prefer "inverse parameterization" relative to standard parameterization of affine transformations for optimization. By this, we mean the following: given a cost $\varphi(\tau_{\boldsymbol{A},\boldsymbol{b}})$ optimized over affine transformations $(\boldsymbol{A}, \boldsymbol{b})$, one optimizes instead $\varphi(\tau_{\boldsymbol{A}^{-1}, -\boldsymbol{A}^{-1}\boldsymbol{b}})$. This nomenclature is motivated by, in the continuum, the inverse of the affine transformation $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ being $\boldsymbol{x} \mapsto \boldsymbol{A}^{-1}(\boldsymbol{x} - \boldsymbol{b})$. Below, we show the chain rule calculation that allows one to easily obtain gradients for inverse-parameterized objectives as linear corrections of the standard-parameterized gradients.

[7] After distributing, the sum over i, j in the first factor can be computed relatively efficiently using a Kronecker product.

Proposition A.3. *Let* $\varphi : \mathbb{R}^{2\times 2} \times \mathbb{R}^2 \to \mathbb{R}$, *and let* $F(\boldsymbol{A}, \boldsymbol{b}) = (\boldsymbol{A}^{-1}, -\boldsymbol{A}^{-1}\boldsymbol{b})$ *denote the inverse parameterization mapping, defined on* $\mathrm{GL}(2) \times \mathbb{R}^2$. *One has*

$$\nabla_{\boldsymbol{A}}[\varphi \circ F](\boldsymbol{A}, \boldsymbol{b}) = -\boldsymbol{A}^{-*}\left(\nabla_{\boldsymbol{A}}[\varphi] \circ F(\boldsymbol{A}, \boldsymbol{b})\right)\boldsymbol{A}^{-*} + \boldsymbol{A}^{-*}\left(\nabla_{\boldsymbol{b}}[\varphi] \circ F(\boldsymbol{A}, \boldsymbol{b})\right)(\boldsymbol{A}^{-1}\boldsymbol{b})^*,$$
$$\nabla_{\boldsymbol{b}}[\varphi \circ F](\boldsymbol{A}, \boldsymbol{b}) = -\boldsymbol{A}^{-*}\left(\nabla_{\boldsymbol{b}}[\varphi] \circ F(\boldsymbol{A}, \boldsymbol{b})\right),$$

where $\boldsymbol{A}^{-*} = (\boldsymbol{A}^{-1})^*$.

Proof. Let $\mathrm{d}[\varphi \circ F]$ denote the differential of $\varphi \circ F$ (and so on). We have for $\boldsymbol{\Delta}_{\boldsymbol{A}}$ and $\boldsymbol{\Delta}_{\boldsymbol{b}}$ the same shape as $\boldsymbol{A}$ and $\boldsymbol{b}$

$$\begin{aligned}\mathrm{d}F_{\boldsymbol{A},\boldsymbol{b}}(\boldsymbol{\Delta}_{\boldsymbol{A}}, \boldsymbol{\Delta}_{\boldsymbol{b}}) &= \left.\frac{\partial}{\partial t}\right|_{t=0} \left((\boldsymbol{A} + t\boldsymbol{\Delta}_{\boldsymbol{A}})^{-1}, -(\boldsymbol{A} + t\boldsymbol{\Delta}_{\boldsymbol{A}})^{-1}(\boldsymbol{b} + t\boldsymbol{\Delta}_{\boldsymbol{b}})\right) \\ &= \left(-\boldsymbol{A}^{-1}\boldsymbol{\Delta}_{\boldsymbol{A}}\boldsymbol{A}^{-1}, -\left(\boldsymbol{A}^{-1}\boldsymbol{\Delta}_{\boldsymbol{b}} - \boldsymbol{A}^{-1}\boldsymbol{\Delta}_{\boldsymbol{A}}\boldsymbol{A}^{-1}\boldsymbol{b}\right)\right)\end{aligned}$$

where the asserted expression for the derivative through the matrix inverse follows from, say, the Neumann series. Now, the chain rule and the definition of the gradient imply

$$\mathrm{d}[\varphi \circ F]_{\boldsymbol{A},\boldsymbol{b}}(\boldsymbol{\Delta}_{\boldsymbol{A}}, \boldsymbol{\Delta}_{\boldsymbol{b}}) = \left\langle \left(\nabla_{\boldsymbol{A}}[\varphi \circ F](\boldsymbol{A}, \boldsymbol{b}), \nabla_{\boldsymbol{b}}[\varphi \circ F](\boldsymbol{A}, \boldsymbol{b})\right), \left(-\boldsymbol{A}^{-1}\boldsymbol{\Delta}_{\boldsymbol{A}}\boldsymbol{A}^{-1}, -\left(\boldsymbol{A}^{-1}\boldsymbol{\Delta}_{\boldsymbol{b}} - \boldsymbol{A}^{-1}\boldsymbol{\Delta}_{\boldsymbol{A}}\boldsymbol{A}^{-1}\boldsymbol{b}\right)\right)\right\rangle,$$

and the claim follows by distributing and taking adjoints in order to read off the gradients from the previous expression. □

We remark that centering, as discussed in Appendix A.1, can be implemented identically to the standard parameterization case when using inverse parameterization.

A.3 Additional Experiments and Experimental Details

General details for experiments. We use normalized cross correlation (NCC) and zero-normalized cross correlation (ZNCC) for measuring the performance of registration on textured and spike data respectively. Specifically, for two multichannel images $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{m\times n\times c}$, let $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{Y}}$ be the channel-wise mean-subtracted images from $\boldsymbol{X}$ and $\boldsymbol{Y}$. The quantities NCC and ZNCC are defined as $\mathrm{NCC}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{\langle \boldsymbol{X}, \boldsymbol{Y}\rangle}{\|\boldsymbol{X}\|_F \|\boldsymbol{Y}\|_F}$ and $\mathrm{ZNCC}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{\langle \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}}\rangle}{\|\tilde{\boldsymbol{X}}\|_F \|\tilde{\boldsymbol{Y}}\|_F}$.

