

Analysis of Constrained Optimization Variants of the Map-Seeking Circuit Algorithm

S. R. Harker, C. R. Vogel, and T. Gedeon*

May 28, 2007

Abstract

The map-seeking circuit algorithm (MSC) was developed by Arathorn to efficiently solve the combinatorial problem of correspondence maximization, which arises in applications like computer vision, motion estimation, image matching, and automatic speech recognition [D. W. Arathorn, *Map-Seeking Circuits in Visual Cognition: A Computational Mechanism for Biological and Machine Vision*, Stanford University Press, 2002]. Given an input image, a template image, and a discrete set of transformations, the goal is to find a composition of transformations which gives the best fit between the transformed input and the template. We imbed the associated combinatorial search problem within a continuous framework by using superposition, and we analyze a resulting constrained optimization problem. We present several numerical schemes to compute local solutions, and we compare their performance on a pair of test problems: an image matching problem and the challenging problem of automatically solving a Rubik's cube.

Keywords: correspondence maximization, constrained optimization, map seeking circuit

AMS 65Y20, 90C27, 92-08.

1 Introduction

The map-seeking circuit (MSC) algorithm was recently developed by David Arathorn to efficiently solve difficult problems in visual cognition [2, 3].

*Department of Mathematical Sciences, Montana State University, Bozeman, MT 59717-2400

Arathorn and his colleagues have applied MSC to a range of problems including recognition of 3-dimensional objects from 2-dimensional projections [7], automatic target recognition in cluttered fields [8], motion estimation [16], and limb inverse kinematics (used in robotics to optimally position a robot’s arm in a cluttered environment) [15]. We will demonstrate in this paper that MSC can also be applied to the somewhat more whimsical, but quite difficult problem of automatically solving a Rubik’s cube [11].

In this paper we present a rigorous analysis of certain variants of the MSC algorithm. This analysis is based on the concept of *correspondence maximization*. By this we mean the selection of a single transformation, from a given class of transformations, which most closely matches an “input image” to a “template image”. The notion of “image” is application dependent. For example, it could be a conventional 2-dimensional image or its 3-dimensional analogue, the position of a robot arm, or the orientation of a Rubik’s cube.

We restrict our attention to transformations which can be decomposed into a finite product, where each term in the product is a linear transformation on the space of images. For instance, in simple two-dimensional image matching one might consider products of rotations, horizontal translations, and vertical translations. We also assume that the set of component transformations is discrete. For example, we might consider integer pixel translations and discrete rotations of 5 degrees, 10 degrees, 15 degrees, etc. For inherently discrete applications like solving a Rubik’s cube, this imposes no artificial restrictions. In other applications it may be necessary to employ a multilevel strategy, e.g., MSC may be applied to obtain a good image match at pixel resolution. To obtain sub-pixel resolution one might again apply MSC but with a refined set of transformations. Alternatively, one might apply a more conventional optimization-based method with an initial guess generated by MSC.

Given a finite composition of discrete linear transformations, the correspondence maximization problem can be solved by a brute-force search of the transformation space. Suppose the transformations can be parameterized as

$$T = T_{i_L}^{(L)} \circ \dots \circ T_{i_\ell}^{(\ell)} \circ \dots \circ T_{i_2}^{(2)} \circ T_{i_1}^{(1)}, \quad (1)$$

where $1 \leq i_\ell \leq n_\ell$ for $\ell = 1, \dots, L$. Then a brute-force search would require the evaluation of $n_1 n_2 \dots n_L$ transformations. Such an approach becomes intractable when this product is large. On the other hand, MSC is an iterative algorithm whose dominant cost per iteration is proportional to

evaluation of transformations whose number is the sum

$$n_s \stackrel{\text{def}}{=} n_1 + n_2 + \dots + n_L. \quad (2)$$

A key insight in Arathorn’s development of MSC was that the discrete set of transformations could be imbedded in a continuous framework using *superposition*, i.e., by taking linear combinations

$$T_{(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)})} \stackrel{\text{def}}{=} \sum_{i_L=1}^{n_L} x_{i_L}^{(L)} T_{i_L}^{(L)} \left(\dots \left(\sum_{i_2=1}^{n_2} x_{i_2}^{(2)} T_{i_2}^{(2)} \left(\sum_{i_1=1}^{n_1} x_{i_1}^{(1)} T_{i_1}^{(1)} \right) \right) \dots \right), \quad (3)$$

where the coefficients $x_{i_\ell}^{(\ell)}$ are real-valued. Arathorn’s work was inspired by the physiological structure of the visual cortex of the brain [1]. At the same time the structure of the algorithm inspired him to propose a new theory of the function of the visual cortex, that is based on essential use of the bi-directional connections between the cortices.

In Arathorn’s view successive functional areas along the pathways of the visual cortices implement the transformations $T_{i_\ell}^{(\ell)}$. The activity of the reciprocal connections between layers corresponds to the coefficients $x_{i_\ell}^{(\ell)}$. Neural computations are performed by selectively pruning the activity of these connections using competition between different pathways. Pruning of the activity of connections is represented in MSC by an iterative procedure that competitively updates the coefficients $x_{i_\ell}^{(\ell)}$.

A first attempt at a rigorous mathematical analysis of the MSC algorithm was presented in Gedeon and Arathorn [10]. These authors viewed MSC as a fixed point iteration on the phase space of all admissible values of the coefficients $x_{i_\ell}^{(\ell)}$. Their analysis relied on a Lyapunov function to establish generic convergence of the iterations. They showed that the coefficients $x_{i_\ell}^{(\ell)}$ converge either to the zero vector or to a scalar multiple of a canonical basis vector \mathbf{e}_{i^*} . (This has components $[\mathbf{e}_{i^*}]_j = 1$ if $j = i^*$ and $[\mathbf{e}_{i^*}]_j = 0$ otherwise.) Convergence to the zero vector can be interpreted as “no correspondence was found that maps the input to the template”. Convergence to a multiple of \mathbf{e}_{i^*} indicates that the ℓ th component transformation in the decomposition (1) should be taken to be $T_{i^*}^{(\ell)}$.

The fixed point analysis in [10] failed to provide a concise characterization of the limit of the MSC fixed point iteration. In order to obtain such a characterization, we take a constrained optimization viewpoint in this paper. To this end we define a cost functional with which to quantify the notion of correspondence. Suppose we have a fixed input image I and a

fixed template M , which lie in Hilbert spaces \mathcal{H} and \mathcal{H}' , respectively. Given a transformation $T : \mathcal{H} \rightarrow \mathcal{H}'$, we define the correspondence between I and M associated with the transformation T to be

$$c(T) \stackrel{\text{def}}{=} \langle T(I), M \rangle, \quad (4)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product on \mathcal{H}' . From the discrete representation (1) we obtain the *correspondence array*

$$c(i_1, i_2, \dots, i_L) = \langle T_{i_L}^{(L)} \circ \dots \circ T_{i_2}^{(2)} \circ T_{i_1}^{(1)}(I), M \rangle \quad (5)$$

From the multi-index which maximizes the correspondence array,

$$(i_1^*, i_2^*, \dots, i_L^*) = \arg \max_{i_1, i_2, \dots, i_L} c(i_1, i_2, \dots, i_L), \quad (6)$$

we obtain the optimal transformation

$$T^* = T_{i_L^*}^{(L)} \circ \dots \circ T_{i_2^*}^{(2)} \circ T_{i_1^*}^{(1)} \quad (7)$$

which maximizes $c(T)$ in eqn. (4) over the class of transformations having representation (1).

Brute-force computation of the maximizing multi-index (6) requires assembly of the L -dimensional correspondence array c in (5), which again requires evaluation of all $n_1 \dots n_L$ possible combinations of transformations. To avoid this we employ Arathorn's superposition idea and define the continuous *MSC correspondence functional*,

$$c_{\text{MSC}}(\mathbf{x}_1, \dots, \mathbf{x}_L) = \langle T_{(\mathbf{x}_1, \dots, \mathbf{x}_L)}(I), M \rangle, \quad (8)$$

where $T_{(\mathbf{x}_1, \dots, \mathbf{x}_L)}$ is given in (3). We then seek to maximize this functional subject to the following constraints: For each $\ell = 1, \dots, L$,

$$\sum_{i=1}^{n_\ell} x_i^{(\ell)} = 1, \quad (9)$$

and

$$x_i^{(\ell)} \geq 0 \quad \text{for } i = 1, \dots, n_\ell. \quad (10)$$

One of the goals of this paper is to analyze the structure of the continuous constrained optimization problem of maximizing c_{MSC} in (8) subject to (9)-(10) and relate it the discrete problem (6). We will prove in Section 3 that we obtain a global solution to the constrained optimization problem by taking

$$(\mathbf{x}_*^{(1)}, \dots, \mathbf{x}_*^{(L)}) = (\mathbf{e}_{i_1^*}, \dots, \mathbf{e}_{i_L^*}), \quad (11)$$

where each $\mathbf{e}_{i_\ell^*}$ is a canonical basis vector,

$$[\mathbf{e}_{i_\ell^*}]_j = \delta_{i_\ell^*, j}, \quad j = 1, \dots, n_\ell,$$

and the i_ℓ^* are the maximizing indices in (6). We will also prove that local constrained maximizers must lie at vertices of the polytope defined by the constraint set.

A second goal is to present numerical methods to efficiently solve the constrained optimization problem. This goal is addressed in section 4. In subsection 4.1 we employ adjoint techniques to show that the gradient of the MSC correspondence functional (8) can be computed at the cost of only twice n_s transformations, where n_s is defined in (2). This facilitates the use of gradient-based constrained optimization techniques like the projected gradient method and various quasi-Newton methods [14].

A third goal is to examine Arathorn’s MSC algorithm in the context of our constrained optimization framework. In subsection 4.3 we discuss a particular implementation of MSC which makes use of the gradient of the MSC cost functional (8). However, MSC apparently cannot be viewed simply as a variant of the projected gradient method. MSC does maintain the inequality constraints (10), but it does not maintain the equality constraints (9). We will show that even if one rescales the MSC iterates so the equality constraints are satisfied, the rescaled iterates need not monotonically increase the MSC cost functional. This is in marked contrast to the projected gradient iterates, which maintain both sets of constraints and monotonically increase the cost functional.

It should be noted that Arathorn previously derived order n_s techniques to evaluate basic building blocks of the MSC algorithm, which he refers to as q-vectors in his book [3]. These q-vectors are identical to the gradient vectors for the MSC correspondence functional c_{MSC} . In Arathorn’s papers [4, 7] and [8], c_{MSC} is denoted by Q . In the first two of these papers he makes explicit reference to the gradient of Q as well as to the correspondence array (see equation (5)), which he denotes by $c(\mathbf{j})$. Throughout Arathorn’s publications the “gain vectors”, which he denotes by boldface g , coincide with our coefficient vectors $\mathbf{x}^{(\ell)}$.

In order to motivate the analysis, we present a pair of illustrative examples in section 2. The first is a 2-dimensional image matching test problem, and the second involves the automatic solution of a Rubik’s cube. These provide test problems for the various algorithms in the numerical study presented in section 5. They also illustrate the power of the MSC approach to handle problems in cognition, or “higher-level vision”. In particular, the

image matching problem involves the selection of one object among several objects in a cluttered, noisy input image, as well as the selection of a transformation which maps that object onto the template.

We close with a summary and some conclusions in section 6.

2 Illustrative Examples

2.1 Discrete Image Matching

We present a simple image matching problem to illustrate the discrete layer-wise decomposition of transformations (1) and the correspondence functional (4). In Fig. 1 we show a 100 pixel \times 110 pixel binary input image I and a 23 pixel \times 23 pixel binary template image M . A visual comparison of subfigures (a) and (b) reveals that the input I contains a rotated version of the template M centered near pixel location $i = 50, j = 55$. This comparison process can be broken down into a pair of steps: First extract a 23×23 pixel subimage (a rotated version of the template M) from the input I , and then rotate the subimage (by about 60 degrees clockwise) so that it matches M . We can decompose the subimage extraction into two steps: (i) we first extract a long, skinny 23×110 subimage from I ; and (ii) we then extract a 23×23 sub-subimage from the subimage in step (i). Step (iii) is then to perform rotations on the 23×23 subimage.