A.3.1 Figure 1 Experimental Details

In the experiment comparing the complexity of optimization and covering-based methods for textured motif detection shown in Figure 1, the raw background image used has dimension 2048×1536. The crab template is first placed at the center, then a random transformation is applied to generate the scene $\boldsymbol{y}$. Translation consists of random amounts on both x and y directions uniformly in $[-5, 5]$ pixels. Euclidean transforms in addition apply a rotation with angle uniformly from $[-\frac{\pi}{4}, \frac{\pi}{4}]$. Similarity transforms in addition applies a scaling uniformly from $[0.8, 1.25]$. Generic affine transforms are parameterized by a transformation matrix $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$ and offset vector $\boldsymbol{b} \in \mathbb{R}^2$, with the singular values of $\boldsymbol{A}$ uniformly from $[0.8, 1.25]$ and the left and right orthogonal matrices being rotation matrices with angle uniformly in $[-\frac{\pi}{4}, \frac{\pi}{4}]$. For each of the 4 modes of transform, 10 random images are generated. The optimization formulation used is (4), with $\boldsymbol{x}_o$ the crab body motif shown in Figure 2(a). The optimization-based method uses a multi-scale scheme, which uses a sequence of decreasing values of σ and step sizes, starting at $\sigma = 5$ and step size 0.005σ (except for affine mode which starts at $\sigma = 10$), with σ halved every 50 iterations until stopping criteria over the ZNCC is met, where ZNCC is calculated over the motif support Ω only. For each value of σ, a dilated support $\tilde{\Omega}$ is used, which is the dilation of Ω two σ away from the support of the motif. The background model covers the region up to 5σ away from the motif. The background $\boldsymbol{\beta}$ is initialized as a gaussian-smoothed version of the difference between the initialized image and the ground truth motif, and then continuously updated in the optimization. For the first 5 iterations of every new scale σ, only the background is updated while the transformation parameters are held constant. The covering-based method samples a random transform from the corresponding set of transforms used in each try.

A.3.2 Figure 4 Experimental Details

In the experiment of verifying the convergence of multichannel spike registration as shown in Figure 4, the motif consists of 5 spikes placed at uniformly random positions in a 61×81 image. To allow the spike locations to take non-integer values, we represent each spike as a gaussian density with standard deviation $\sigma_0 = 3$ centered at the spike location, and evaluated on the grid. A random affine transformation of the motif is generated as the scene. As a result, we are able to use this σ_0-smoothed input in (10) *without extra smoothing*, and we can compensate the variance of the filter applied to $\boldsymbol{x}_o$ in the formulation to account for the fact that we already smoothed by σ_0 when generating the data. The smoothing level in the registration is chosen according to equation (16) in Theorem 5.1. Due to the discretization effect and various artifacts, the step sizes prescribed in Theorem 5.1 will lead to divergence, so we reduce the step sizes by multiplying a factor of 0.2.

A.3.3 Further Experimental Details

The beach background used for embedding the crab template throughout the experiments is `CC0`-licensed and available online: `https://www.flickr.com/photos/scotnelson/28315592012`. Our code and data are available at `https://github.com/sdbuch/refine`.

A.4 Canonized Object Preprocessing and Calibration for Hierarchical Detection

The hierarchical detection network implementation prescription in Section 4.3 assumes the occurrence maps $\boldsymbol{x}_v$ for $v \in V \setminus \{1, \dots, K\}$ are given; in practice, these are first calculated using the template $\boldsymbol{y}_o$ and its motifs, by a process we refer to as extraction. Simultaneously, to extract these occurrence maps and have them be useful for subsequent detections it is necessary to have appropriate choices for the various hyperparameters involved in the network: we classify these as 'registration' hyperparameters (for each $v \in V$, the step size ν_v; the image, scene, and input smoothing parameters σ_v^2, $\sigma_{0,v}^2$, and σ_{in}^2; the number of registration iterations T_v; and the vertical ("height") and horizontal ("width") stride sizes $\Delta_{H,v}$ and $\Delta_{W,v}$) or 'detection' hyperparameters (for each $v \in V$, the suppression parameter α_v and the threshold parameter γ_v). We describe these issues below, as well as other relevant implementation issues.

Hyperparameter selection. We discuss this point first, because it is necessary to process the 'leaf' motifs before any occurrence maps can be extracted. In practice, we 'calibrate' these hyperparameters by testing whether detections succeed or fail given the canonized template $\boldsymbol{y}_o$ as input to the (partial) network. Below, we first discuss hyperparameters related to visual motifs (i.e., the formulation (12)), then hyperparameters for spiky motifs (i.e., the formulation (13)).

Stride density and convergence speed: The choice of these parameters encompasses a basic computational tradeoff: setting T_v larger allows to leverage the entire basin of attraction of the formulations (12) and (13), enabling more reliable values of $\min_{\boldsymbol{\lambda} \in \Lambda_v} \text{loss}(v, \boldsymbol{\lambda})$ and the use of larger values of $\Delta_{H,v}$ and $\Delta_{W,v}$; however, it requires more numerical operations (convolutions and interpolations) for each stride $\boldsymbol{\lambda} \in \Lambda_v$. In our experiments we err on the side of setting T_v large, and tune the stride sizes $\Delta_{W,v}$ and $\Delta_{H,v}$ over multiples of 4 (setting them as large as possible while being able to successfully detect motifs). The choice of the step sizes ν_v is additionally complicated by the smoothing and motif-dependence of this parameter. As we describe in Section 3.3, we treat the step sizes taken on each component of $(\boldsymbol{A}, \boldsymbol{b})$ independently, and in our experiments use a small multiple (i.e., 1/10) of $t_v^{\boldsymbol{A}} = 4\sigma / \max\{m_v^2, n_v^2\}$ and $t_v^{\boldsymbol{b}} = 2\sigma / \max\{m_v, n_v\}$ for all visual motifs. This prescription is a heuristic that we find works well for the motifs and smoothing parameters (see the next point below) we test, inspired by the theoretical prescriptions in Section 5 for spike alignment that we discuss later in this section.