In order to apply the optimization framework outlined in section 1, we place the image I in the Hilbert space $\mathcal{H} = R^{100 \times 110}$ with Frobenius inner product

$$\langle I, J \rangle_{100 \times 110} = \sum_{i=1}^{100} \sum_{j=1}^{110} I(i, j) J(i, j). \quad (12)$$

Similarly the template M can be placed in $\mathcal{H}' = R^{23 \times 23}$ with analogous inner product. Admissible transformations can be parameterized as

$$T_{i,j,k} = T_k^{(3)} \circ T_j^{(2)} \circ T_i^{(1)}. \quad (13)$$

Here $T_i^{(1)}$ extracts a 23×110 pixel subimage from I ,

$$T_i^{(1)}(I) = \{I(i + i', j) \mid 0 \leq i' \leq 23, 1 \leq j \leq 110\}.$$

for $i = 1, \dots, 78$. Similarly, $T_j^{(2)}$ extracts a 23×23 pixel subimage from $T_i^{(1)}(I)$, and j ranges from 1 to 88.

We apply discrete rotations of θ_k radians, with $\theta_k = (k - 1)\pi/40$, for $k = 1, \dots, 40$. Given a discrete 23×23 image J , we take

$$T_k^{(3)} J(x_i, y_j) = \text{interp } J(R(\theta_k)(x_i, y_j)).$$

Here $R(\theta_k)$ represents rotation by θ_k radians,

$$R(\theta_k)(x, y) = \begin{bmatrix} \cos(\theta_k) & \sin(\theta_k) \\ -\sin(\theta_k) & \cos(\theta_k) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},$$

and $\text{interp } J$ means that we interpolate values of J from rotated grid points to regular grid points (x_i, y_j) using bilinear interpolation.

The correspondence array c , defined in eqn (5), associated with the decomposition (13) has size $78 \times 88 \times 40$. In Fig. 1 we display 78×88 slices of this array for fixed indices $k = 1$ and $k = 21$. Our goal will be to find the largest entry of this array. The indices of this entry characterize the transformation which best matches the input to the template.

Gradient computations will require adjoints of the transformations $T_{i_\ell}^{(\ell)}$, which we denote by $U_{i_\ell}^{(\ell)}$. One can easily verify that $U_j^{(2)}$ imbeds a 23×23 image M in a 23×110 array by zero filling,

$$[U_j^{(2)}(M)](i, j') = \begin{cases} M(i, j'), & j \leq j' \leq j + 87, \\ 0, & \text{otherwise.} \end{cases}$$

The adjoint $U_i^{(1)}$ of $T_i^{(1)}$ has an analogous representation; it imbeds a 23×110 array in a 100×110 array by zero filling. The adjoint $U_k^{(3)}$ of the rotation operator $T_k^{(3)}$ involves rotation by $-\theta_k$ radians and interpolation.

2.2 Rubik's Cube

The Rubik's cube [11] is a permutation puzzle involving 27 small cubes, or "cubies", arranged in a $3 \times 3 \times 3$ pattern. There are 6 distinct colored stickers affixed to the faces of each cubie; in the solved position each side of the Rubik's cube is a solid color. The cube is constructed in such a way that the 9 cubies that compose each side can be rotated by 90, 180, and 270 degrees and these rotations, or "moves", can be composed. After application of even a short sequence of such moves the color pattern of the cube can be quite complex.

The hidden central cubie and the six middle cubies on the sides of the Rubik's cube never change position relative to each other, so we consider them fixed in space. This reduces the puzzle to considering the remaining 20

cubies: 12 edge cubies, each with two exposed colors, and 8 corner cubies, each with 3 exposed colors.

Each corner cubie can occur in any one of eight different corner locations on the larger cube, and for each corner location there are three possible orientations of such a cubie. Thus there are $8 \times 3 = 24$ states that a given corner cubie may be in. Similarly, each edge cubie can occur in one of twelve locations, and for each location there are two possible orientations. Therefore there are also 24 possible states for each of the edge cubies. For the purposes of the computation, we represent the state of each cubie, both corner and edge, as a canonical basis vector in R^{24} . For the corner cubie we can think of 24 coordinates divided into 8 groups of 3; as an example, if a particular cubie is represented by the canonical basis vector e_7 that means that the cubie is in the position 3 in the orientation 1. Obviously, the assignment of numbers to positions and orientations is arbitrary, but fixed. Similarly for edge cubies the 24 coordinates are divided into 12 “position” groups of 2 “orientation” coordinates. Since there are 20 cubies whose position is not fixed in space, the entire Rubik’s cube can be represented as a vector of zeroes and ones in R^{480} , with exactly 20 ones. When applying our computational framework, we set M to be such a representation of the fully solved cube and I to be a representation of the cube that we want to solve.

Now we describe the linear maps in each layer. There are six basic moves which represents clockwise 90 degree rotations of each of the six sides of the Rubik’s cube. We denote them by u, d, f, b, l, r for turning clockwise the “up,” “down,” “front,” “back,” “left,” and “right” sides. All more complex moves can be generated by composition of these six moves. In addition, we will consider a 7th “z”, or “zero” move, which simply leaves all the cubies in place.

We index the moves from the set $\{u, d, f, b, l, r, z\}$ to obtain the transformations $T_i^{(\ell)}$, $i = 1, \dots, 7 = n_\ell$. Unlike in the previous pattern matching example, the transformations $T_i^{(\ell)}$ remain the same for every layer. The number of layers L corresponds to the number of compositions of the basic moves used to rearrange the cube. With one exception, the adjoint transformations $U_i^{(\ell)}$ correspond to counter-clockwise 90 degree rotations of the six sides of the cube. The identity transformation, which corresponds to the z move, is its own adjoint. Since we represent the state of the Rubik’s cube by a vector in R^{480} , we must represent each transformation $T_i^{(\ell)}$ by a 480×480 matrix. The structure of these matrices is very simple. Since for each cubie we have to specify its new position and orientation, the matrix will be block diagonal with 24×24 blocks along the diagonal. Each of these

blocks will be a permutation matrix, which sends the number 1 in position i in the 24-dimensional vector specifying the current state, to 1 in position j , specifying the new state.

To summarize, we start with a solved Rubik’s cube, apply randomly L moves and represent the resulting cube in R^{480} . This is our “input image” I . The “template image” M is a representation of the solved cube. We seek to find a composition of L transformations that will transform the vector I to the vector M . Each layer contains the same seven transformations, which represent all legal moves on the cube.

3 Analysis of the Constrained Optimization Problem

In this section we analyze the structure of the problem of maximizing the MSC cost functional (8) subject to the constraints (9)-(10). Let Ω denote the set of vectors in $R^{n_1} \times \dots \times R^{n_L}$ which satisfy these constraints. Ω can be viewed geometrically as a polytope, i.e., the convex hull of a finite set of points which are the vertices of the polytope. The vertices of Ω consist of all vectors of the form $(\mathbf{e}_{i_1}, \dots, \mathbf{e}_{i_L})$, where \mathbf{e}_{i_ℓ} denotes the i_ℓ th canonical basis vector for R^{n_ℓ} , with entries $[\mathbf{e}_{i_\ell}]_j = 1$ if $i_\ell = j$ and 0 otherwise. We define (i_1, \dots, i_L) to be the *associated multi-index* for the vertex $(\mathbf{e}_{i_1}, \dots, \mathbf{e}_{i_L})$.

The following lemma shows that the MSC cost functional is multilinear, i.e., it is linear in each of the layer coefficient vectors $\mathbf{x}^{(\ell)}$. This follows immediately from the linearity of each of the component transformations $T_i^{(\ell)}$ in (8) and the bilinearity of the inner product.

Lemma 1

$$c_{\text{MSC}}(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)}) = \sum_{i_1=1}^{n_1} \dots \sum_{i_L=1}^{n_L} c(i_1, \dots, i_L) x_{i_1}^{(1)} \dots x_{i_L}^{(L)} \quad (14)$$

Our next result shows that there is a vertex of Ω which is a global solution to the constrained maximization problem. The associated multi-index is then a maximizing multi-index in the sense of (6). As an immediate corollary, if the correspondence array c in (5) has a unique maximal entry, then the vertex with the associated multi-index of the maximal entry of the correspondence array is the unique global solution to the constrained maximization problem.

Theorem 2 *There exist canonical basis vectors $\mathbf{e}_{i_\ell^*}$, $\ell = 1, \dots, L$, for which*

$$c_{\text{MSC}}(\mathbf{e}_{i_1^*}, \dots, \mathbf{e}_{i_L^*}) \geq c_{\text{MSC}}(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)}) \quad (15)$$

for all $(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)}) \in \Omega$.

Proof. Assume by way of contradiction that there is no vertex satisfying (15). Then there exists a layer ℓ whose associated maximizing layer vector $\mathbf{x}_*^{(\ell)}$ is not a canonical basis vector. Assume without loss of generality that $\ell = 1$, and denote the remaining maximizing layer vectors by $\mathbf{x}_*^{(2)}, \dots, \mathbf{x}_*^{(L)}$. To simplify notation, let $\mathbf{x}_*^{(1)}$ have components x_1, \dots, x_{n_1} . By Lemma 1,

$$\begin{aligned} c_{\text{MSC}}(\mathbf{x}_*^{(1)}, \mathbf{x}_*^{(2)}, \dots, \mathbf{x}_*^{(L)}) &= \sum_{i=1}^{n_1} \underbrace{\left(\sum_{i_2=1}^{n_2} \cdots \sum_{i_L=1}^{n_L} c(i_1, i_2, \dots, i_L) x_{i_2}^{(2)} \cdots x_{i_L}^{(L)} \right)}_{\bar{c}_i} x_i \\ &\leq \sum_{i=1}^{n_1} \bar{c}_i [\mathbf{e}_{i_1^*}]_i, \end{aligned} \quad (16)$$

where $i_1^* = \arg \max_i \bar{c}_i$. Thus

$$c_{\text{MSC}}(\mathbf{x}_*^{(1)}, \mathbf{x}_*^{(2)}, \dots, \mathbf{x}_*^{(L)}) \leq c_{\text{MSC}}(\mathbf{e}_{i_1^*}, \mathbf{x}_*^{(2)}, \dots, \mathbf{x}_*^{(L)}).$$

Now repeat the same argument for each of the remaining $L - 1$ layers whose associated maximizing component vector $\mathbf{x}_*^{(\ell)}$ is not a canonical basis vector. In this way we construct a vertex $(\mathbf{e}_{i_1^*}, \mathbf{e}_{i_2^*}, \dots, \mathbf{e}_{i_L^*})$ that satisfies (15). This contradiction completes the proof.

Corollary 3 *If the correspondence array c has a unique maximizing multi-index (i_1^*, \dots, i_L^*) , then the vertex $(\mathbf{e}_{i_1^*}, \dots, \mathbf{e}_{i_L^*})$ is the unique constrained maximizer for c_{MSC} .*

In order to better understand local constrained maxima, we next examine the equality constrained problem

$$\max c_{\text{MSC}}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(L)}) \quad \text{subject to} \quad \sum_{i_\ell=1}^{n_\ell} x_{i_\ell}^{(\ell)} = 1, \quad \ell = 1, \dots, L. \quad (17)$$

We show that nondegenerate stationary points for (17) are “saddle-like” in the sense that the reduced Hessian at such points has eigenvalues with mixed signs.