Smoothing parameters: The smoothing level σ_v^2 in (12) increases the size of the basin of attraction when set larger. For this specific formulation, we find it more efficient to expand the basin by striding, and enforce a relatively small value of $\sigma_v^2 = 9$ for all visual motifs. Without input smoothing, we empirically observe that the first-round-multiscale cost-smoothed formulation (12) is slightly unstable with respect to

high-frequency content in $\boldsymbol{y}$: this motivates us to introduce this extra smoothing with variance $\sigma^2_{\text{in}} = 9/4$, which removes interpolation artifacts that hinder convergence. We find the multiscale smoothing mode of operation described in Section 4.3 to be essential for distinguishing between strides $\boldsymbol{\lambda}$ which have "failed" to register the motif $\boldsymbol{x}_v$ and those that have succeeded, through the error $\text{loss}(\boldsymbol{\lambda}, v)$: in all experiments, we run the second-phase multiscale round for (12) as described in Section 4.3, for 256 iterations and with $\sigma^2 = 10^{-2}$ and $\sigma^2_{\text{in}} = 10^{-12}$. We describe the choice of $\sigma^2_{0,v}$ below, as it is more of a spike registration hyperparameter (c.f. (14)).

Detection parameters: The scale parameters α_v are set based on the size of the basin of attraction around the true transformation of $\boldsymbol{x}_v$, and in particular on the scale of $\text{loss}(\boldsymbol{\lambda}, v)$ at "successes" and "failures" to register. In our experiments, we simply set $\alpha_v = 1$ for visual motifs. The choice of the threshold parameter γ_v is significantly more important: it accounts for the fact that the final cost $\text{loss}(\boldsymbol{\lambda}, v)$ at a successful registration is sensitive to both the motif $\boldsymbol{x}_v$ *and* the background/visual clutter present in the input $\boldsymbol{y}$. In our experiments in Section 4.4, we tune the parameters γ_v on a per-motif basis by calculating $\text{loss}(\boldsymbol{\lambda}, v)$ for embeddings $\boldsymbol{y}_o \circ \boldsymbol{\tau}_0$ for $\boldsymbol{\tau}_0 \in \text{SO}(2)$ up to some maximum rotation angle in visual clutter, classifying each $\boldsymbol{\lambda}$ as either a successful registration or a failure, and then picking $\boldsymbol{\gamma}_v$ to separate the successful runs for all rotation angles from the failing runs. For the motifs and range of rotation angles we consider, we find that such a threshold always exists. However, at larger rotation angles we run into issues with the left and right eye motifs being too similar to each other, leading to spurious registrations and the non-existence of a separating threshold. In practice, this calibration scheme also requires a method of generating visual clutter that matches the environments one intends to evaluate in. The calibrated threshold parameters used for our experiments in Section 4.4 are available in our released implementation.

Hyperparameters for spiky motifs: The same considerations apply to hyperparameter selection for spiky motifs (i.e., the formulation (13)). However, the extra structure in such data facilitates a theoretical analysis that corroborates the intuitive justifications for hyperparameter tradeoffs we give above and leads to specific prescriptions for most non-detection hyperparameters, allowing them to be set in a completely tuning-free fashion. We present these results in Section 5. For detection hyperparameters, we follow the same iterated calibration process as for visual motifs, with scale parameters $\alpha_v = 2.5 \cdot 10^5$ (typical values of the cost (13) are much smaller than those of the cost (12), due to the fact that the gaussian density has a small L^2 norm). For the occurrence map smoothing parameters $\sigma^2_{0,v}$, our network construction above necessitates setting these parameters to be the same for all $v \in V$; we find empirically that a setting $\sigma^2_{0,v} = 9$ is sufficient to avoid interpolation artifacts. Finally, the bounding box masks Ω_v are set during the extraction process (see below), and are dilated by twice the total size of the filters $\boldsymbol{g}_{\sigma^2_v}$. In practice, when implementing gaussian filters, we make the image size square, with side lengths 6σ (rounded to the next largest odd integer).

Occurrence map extraction. Although the criteria above (together with the theoretical guidance from Section 5) are sufficient to develop a completely automatic calibration process for the various hyperparameters above, in practice we perform calibration and occurrence map extraction in a 'human-in-the-loop' fashion. The extraction process can be summarized as follows (it is almost identical to the detection process described in Section 4.3, with a few extra steps implicitly interspersed with calibration of the various hyperparameters):

1. **Use the canonized template as input:** We set $\boldsymbol{y}_o$ as the network's input.
2. **Process leaf motifs:** Given suitable calibrated settings of the hyperparameters for leaf motifs $v \in V$, perform detection and generate all occurrence maps $\boldsymbol{\omega}_v$ via (14).
3. **Extract occurrence motifs at depth** $\text{diam}(G) - 1$**:** For each v with $d(v) = \text{diam}(G) - 1$, we follow the assumptions made in Section 4.1 (in particular, that each visual motif occurs only once in $\boldsymbol{y}_o$ and G is a tree) and after aggregating the occurrence map from v's child nodes via (11), we extract $\boldsymbol{x}_v$ from $\boldsymbol{y}_v$ by cropping to the bounding box for the support of $\boldsymbol{y}_v$. Technically, since (14) uses a gaussian filter, the support will be nonzero everywhere, and instead we threshold at a small nonzero value (e.g. 1/20 in our experiments) to determine the "support".

4. **Continue to the root of G:** Perform registration to generate the occurrence maps for nodes at depth $\mathrm{diam}(G) - 1$, then continue to iterate the above steps until the root node is reached and processed.

Note that the extracted occurrence motifs $\boldsymbol{x}_v$ for $v \in V \setminus \{1, \dots, K\}$ depend on proper settings of the registration and detection hyperparameters: if these parameters are set imprecisely, the extracted occurrence maps will not represent ideal detections (e.g. they may not be close to a full-amplitude gaussian at the locations of the motifs in $\boldsymbol{y}_o$ as they should, or they may not suppress failed detections enough).

Other implementation issues. The implementation issue of centering, discussed in Appendix A.1, is relevant to the implementation of the unrolled solvers for (13) and (14). We find that a useful heuristic is to center the transformation $\boldsymbol{\tau}$ at the location of the center pixel of the embedded motif $\boldsymbol{x}_v$ (i.e., for a stride $\boldsymbol{\lambda} \in \Lambda_v$, at the coordinates $\boldsymbol{\lambda} + ((m_v - 1)/2, (n_v - 1)/2)$). To implement this centering, the locations of the detections in the spike map definition (14) need to have the offsets $((m_v - 1)/2, (n_v - 1)/2)$ added.