Components of vectors in the feasible set for (17) can be represented as

$$\mathbf{x}^{(\ell)} = \bar{\mathbf{x}}^{(\ell)} + Z^{(\ell)}\mathbf{s}^{(\ell)}, \quad \ell = 1, \dots, L, \quad (18)$$

where $\bar{\mathbf{x}}^{(\ell)} \in R^{n_\ell}$ is fixed and satisfies $\sum_i \bar{x}_i^{(\ell)} = 1$, $\mathbf{s}^{(\ell)} \in R^{n_\ell-1}$, and the $n_\ell - 1$ columns $\mathbf{z}_i^{(\ell)}$ of $Z^{(\ell)}$ form a basis for $\{\mathbf{x} \in R^{n_\ell} \mid \sum_i x_i = 0\}$. Now consider the reduced cost functional

$$c_{\text{reduced}}(\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(L)}) = c_{\text{MSC}}(\bar{\mathbf{x}}^{(1)} + Z^{(1)}\mathbf{s}^{(1)}, \dots, \bar{\mathbf{x}}^{(L)} + Z^{(L)}\mathbf{s}^{(L)}). \quad (19)$$

Lemma 4 *The Hessian of c_{reduced} has an $L \times L$ block representation, where the block diagonal components are $(n_\ell - 1) \times (n_\ell - 1)$ zero matrices.*

Proof. By Lemma 1, c_{MSC} is linear in each of the layer vectors $\mathbf{x}^{(\ell)}$. Thus the Hessian of c_{reduced} has an $L \times L$ block representation whose diagonal blocks are $n_\ell \times n_\ell$ zero matrices. By the chain rule,

$$\frac{\partial^2 c_{\text{reduced}}}{\partial s_i^{(\ell)} \partial s_j^{(\ell)}} = \left(\mathbf{z}_i^{(\ell)}\right)^T [\text{Hess } c_{\text{MSC}}]_{\ell\ell} \mathbf{z}_j^{(\ell)} = 0,$$

where $[\text{Hess } c_{\text{MSC}}]_{\ell\ell}$ denotes the ℓ th diagonal block of $\text{Hess } c_{\text{MSC}}$ and $\mathbf{z}_i^{(\ell)}$ is the i th column of $Z^{(\ell)}$.

We say that a critical point \mathbf{s} for c_{reduced} is *nondegenerate* if $\text{Hess } c_{\text{reduced}}(\mathbf{s})$ is not the zero matrix. We call a nondegenerate critical point *saddle-like* if $\text{Hess } c_{\text{reduced}}(\mathbf{s})$ has eigenvalues of mixed signs. As a consequence of the above lemma, we obtain the following results.

Theorem 5 *Any nondegenerate critical point for c_{reduced} is saddle-like.*

Proof. From Lemma 4, the trace (i.e., the sum of the diagonal entries) of $\text{Hess } c_{\text{reduced}}(\mathbf{s})$ is zero. But the trace of a matrix is equal to sum of its eigenvalues. $\text{Hess } c_{\text{reduced}}(\mathbf{s})$ is symmetric, so it has real eigenvalues. At least one of these is nonzero if \mathbf{s} is a nondegenerate critical point.

Corollary 6 *If \mathbf{x} is a local constrained maximizer for c_{MSC} and the reduced Hessian associated with the equality constraints does not vanish, then \mathbf{x} must lie on the boundary of the constraint set Ω .*

Example. In the 2-layer case the MSC cost functional (8) has a bilinear representation

$$\begin{aligned} c_{\text{MSC}}(\mathbf{x}, \mathbf{y}) &= \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} c_{ij} x_i y_j \\ &= \mathbf{x}^T C \mathbf{y} \end{aligned} \tag{20}$$

$$= \frac{1}{2} \begin{bmatrix} \mathbf{x}^T & \mathbf{y}^T \end{bmatrix} \begin{bmatrix} 0 & C \\ C^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \tag{21}$$

where the $n_1 \times n_2$ matrix C has correspondence array entries c_{ij} given in (5). We consider three specific cases generated by 2×2 matrices

$$C_1 = \begin{bmatrix} 1 & -2 \\ 0 & -1 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 2 & -2 \\ -2 & 1 \end{bmatrix}, \quad C_3 = \begin{bmatrix} 1 & -2 \\ 0 & 0 \end{bmatrix}.$$

Here $L = 2$, $n_1 = n_2 = 2$, and the constraints (9)-(10) reduce to $x_1 + x_2 = y_1 + y_2 = 1$ and $x_i \geq 0$, $y_i \geq 0$ for $i = 1, 2$. We will show that in these three cases the correspondence functional has, respectively, (a) a single (local and global) constrained maximizer; (b) a pair of distinct local constrained maximizers; and (c) an entire line segment consisting of local constrained maximizers.

Proceeding as in (18) we take $\bar{\mathbf{x}}^{(\ell)} = \mathbf{e}_1 = [1, 0]^T$ and $Z^{(\ell)} = \mathbf{z} = [-1, 1]^T$ for $\ell = 1, 2$, and we set $s = s^{(1)}$ and $t = s^{(2)}$. We can then represent the reduced cost functional associated with matrices C_k for $k = 1, 2, 3$ as

$$\begin{aligned} f_k(s, t) &= (\mathbf{e}_1 + s\mathbf{z})^T C_k (\mathbf{e}_1 + t\mathbf{z}) \\ &= \mathbf{z}^T C_k \mathbf{z} s t + \mathbf{z}^T C_k \mathbf{e}_1 s + \mathbf{e}_1^T C_k \mathbf{z} t + \text{const.} \end{aligned}$$

As (s, t) vary across the unit square, $\mathbf{x} = \mathbf{e}_1 + s\mathbf{z}$ and $\mathbf{y} = \mathbf{e}_1 + t\mathbf{z}$ range over the constraint set Ω .

Fig. 2 shows contour plots and gradient fields of f_k for each case. The critical points (where the gradient vanishes) for f_k , $k = 1, 2, 3$, are $\mathbf{s}_1 = (3/2, 1/2)$, $\mathbf{s}_2 = (4/7, 4/7)$, $\mathbf{s}_3 = (1, 1/3)$, respectively. For each k , the global maximizer of f_k constrained to the unit square is $(s, t) = (0, 0)$. This reflects the fact that in each case, the vertex $\mathbf{x} = \mathbf{y} = \mathbf{e}_1$ solves the constrained maximization problem and the (1, 1) entry of the matrix C_k (and the correspondence array) is maximal. For the case $k = 1$, $(s, t) = (0, 0)$ is the only (local and global) constrained maximizer. For the case $k = 2$, f_k has a second local constrained maximizer at $(s, t) = (1, 1)$. For $k = 3$, each point along the line segment $s = 1$, $1/3 < t \leq 1$, is a local constrained maximizer of f_k .