The network construction in Section 4.3 relies on the extraction process described above to employ an identical enumeration strategy in the traversal of the graph G as the detection process (i.e., assuming that nodes are ordered in increasing order above). In our implementation described in Section 4.4, we instead label nodes arbitrarily when preparing the network's input, and leave consistent enumeration of nodes during traversal to the NetworkX graph processing library [HSS08].

B Proof of Theorem 5.1

We consider a continuum model for multichannel spike alignment, motivated by the higher-level features arising in the hierarchical detection network developed in Section 4: signals $\boldsymbol{X}$ are represented as elements of $\mathbb{R}^{\mathbb{R}\times\mathbb{R}\times C}$, and are identifiable with C-element real-valued vector fields on the (continuous, infinite) image plane $\mathbb{R}^2$. In this setting, we write $\|\boldsymbol{X}\|_{L^2}^2 = \sum_{i=1}^C \|\boldsymbol{X}_i\|_{L^2}^2$ for the natural product norm (in words, the ℓ^2 norm of the vector of channelwise L^2 norms of $\boldsymbol{X}$). For $\boldsymbol{p} \in \mathbb{R}^2$, let $\boldsymbol{\delta}_{\boldsymbol{p}} \in \mathbb{R}^{\mathbb{R}\times\mathbb{R}}$ denote a Dirac distribution centered at $\boldsymbol{p}$, defined via

$$\int_{\mathbb{R}^2} \boldsymbol{\delta}_{\boldsymbol{p}}(\boldsymbol{x}) f(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = f(\boldsymbol{p})$$

for all Schwartz functions f [SW71, §I.3]. This models a 'perfect' spike signal. For $\boldsymbol{p} \in \mathbb{R}^2$ and $\boldsymbol{M} \in \mathbb{R}^{2\times 2}$ positive semidefinite, let $\boldsymbol{g}_{\boldsymbol{p},\boldsymbol{M}}$ denote the gaussian density on $\mathbb{R}^2$ with mean $\boldsymbol{p}$ and covariance matrix $\boldsymbol{M}$. Consider a target signal

$$\boldsymbol{X}_o = \sum_{i=1}^{c} \boldsymbol{\delta}_{\boldsymbol{v}_i} \otimes \boldsymbol{e}_i, \tag{28}$$

and an observation

$$\boldsymbol{X} = \sum_{i=1}^{c} \boldsymbol{\delta}_{\boldsymbol{u}_i} \otimes \boldsymbol{e}_i \tag{29}$$

satisfying

$$\boldsymbol{v}_i = \boldsymbol{A}_\star \boldsymbol{u}_i + \boldsymbol{b}_\star \tag{30}$$

for some $(\boldsymbol{A}_\star, \boldsymbol{b}_\star) \in \mathrm{GL}(2)\times\mathbb{R}^2$. These represent the unknown ground-truth affine transform to be recovered. Consider the objective function

$$\varphi_{L^2,\sigma}(\boldsymbol{A}, \boldsymbol{b}) \equiv \frac{1}{2c} \left\| \boldsymbol{g}_{\boldsymbol{0},\sigma^2\boldsymbol{I}-\sigma_0^2(\boldsymbol{A}^*\boldsymbol{A})^{-1}} * \left(\det{}^{1/2}(\boldsymbol{A}^*\boldsymbol{A}) \left(\boldsymbol{g}_{\boldsymbol{0},\sigma_0^2\boldsymbol{I}} * \boldsymbol{X} \right) \circ \boldsymbol{\tau}_{\boldsymbol{A},\boldsymbol{b}} \right) - \boldsymbol{g}_{\boldsymbol{0},\sigma^2\boldsymbol{I}} * \boldsymbol{X}_o \right\|_{L^2}^2,$$

where $\boldsymbol{A}^*$ denotes the transpose, convolutions are applied channelwise, and for a signal $\boldsymbol{S} \in \mathbb{R}^{\mathbb{R}\times\mathbb{R}\times c}$, $\boldsymbol{S} \circ \boldsymbol{\tau}_{\boldsymbol{A},\boldsymbol{b}}(u, v) = \boldsymbol{S}(a_{11}u + a_{12}v + b_1, a_{21}u + a_{22}v + b_2)$. We study the following "inverse parameterization" of this function:

$$\varphi_{L^2,\sigma}^{\mathrm{inv}}(\boldsymbol{A}, \boldsymbol{b}) \equiv \varphi_{L^2,\sigma}(\boldsymbol{A}^{-1}, -\boldsymbol{A}^{-1}\boldsymbol{b}).$$

We analyze the performance of gradient descent for solving the optimization problem

$$\min_{\boldsymbol{A},\boldsymbol{b}} \varphi^{\mathrm{inv}}_{L^2,\sigma}(\boldsymbol{A},\boldsymbol{b}).$$

Under mild conditions, local minimizers of this problem are global. Moreover, if σ is set appropriately, the method exhibits linear convergence to the truth:

Theorem B.1 (Multichannel Spike Model, Affine Transforms, L^2)**.** *Consider an instance of the multichannel spike model* (28)-(29)-(30)*, with* $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_c] \in \mathbb{R}^{2\times c}$*. Assume that the spikes* $\boldsymbol{U}$ *are centered and nondegenerate, so that* $\boldsymbol{U}\mathbf{1} = \mathbf{0}$ *and* $\mathrm{rank}(\boldsymbol{U}) = 2$*. Then gradient descent*

$$\begin{aligned} \boldsymbol{A}_{k+1} &= \boldsymbol{A}_k - \nu t_{\boldsymbol{A}} \nabla_{\boldsymbol{A}} \varphi^{\mathrm{inv}}_{L^2,\sigma}(\boldsymbol{A}_k, \boldsymbol{b}_k), \\ \boldsymbol{b}_{k+1} &= \boldsymbol{b}_k - \nu t_{\boldsymbol{b}} \nabla_{\boldsymbol{b}} \varphi^{\mathrm{inv}}_{L^2,\sigma}(\boldsymbol{A}_k, \boldsymbol{b}_k) \end{aligned}$$

with smoothing

$$\sigma^2 \geq 2\frac{\max_i \|\boldsymbol{u}_i\|_2^2}{s_{\min}(\boldsymbol{U})^2} \left(s_{\max}(\boldsymbol{U})^2 \|\boldsymbol{A}_\star - \boldsymbol{I}\|_F^2 + c\|\boldsymbol{b}_\star\|_2^2 \right)$$

and step sizes

$$\begin{aligned} t_{\boldsymbol{A}} &= \frac{c}{s_{\max}(\boldsymbol{U})^2}, \\ t_{\boldsymbol{b}} &= 1, \\ \nu &= 8\pi\sigma^4, \end{aligned}$$

from initialization $\boldsymbol{A}_0 = \boldsymbol{I}, \boldsymbol{b}_0 = \mathbf{0}$ *satisfies*

$$t_{\boldsymbol{A}}^{-1}\|\boldsymbol{A}_k - \boldsymbol{A}_\star\|_F^2 + \|\boldsymbol{b}_k - \boldsymbol{b}_\star\|_2^2 \leq \left(1 - \frac{1}{2\kappa}\right)^{2k} \left(t_{\boldsymbol{A}}^{-1}\|\boldsymbol{I} - \boldsymbol{A}_\star\|_F^2 + \|\boldsymbol{b}_\star\|_2^2 \right), \tag{31}$$

where

$$\kappa = \frac{s_{\max}(\boldsymbol{U})^2}{s_{\min}(\boldsymbol{U})^2},$$

with $s_{\min}(\boldsymbol{U})$ *and* $s_{\max}(\boldsymbol{U})$ *denoting the minimum and maximum singular values of the matrix* $\boldsymbol{U}$*.*

Proof. Below, we use the notation $\|\boldsymbol{M}\|_{\ell^p\to\ell^q} = \sup_{\|\boldsymbol{x}\|_p\leq 1}\|\boldsymbol{M}\boldsymbol{x}\|_q$. We begin by rephrasing the objective function in a simpler form: by properties of the gaussian density,

$$\varphi_{L^2,\sigma}(\boldsymbol{A},\boldsymbol{b}) = \frac{1}{2c}\sum_{i=1}^{c} \left\| g_{\boldsymbol{A}^{-1}(\boldsymbol{u}_i - \boldsymbol{b}),\sigma^2\boldsymbol{I}} - g_{\boldsymbol{v}_i,\sigma^2\boldsymbol{I}} \right\|_{L^2}^2,$$

whence by an orthogonal change of coordinates

$$\varphi_{L^2,\sigma}(\boldsymbol{A},\boldsymbol{b}) = \frac{1}{c}\sum_{i=1}^{c} \psi\left(\tfrac{1}{2}\|\boldsymbol{A}^{-1}(\boldsymbol{u}_i - \boldsymbol{b}) - \boldsymbol{v}_i\|_2^2\right),$$

where

$$\begin{aligned} \psi(t^2/2) &= \tfrac{1}{2}\left\| g_{t\boldsymbol{e}_1,\sigma^2\boldsymbol{I}} - g_{\mathbf{0},\sigma^2\boldsymbol{I}} \right\|_{L^2}^2 \\ &= \frac{1}{4\pi\sigma^2} - \left\langle g_{t\boldsymbol{e}_1,\sigma^2\boldsymbol{I}}, g_{\mathbf{0},\sigma^2\boldsymbol{I}} \right\rangle \\ &= \frac{1}{4\pi\sigma^2} - \frac{1}{(2\pi\sigma^2)^2}\left(\int_{\mathbb{R}} e^{-s^2/\sigma^2} ds\right)\left(\int_{\mathbb{R}} e^{-(s-t)^2/2\sigma^2} e^{-s^2/2\sigma^2} ds\right) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{4\pi\sigma^2} - \frac{2^{-1/2}}{(2\pi\sigma^2)^{3/2}} \int_{\mathbb{R}} e^{-\frac{(s-t/2)^2}{\sigma^2}} e^{-\frac{t^2}{4\sigma^2}} ds \\
&= \frac{1}{4\pi\sigma^2} \left(1 - \exp\left(-\frac{t^2/2}{2\sigma^2}\right)\right).
\end{aligned}
$$

So

$$
\varphi^{\mathrm{inv}}_{L^2,\sigma}(\boldsymbol{A}, \boldsymbol{b}) = \varphi_{L^2,\sigma}(\boldsymbol{A}^{-1}, -\boldsymbol{A}^{-1}\boldsymbol{b}) = \frac{1}{c} \sum_{i=1}^{c} \psi(\tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{u}_i + \boldsymbol{b} - \boldsymbol{v}_i\|_2^2).
$$

Differentiating, we obtain

$$
\begin{aligned}
\nabla_{\boldsymbol{A}} \varphi^{\mathrm{inv}}_{L^2,\sigma}(\boldsymbol{A}, \boldsymbol{b}) &= \frac{1}{c} \sum_{i=1}^{c} \dot{\psi}(\tfrac{1}{2}\|\boldsymbol{\delta}_i\|_2^2) \boldsymbol{\delta}_i \boldsymbol{u}_i^* \\
\nabla_{\boldsymbol{b}} \varphi^{\mathrm{inv}}_{L^2,\sigma}(\boldsymbol{A}, \boldsymbol{b}) &= \frac{1}{c} \sum_{i=1}^{c} \dot{\psi}(\tfrac{1}{2}\|\boldsymbol{\delta}_i\|_2^2) \boldsymbol{\delta}_i,
\end{aligned}
$$

where for concision

$$
\begin{aligned}
\boldsymbol{\delta}_i &= \boldsymbol{A}\boldsymbol{u}_i + \boldsymbol{b} - \boldsymbol{v}_i \\
&= (\boldsymbol{A} - \boldsymbol{A}_\star)\boldsymbol{u}_i + \boldsymbol{b} - \boldsymbol{b}_\star.
\end{aligned}
$$