The critical points \mathbf{s}_k of f_k for cases $k = 2$ and $k = 3$ are indicated by asterisks (*) in subplots (b) and (c) of Fig. 2. The dashed lines in subplots (b) and (c) are the stable and unstable manifolds of the critical points and are determined by the eigenvectors of the Hessian of f_k evaluated at \mathbf{s}_k . The unstable manifold, which is determined by the eigenvector corresponding to the positive eigenvalue, divides the plane into basins of attraction of local constrained maximizers. Hence, it is known as a *separatrix*.

To “flow along the projected gradient field”, means to move along a trajectory of the ordinary differential equation (ODE)

$$\frac{d\mathbf{x}}{dt} = P_{\Omega} \mathbf{g}_k(\mathbf{x}(t)). \quad (22)$$

The right hand side of this ODE is the gradient of f_k projected onto the constraint set Ω and is precisely defined in Section 4.2. Starting from any point not on the separatrix, such a flow will lead to a local constrained maximizer of f_k . For instance, the separatrix in Fig. 2 (b) is the dashed line from $(s, t) = (1/7, 1)$ to $(1, 1/7)$. Starting points to the left and below this line all flow into the local constrained maximizer $(s, t) = (1, 1)$ (in the lower right corner) for the reduced objective function f_2 , which is associated with the vertex $\mathbf{x}^{(1)} = \mathbf{x}^{(2)} = (0, 1)$ for f_2 . Starting points to the right and above the separatrix flow to the global constrained maximizer $(s, t) = (0, 0)$ for f_2 , which is associated with vertex $\mathbf{x}^{(1)} = \mathbf{x}^{(2)} = (1, 0)$ for f_2 . In Fig. 2 (c), starting points to the right and above the separatrix flow to $(s, t) = (0, 0)$. Starting points to below and to the right of separatrix flow to the line segment $s = 1$, $1/3 < t \leq 1$. On the other hand, in Fig. 2 (a), the entire unit square lies in the same basin of attraction for f_1 . Any starting point will flow to the global constrained maximizer $(s, t) = (0, 0)$.

4 Numerical Methods

One of the keys to the efficient implementation of numerical optimization methods is the fast computation of gradients.

4.1 Adjoint Gradient Computations

We assume the transformations $T_i^{(\ell)}$ map the Hilbert space \mathcal{H}_{ℓ} into Hilbert space $\mathcal{H}_{\ell+1}$. Let $U_i^{(\ell)} : \mathcal{H}_{\ell+1} \rightarrow \mathcal{H}_{\ell}$ denote the adjoint operator, characterized by

$$\langle T_i^{(\ell)}(I), J \rangle_{\mathcal{H}_{\ell+1}} = \langle I, U_i^{(\ell)}(J) \rangle_{\mathcal{H}_{\ell}}, \quad I \in \mathcal{H}_{\ell}, J \in \mathcal{H}_{\ell+1}.$$

We define the layer-wise superposition

$$T_{\mathbf{x}^{(\ell)}}^{(\ell)} = \sum_{i=1}^{n_\ell} x_i^{(\ell)} T_i^{(\ell)}$$

and its adjoint,

$$U_{\mathbf{x}^{(\ell)}}^{(\ell)} = \sum_{i=1}^{n_\ell} x_i^{(\ell)} U_i^{(\ell)}.$$

Then

$$\begin{aligned} c_{\text{MSC}}(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)}) &\stackrel{\text{def}}{=} \langle T_{\mathbf{x}^{(L)}}^{(L)} \circ \dots \circ T_{\mathbf{x}^{(\ell)}}^{(\ell)} \circ \dots \circ T_{\mathbf{x}^{(1)}}^{(1)}(I), M \rangle_{\mathcal{H}_{L+1}} & (23) \\ &= \langle T_{\mathbf{x}^{(\ell)}}^{(\ell)} \circ \dots \circ T_{\mathbf{x}^{(1)}}^{(1)}(I), U_{\mathbf{x}^{(\ell+1)}}^{(\ell+1)} \circ \dots \circ U_{\mathbf{x}^{(L)}}^{(L)}(M) \rangle_{\mathcal{H}_{\ell+1}} \\ &= \sum_{i=1}^{n_\ell} x_i^{(\ell)} \langle T_i^{(\ell)} \circ \dots \circ T_{\mathbf{x}^{(1)}}^{(1)}(I), U_{\mathbf{x}^{(\ell+1)}}^{(\ell+1)} \circ \dots \circ U_{\mathbf{x}^{(L)}}^{(L)}(M) \rangle_{\mathcal{H}_{\ell+1}} \end{aligned} \quad (24)$$

Consequently,

$$\frac{\partial c_{\text{MSC}}}{\partial x_i^{(\ell)}} = \langle I_i^{(\ell)}, M^{(\ell)} \rangle_{\mathcal{H}_{\ell+1}}, \quad (25)$$

where for $i = 1, \dots, n_\ell$,

$$I_i^{(\ell)} \stackrel{\text{def}}{=} T_i^{(\ell)} \circ T_{\mathbf{x}^{(\ell-1)}}^{(\ell-1)} \circ \dots \circ T_{\mathbf{x}^{(1)}}^{(1)}(I) \quad (26)$$

and

$$M^{(\ell)} \stackrel{\text{def}}{=} U_{\mathbf{x}^{(\ell+1)}}^{(\ell+1)} \circ \dots \circ U_{\mathbf{x}^{(L)}}^{(L)}(M). \quad (27)$$

In the discussion to follow, we indicate the computation of the gradient of c_{MSC} at $\mathbf{x} = (\mathbf{x}^{(L)}, \dots, \mathbf{x}^{(1)})$ by

$$\mathbf{g} \leftarrow \text{GRADIENT}(\mathbf{x}).$$

Using formulas (25)-(27), this operation can be carried out by evaluating only $2(n_1 + \dots + n_L)$ transformations. Additional costs are incurred in computing linear combinations and inner products. If only the MSC cost function is desired, the number of transformations can be cut in half and only one inner product is needed; see eqn. (23).