In these terms, we have the following expression for a single iteration of gradient descent:

$$
\begin{aligned}
\boldsymbol{A}^+ &= \boldsymbol{A} - \frac{t_{\boldsymbol{A}}}{c} \sum_{i=1}^{c} \nu \dot{\psi}(\tfrac{1}{2}\|\boldsymbol{\delta}_i\|_2^2) \boldsymbol{\delta}_i \boldsymbol{u}_i^* \\
&= \boldsymbol{A} - \frac{t_{\boldsymbol{A}}}{c} \sum_{i=1}^{c} \nu \dot{\psi}(\tfrac{1}{2}\|\boldsymbol{\delta}_i\|_2^2) (\boldsymbol{A} - \boldsymbol{A}_\star)\boldsymbol{u}_i \boldsymbol{u}_i^* - \frac{t_{\boldsymbol{A}}}{c} \sum_{i=1}^{c} \nu \dot{\psi}(\tfrac{1}{2}\|\boldsymbol{\delta}_i\|_2^2) (\boldsymbol{b} - \boldsymbol{b}_\star)\boldsymbol{u}_i^* \\
&= \boldsymbol{A} - \frac{\nu t_{\boldsymbol{A}}}{c} (\boldsymbol{A} - \boldsymbol{A}_\star)\boldsymbol{U}\dot{\boldsymbol{\Psi}}\boldsymbol{U}^* - \frac{\nu t_{\boldsymbol{A}}}{c} (\boldsymbol{b} - \boldsymbol{b}_\star)\dot{\boldsymbol{\psi}}^* \boldsymbol{U}^*
\end{aligned}
$$

and

$$
\begin{aligned}
\boldsymbol{b}^+ &= \boldsymbol{b} - \frac{t_{\boldsymbol{b}}}{c} \sum_{i=1}^{c} \nu \dot{\psi}(\tfrac{1}{2}\|\boldsymbol{\delta}_i\|_2^2) \boldsymbol{\delta}_i \\
&= \boldsymbol{b} - \frac{t_{\boldsymbol{b}}}{c} (\boldsymbol{b} - \boldsymbol{b}_\star) \left\langle \boldsymbol{1}, \nu \dot{\boldsymbol{\psi}} \right\rangle - \frac{\nu t_{\boldsymbol{b}}}{c} (\boldsymbol{A} - \boldsymbol{A}_\star)\boldsymbol{U}\dot{\boldsymbol{\psi}},
\end{aligned}
$$

where above, we have set

$$
\dot{\boldsymbol{\Psi}} = \begin{bmatrix} \dot{\psi}(\frac{1}{2}\|\boldsymbol{\delta}_1\|_2^2) & & \\ & \ddots & \\ & & \dot{\psi}(\frac{1}{2}\|\boldsymbol{\delta}_c\|_2^2) \end{bmatrix} \in \mathbb{R}^{c \times c}, \qquad \dot{\boldsymbol{\psi}} = \begin{bmatrix} \dot{\psi}(\frac{1}{2}\|\boldsymbol{\delta}_1\|_2^2) \\ \vdots \\ \dot{\psi}(\frac{1}{2}\|\boldsymbol{\delta}_c\|_2^2) \end{bmatrix} \in \mathbb{R}^{c}. \tag{32}
$$

Writing $\boldsymbol{\Delta}_{\boldsymbol{A}} = \boldsymbol{A} - \boldsymbol{A}_\star$, $\boldsymbol{\Delta}_{\boldsymbol{b}} = \boldsymbol{b} - \boldsymbol{b}_\star$, we have

$$
\begin{aligned}
\boldsymbol{\Delta}_{\boldsymbol{A}}^+ &= \boldsymbol{\Delta}_{\boldsymbol{A}} \Big(\boldsymbol{I} - \tfrac{\nu t_{\boldsymbol{A}}}{c} \boldsymbol{U}\dot{\boldsymbol{\Psi}}\boldsymbol{U}^* \Big) - \tfrac{\nu t_{\boldsymbol{A}}}{c} \boldsymbol{\Delta}_{\boldsymbol{b}} \dot{\boldsymbol{\psi}}^* \boldsymbol{U}^* \\
\boldsymbol{\Delta}_{\boldsymbol{b}}^+ &= \Big(1 - \tfrac{t_{\boldsymbol{b}}}{c} \left\langle \boldsymbol{1}, \nu \dot{\boldsymbol{\psi}} \right\rangle \Big) \boldsymbol{\Delta}_{\boldsymbol{b}} - \boldsymbol{\Delta}_{\boldsymbol{A}} \tfrac{\nu t_{\boldsymbol{b}}}{c} \boldsymbol{U}\dot{\boldsymbol{\psi}}.
\end{aligned}
$$

To facilitate a convergence proof, we modify this equation to pertain to scaled versions of $\boldsymbol{\Delta}_{\boldsymbol{A}}$, $\boldsymbol{\Delta}_{\boldsymbol{b}}$:

$$
t_{\boldsymbol{A}}^{-1/2} \boldsymbol{\Delta}_{\boldsymbol{A}}^+ = (t_{\boldsymbol{A}}^{-1/2} \boldsymbol{\Delta}_{\boldsymbol{A}}) \Big(\boldsymbol{I} - \tfrac{t_{\boldsymbol{A}}}{c} \boldsymbol{U}(\nu\dot{\boldsymbol{\Psi}})\boldsymbol{U}^* \Big) - \tfrac{t_{\boldsymbol{A}}^{1/2} t_{\boldsymbol{b}}^{1/2}}{c} (t_{\boldsymbol{b}}^{-1/2} \boldsymbol{\Delta}_{\boldsymbol{b}})(\nu\dot{\boldsymbol{\psi}})^* \boldsymbol{U}^*
$$

$$t_b^{-1/2}\Delta_b^+ \;=\; \Big(1-\tfrac{t_b}{c}\left\langle \mathbf{1}, \nu\dot{\psi}\right\rangle\Big)(t_b^{-1/2}\Delta_b) - (t_A^{-1/2}\Delta_A)\tfrac{t_b^{1/2}t_A^{1/2}}{c}\boldsymbol{U}(\nu\dot{\psi}).$$

In matrix-vector form, and writing $\bar{\Delta}_A = t_A^{-1/2}\Delta_A$ and $\bar{\Delta}_b = t_b^{-1/2}\Delta_b$, we have

$$\begin{aligned}
\begin{bmatrix} \mathrm{vec}(\bar{\Delta}_A) \\ \bar{\Delta}_b \end{bmatrix}^+ &= \left(\boldsymbol{I}_6 - \begin{bmatrix} \frac{t_A}{c}\boldsymbol{U}(\nu\dot{\boldsymbol{\Psi}})\boldsymbol{U}^* \otimes \boldsymbol{I}_2 & \frac{t_A^{1/2}t_b^{1/2}}{c}(\boldsymbol{U}(\nu\dot{\psi})) \otimes \boldsymbol{I}_2 \\ \frac{t_A^{1/2}t_b^{1/2}}{c}((\nu\dot{\psi})^*\boldsymbol{U}^*) \otimes \boldsymbol{I}_2 & \frac{t_b}{c}\left\langle \mathbf{1}, \nu\dot{\psi}\right\rangle \otimes \boldsymbol{I}_2 \end{bmatrix}\right)\begin{bmatrix} \mathrm{vec}(\bar{\Delta}_A) \\ \bar{\Delta}_b \end{bmatrix} \\
&= \left(\boldsymbol{I}_6 - \begin{bmatrix} \frac{t_A}{c}\boldsymbol{U}(\nu\dot{\boldsymbol{\Psi}})\boldsymbol{U}^* & \frac{t_A^{1/2}t_b^{1/2}}{c}(\boldsymbol{U}(\nu\dot{\psi})) \\ \frac{t_A^{1/2}t_b^{1/2}}{c}((\nu\dot{\psi})^*\boldsymbol{U}^*) & \frac{t_b}{c}\left\langle \mathbf{1}, \nu\dot{\psi}\right\rangle \end{bmatrix} \otimes \boldsymbol{I}_2\right)\begin{bmatrix} \mathrm{vec}(\bar{\Delta}_A) \\ \bar{\Delta}_b \end{bmatrix} \\
&\doteq \boldsymbol{M}\begin{bmatrix} \mathrm{vec}(\bar{\Delta}_A) \\ \bar{\Delta}_b \end{bmatrix}, \qquad (33)
\end{aligned}$$

where in this context $\otimes$ denotes the Kronecker product of matrices. Since $\boldsymbol{I}_6 = \boldsymbol{I}_4 \otimes \boldsymbol{I}_2$, and because the eigenvalues of a Kronecker product of symmetric matrices are the pairwise products of the eigenvalues of each factor, we have

$$\|\boldsymbol{M}\|_{\ell^2\to\ell^2} = \left\|\boldsymbol{I} - \begin{bmatrix} \frac{t_A}{c}\boldsymbol{U}(\nu\dot{\boldsymbol{\Psi}})\boldsymbol{U}^* & \frac{t_A^{1/2}t_b^{1/2}}{c}(\boldsymbol{U}(\nu\dot{\psi})) \\ \frac{t_A^{1/2}t_b^{1/2}}{c}((\nu\dot{\psi})^*\boldsymbol{U}^*) & \frac{t_b}{c}\left\langle \mathbf{1}, \nu\dot{\psi}\right\rangle \end{bmatrix}\right\|_{\ell^2\to\ell^2}.$$

By our choice of t_A and t_b, and the assumption $\boldsymbol{U}\mathbf{1} = \mathbf{0}$, we can write

$$\begin{bmatrix} \frac{t_A}{c}\boldsymbol{U}(\nu\dot{\boldsymbol{\Psi}})\boldsymbol{U}^* & \frac{t_A^{1/2}t_b^{1/2}}{c}(\boldsymbol{U}(\nu\dot{\psi})) \\ \frac{t_A^{1/2}t_b^{1/2}}{c}((\nu\dot{\psi})^*\boldsymbol{U}^*) & \frac{t_b}{c}\left\langle \mathbf{1}, \nu\dot{\psi}\right\rangle \end{bmatrix} = \begin{bmatrix} \frac{t_A}{c}\boldsymbol{U}(\nu\dot{\boldsymbol{\Psi}} - \boldsymbol{I})\boldsymbol{U}^* + \frac{\boldsymbol{U}\boldsymbol{U}^*}{\|\boldsymbol{U}\|^2_{\ell^2\to\ell^2}} & \frac{t_A^{1/2}t_b^{1/2}}{c}(\boldsymbol{U}(\nu\dot{\psi} - \mathbf{1})) \\ \frac{t_A^{1/2}t_b^{1/2}}{c}((\nu\dot{\psi} - \mathbf{1})^*\boldsymbol{U}^*) & \frac{t_b}{c}\left\langle \mathbf{1}, \nu\dot{\psi} - \mathbf{1}\right\rangle + 1 \end{bmatrix}$$

and so by the triangle inequality for the operator norm

$$\begin{aligned}
\|\boldsymbol{M}\|_{\ell^2\to\ell^2} &\le \left\|\boldsymbol{I} - \begin{bmatrix} \frac{\boldsymbol{U}\boldsymbol{U}^*}{\|\boldsymbol{U}\|^2_{\ell^2\to\ell^2}} & \\ & 1 \end{bmatrix}\right\|_{\ell^2\to\ell^2} + \left\|\begin{bmatrix} \frac{\boldsymbol{U}}{\|\boldsymbol{U}\|_{\ell^2\to\ell^2}}(\nu\dot{\boldsymbol{\Psi}} - \boldsymbol{I})\frac{\boldsymbol{U}^*}{\|\boldsymbol{U}\|_{\ell^2\to\ell^2}} & \frac{1}{\sqrt{c}}\frac{\boldsymbol{U}}{\|\boldsymbol{U}\|_{\ell^2\to\ell^2}}(\nu\dot{\psi} - \mathbf{1}) \\ \frac{1}{\sqrt{c}}(\nu\dot{\psi} - \mathbf{1})^*\frac{\boldsymbol{U}^*}{\|\boldsymbol{U}\|_{\ell^2\to\ell^2}} & \left\langle \frac{\mathbf{1}}{c}, \nu\dot{\psi} - \mathbf{1}\right\rangle \end{bmatrix}\right\|_{\ell^2\to\ell^2} \\
&\le 1 - \frac{1}{\kappa} + 2\|\nu\dot{\psi} - \mathbf{1}\|_{\ell^\infty}, \qquad (34)
\end{aligned}$$