4.2 Projected Gradient Method

The (orthogonal) projection of $\mathbf{x} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)})$ onto the constraint set Ω is given by

$$P_{\Omega}(\mathbf{x}) = \arg \min_{\mathbf{z} \in \Omega} \|\mathbf{z} - \mathbf{x}\|.$$

We denote the approximate solution at iteration k by $\mathbf{x}(k)$. Idealized projected gradient iteration takes the following form:

```

for  $k = 1, 2, \dots$ 
   $\mathbf{g} \leftarrow \text{GRADIENT}(\mathbf{x}(k))$ 
   $\tau^* \leftarrow \arg \max_{\tau > 0} c_{\text{MSC}}(P_{\Omega}(\mathbf{x}(k) + \tau \mathbf{g}))$ 
   $\mathbf{x}(k+1) \leftarrow P_{\Omega}(\mathbf{x}(k) + \tau^* \mathbf{g})$ 
end

```

From Calamai and Moré [9], this idealized iteration is guaranteed to converge to a stationary point for our constrained maximization problem. Moreover, the active set will be identified in finitely many iterations. From results in section 3, in the generic case the iteration must then converge to a vertex which is a local constrained maximizer in finitely many steps.

Due to the multilinear nature of the cost function and the simplicity of the constraints (9)-(10), we can efficiently compute the parameter τ^* using the “bent line search” scheme in Section 4 of Moré and Toraldo [13]. Since the constrained maximizer lies on the boundary of the polytope Ω , as the parameter τ increases, the projection onto Ω of the path $\mathbf{x}(k) + \tau \mathbf{g}$ consists of a sequence of projections onto faces of the boundary of the polytope. This projected path is continuous and consists of a sequence of line segments; we explicitly compute the points at which the path bends.

4.3 A Map Seeking Circuit Algorithm

The iteration which we describe here is one of a broad class of schemes proposed by Arathorn [3]. In [10] Arathorn and Gedeon used dynamical systems techniques to analyze the convergence of this iteration. They assumed that the correspondence array c in (5) has nonnegative entries. This can be assured by adding an appropriate positive constant to each component of c .

When applied to correspondence maximization, the algorithm in [10] takes the following form.

```

 $\mathbf{x}(0) \leftarrow \mathbf{1}$ 
for  $k = 1, 2, \dots$ 
   $\mathbf{g} \leftarrow \text{GRADIENT}(\mathbf{x}(k))$       % gradient of  $c_{\text{MSC}}$  at  $\mathbf{x}(k)$ 
  for  $\ell = 1, \dots, n_\ell$ 
     $\mathbf{x}^{(\ell)}(k+1) \leftarrow \max\{\mathbf{0}, \mathbf{x}^\ell(k) - \kappa(\mathbf{1} - \mathbf{g}^{(\ell)} / \|\mathbf{g}^{(\ell)}\|_\infty)\}$ 
  end
end
end

```

Here $\mathbf{1}$ and $\mathbf{0}$ denote vectors whose components are all 1 and 0, respectively; κ is a positive “competition” parameter; $\|\mathbf{g}\|_\infty = \max_i |g_i|$; and $\max\{\cdot, \cdot\}$ indicates component-wise maximum.

Note that the MSC iteration maintains the nonnegativity constraints (10), but it does not maintain the equality constraints (9). The components of \mathbf{g} are always nonnegative because the components of c and of $\mathbf{x}(k)$ are nonnegative; see Lemma 1. Since the components of $\mathbf{1} - \mathbf{g}/\|\mathbf{g}\|_\infty$ are nonnegative, the function $\|\mathbf{x}\|_1 = \sum_i x_i$ decreases monotonically with each MSC iteration. This is the basis for the proof [10] that in the generic case the MSC iterations converge to some $(\mathbf{x}_*^{(1)}, \dots, \mathbf{x}_*^{(L)})$ where each $\mathbf{x}_*^{(\ell)}$ is either to a scalar multiple of a vertex, with scalar between 0 and 1, or a zero vector.

4.4 Comparison of Projected Gradient and MSC

Suppose $\mathbf{x} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)})$ lies in the interior of the constraint set Ω . Then for sufficiently small, positive τ , the projected gradient path can be represented as follows: For $\ell = 1, \dots, L$,

$$\mathbf{x}^{(\ell)}(\tau) = \mathbf{x}^{(\ell)} + \tau(\mathbf{g}^{(\ell)} - \text{mean}(\mathbf{g}^{(\ell)})\mathbf{1}). \quad (28)$$

Note that $\text{mean}(\mathbf{g}) = \mathbf{1}^T \mathbf{g} / \mathbf{1}^T \mathbf{1}$ is the projection of \mathbf{g} onto $\text{span}\{\mathbf{1}\}$. Similarly, if the MSC competition parameter κ is sufficiently small and positive, then the MSC path can be represented as

$$\mathbf{x}^{(\ell)}(\lambda) = \mathbf{x}^{(\ell)} + \lambda^{(\ell)}(\mathbf{g}^{(\ell)} - \|\mathbf{g}^{(\ell)}\|_\infty \mathbf{1}), \quad (29)$$

where $\lambda^{(\ell)} = \kappa / \|\mathbf{g}^{(\ell)}\|_\infty$.

In general, the “MSC correction” $\|\mathbf{g}\|_\infty$ is larger than “projected gradient correction” $\text{mean}(\mathbf{g})$, but one might expect that a rescaled version of the MSC path should behave like the projected gradient path. The analysis that we next present shows that this is not the case.

Consider the single-layer case ($L = 1$), where the cost functional can be expressed as

$$c_{\text{MSC}}(\mathbf{x}) = \mathbf{g}^T \mathbf{x},$$

where $\mathbf{x} \in R^n$ and $n = n_1$. Take the rescaled MSC update to be

$$\mathbf{y}(\lambda) = \frac{\mathbf{x}(\lambda)}{\|\mathbf{x}(\lambda)\|_1} = \frac{\mathbf{x} + \lambda(\mathbf{g} - \|\mathbf{g}\|_\infty \mathbf{1})}{\|\mathbf{x}\|_1 + \lambda(\|\mathbf{g}\|_1 - n\|\mathbf{g}\|_\infty)}. \quad (30)$$

Note that $\mathbf{y}(\lambda)$ satisfies the equality constraint (9). Since $\mathbf{y} = \mathbf{x}/\|\mathbf{x}\|_1$ lies in the interior of Ω for sufficiently small λ , the second equality in (30) holds and $\mathbf{y}(\lambda)$ also satisfies the nonnegativity constraint (10).