since

$$\begin{aligned}
\left\|\begin{bmatrix} \frac{\boldsymbol{U}}{\|\boldsymbol{U}\|_{\ell^2\to\ell^2}}(\nu\dot{\boldsymbol{\Psi}} - \boldsymbol{I})\frac{\boldsymbol{U}^*}{\|\boldsymbol{U}\|_{\ell^2\to\ell^2}} & \frac{1}{\sqrt{c}}\frac{\boldsymbol{U}}{\|\boldsymbol{U}\|_{\ell^2\to\ell^2}}(\nu\dot{\psi} - \mathbf{1}) \\ \frac{1}{\sqrt{c}}(\nu\dot{\psi} - \mathbf{1})^*\frac{\boldsymbol{U}^*}{\|\boldsymbol{U}\|_{\ell^2\to\ell^2}} & \left\langle \frac{\mathbf{1}}{c}, \nu\dot{\psi} - \mathbf{1}\right\rangle \end{bmatrix}\right\|_{\ell^2\to\ell^2} &\le \left\|\begin{bmatrix} \frac{\boldsymbol{U}\,\mathrm{diag}(\nu\dot{\psi}-\mathbf{1})\boldsymbol{U}^*}{\|\boldsymbol{U}\|^2_{\ell^2\to\ell^2}} & \mathbf{0} \\ \mathbf{0} & \left\langle \frac{\mathbf{1}}{c}, \nu\dot{\psi} - \mathbf{1}\right\rangle \end{bmatrix}\right\|_{\ell^2\to\ell^2} \\
&\quad + \left\|\begin{bmatrix} \mathbf{0} & \frac{1}{\sqrt{c}}\frac{\boldsymbol{U}(\nu\dot{\psi}-\mathbf{1})}{\|\boldsymbol{U}\|_{\ell^2\to\ell^2}} \\ \frac{1}{\sqrt{c}}\frac{(\nu\dot{\psi}-\mathbf{1})^*\boldsymbol{U}^*}{\|\boldsymbol{U}\|_{\ell^2\to\ell^2}} & \mathbf{0} \end{bmatrix}\right\|_{\ell^2\to\ell^2}
\end{aligned}$$

and by Hölder's inequality

$$|\langle \tfrac{\mathbf{1}}{c}, \nu\dot{\psi} - \mathbf{1}\rangle| \le \|\nu\dot{\psi} - \mathbf{1}\|_{\ell^\infty}, \quad \tfrac{1}{\sqrt{c}}\|\nu\dot{\psi} - \mathbf{1}\|_{\ell^2} \le \|\nu\dot{\psi} - \mathbf{1}\|_{\ell^\infty}.$$

Inductive argument for (31). We begin by noting that since $\boldsymbol{A}_0 = \boldsymbol{I}$, $\boldsymbol{b}_0 = \mathbf{0}$, (31) holds for $k = 0$. Now assume that it is true for $0, 1, \ldots, k-1$. If we can verify that

$$2\|\nu\dot{\psi} - \mathbf{1}\|_{\ell^\infty} \le \frac{1}{2\kappa}, \qquad (35)$$

then by (33) and (34) together with $t_b = 1$, we have

$$\left\| \begin{bmatrix} t_A^{-1/2}\mathrm{vec}(A_k - A_\star) \\ b_k - b_\star \end{bmatrix} \right\|_F^2 \leq \left(1 - \frac{1}{2\kappa}\right)^2 \left\| \begin{bmatrix} t_A^{-1/2}\mathrm{vec}(A_{k-1} - A_\star) \\ b_{k-1} - b_\star \end{bmatrix} \right\|_F^2.$$

Applying the inductive hypothesis, we obtain (31) for iteration k. So, once we can show that under the inductive hypothesis, (35) holds, the result will be established.

We begin by showing that under the inductive hypothesis, the errors δ_i are all bounded. Indeed, by the parallelogram law

$$\begin{aligned} \|\delta_i\|_2^2 &= \|A_{k-1}u_i + b_{k-1} - v_i\|_2^2 \\ &= \|(A_{k-1} - A_\star)u_i + (b_{k-1} - b_\star)\|_2^2 \\ &\leq 2\frac{\|A_{k-1} - A_\star\|_F^2}{t_A} t_A\|u_i\|_2^2 + 2\|b_{k-1} - b_\star\|_2^2, \end{aligned}$$

and so applying the inductive hypothesis to bound

$$t_A^{-1}\|A_{k-1} - A_\star\|_F^2 + \|b_{k-1} - b_\star\|_2^2 \leq t_A^{-1}\|I - A_\star\|_F^2 + \|b_\star\|_2^2,$$

we obtain for all i

$$\begin{aligned} \|\delta_i\|_2 &\leq \sqrt{2} \times \sqrt{t_A^{-1}\|A_\star - I\|_F^2 + \|b_\star\|_2^2} \times \max\left\{t_A^{1/2}\|u_i\|_2, 1\right\} \\ &\leq \sqrt{2} \times \sqrt{t_A^{-1}\|A_\star - I\|_F^2 + \|b_\star\|_2^2} \times \frac{\sqrt{c}\|U\|_{\ell^1 \to \ell^2}}{\|U\|_{\ell^2 \to \ell^2}}. \end{aligned} \tag{36}$$

Since

$$\psi(s) = \frac{1}{4\pi\sigma^2}\left(1 - \exp\left(-\frac{s}{2\sigma^2}\right)\right),$$

we have

$$\dot{\psi}(s) = \frac{1}{8\pi\sigma^4}\exp\left(-\frac{s}{2\sigma^2}\right),$$

and for all $s \geq 0$

$$\left|1 - \nu\dot{\psi}(s)\right| = \left|1 - 8\pi\sigma^4\dot{\psi}(s)\right| \leq \frac{s}{2\sigma^2}$$

by the standard exponential convexity estimate. Plugging in our bound (36), we obtain for all i

$$\left|1 - \nu\dot{\psi}(\tfrac{1}{2}\|\delta_i\|_2^2)\right| \leq \frac{t_A^{-1}\|A_\star - I\|_F^2 + \|b_\star\|_2^2}{2\sigma^2} \times \frac{c\|U\|_{\ell^1 \to \ell^2}^2}{\|U\|_{\ell^2 \to \ell^2}^2}.$$

Under our choice of t_A and hypotheses on σ, this is bounded by $\frac{1}{4\kappa}$. □