The projected gradient path is guaranteed to not decrease the cost functional, so

$$0 \leq \frac{d}{d\tau} c_{\text{MSC}}(\mathbf{x}(\tau))|_{\tau=0} = \mathbf{g}^T \frac{d}{d\tau} \mathbf{x}(\tau)|_{\tau=0},$$

where $\mathbf{x}(\tau)$ is given in (28). The analogous result for the rescaled MSC path need not hold. To see this, from (30), the chain rule, and the quotient rule,

$$\frac{d}{d\lambda} c_{\text{MSC}}(\mathbf{y}(\lambda))|_{\lambda=0} = \frac{1}{\|\mathbf{x}\|_1} \left[\underbrace{\|\mathbf{g}\|_2^2 - \|\mathbf{g}\|_\infty \|\mathbf{g}\|_1}_{t_1} + \underbrace{\frac{\mathbf{g}^T \mathbf{x}}{\|\mathbf{x}\|_1}}_{t_2} \underbrace{(n\|\mathbf{g}\|_\infty - \|\mathbf{g}\|_1)}_{t_3} \right] \quad (31)$$

By Hölder's inequality,

$$\|\mathbf{g}\|_2^2 = \sum_{i=1}^n g_i^2 \leq \|\mathbf{g}\|_1 \|\mathbf{g}\|_\infty, \quad (32)$$

with equality only if all the g_i s are equal. If this is not the case, the term t_1 in (31) is negative. Similarly,

$$\|\mathbf{g}\|_1 = \sum_{i=1}^n g_i \leq \|\mathbf{g}\|_\infty \sum_{i=1}^n 1 = n \|\mathbf{g}\|_\infty. \quad (33)$$

Thus t_3 in (31) is nonnegative. Hence the size of the term $t_2 = \mathbf{g}^T \mathbf{x}/\|\mathbf{x}\|_1$ determines the sign of $\frac{d}{d\lambda} c_{\text{MSC}}(\mathbf{y}(\lambda))|_{\lambda=0}$, and hence, whether or not the scaled MSC iterates (30) increase the cost functional for λ sufficiently small.

To make t_2 small we select x_i to be relatively large when g_i is relatively small and we select x_i to be relatively small when g_i is relatively large. One can easily generate examples for which t_2 is sufficiently small that $t_1 + t_2 t_3 < 0$, and hence, the scaled MSC iterates actually *decrease* the cost functional while maintaining the constraints (9)-(10).

5 Numerical Results

5.1 Image Matching Test Problem

This problem is described in subsection 2.1. Results are presented for an implementation of the sequential quadratic programming (SQP) algorithm in the MATLAB Optimization Toolbox [12] as well as MATLAB implementations of projected gradient (PG) iteration described in subsection 4.2 and the MSC iteration described in subsection 4.3. All 3 methods were applied to minimize the MSC cost functional (8) subject to constraints (9)-(10). Computations were performed on a SunBlade 1000 workstation.

In addition, we implemented a “pruning by threshold” variant of MSC iteration, in which we set to zero any coefficient $x_i^{(\ell)}$ which failed to exceed a prescribed threshold τ . We also did not apply any transformation $T_i^{(\ell)}$ and adjoint transformation $U_i^{(\ell)}$ for which $x_i^{(\ell)}$ was set to zero.

For SQP and PG iterations, the initial guess was taken to be

$$x_i^{(\ell)} = 1/n_\ell, \quad i = 1, \dots, n_\ell, \ell = 1, \dots, L. \quad (34)$$

For MSC, we took $x_i^{(\ell)} = 1$ initially. The number of layers is $L = 3$ for this problem, and the number of unknown coefficients $x_i^{(\ell)}$ is $n_s = 78 + 88 + 40 = 206$. We selected the MSC competition parameter value $\kappa = .7$ in order to balance robustness against computational speed. Smaller values of κ yield smaller step sizes for MSC; in general this gives more robustness but higher iteration counts, and hence, longer computation times. We used MATLAB default parameters for SQP and specified the “medium scale” problem size option. Our implementation of PG (with exact solutions to the bent line search subproblems) is parameter free. By conducting a brute-force search, we found the maximal entry of the $78 \times 88 \times 40$ correspondence array. By Theorem 2 this provided us with the global constrained maximizer for the MSC cost functional.

Table 1 summarizes the numerical results. We found that all three methods correctly identified the global constrained maximizer. The computation time required for PG to converge was more than four times that of MSC, while SQP required more than twice as much time as PG to converge for this test problem. The “pruning by threshold” variant of MSC converged more rapidly and took about one third less computational time than the standard MSC iteration.

The purpose of Figs. 3 and 4 is to visualize the performance of PG for this test problem. Fig. 3 shows the evolution with PG iteration count of

	MSC	MSC-thresh	PG	SQP
cost function maximized?	Yes	Yes	Yes	Yes
constraints satisfied?	NA	NA	Yes	Yes
gradient evaluations	28	18	15	60
cost function evaluations	NA	NA	223	119
computation time in seconds	8.85	5.80	38.56	85.49

Table 1: Results for image matching problem. Note that MSC maintains inequality constraints but not equality constraints. Also, MSC requires only gradients of the MSC cost functional. *MSC-thresh* denotes the “pruning by threshold” variant of MSC with threshold parameter $\tau = 0.5$.

the pixel-wise product of the input image I and the transformed template, $U(M)$, where $U = U_{\mathbf{x}^{(1)}}^{(1)} \circ U_{\mathbf{x}^{(2)}}^{(2)} \circ U_{\mathbf{x}^{(3)}}^{(3)}$ is the adjoint of the superposition of transformations in equation (3). Fig. 4 shows the evolution of coefficient layer vectors $\mathbf{x}^{(\ell)}$, $\ell = 1, 2, 3$ with iteration count. We also plot PG iteration count vs MSC correspondence functional, which by virtue of equations (23) and (24), can be computed as the Frobenius inner product of I and $U(M)$. It should be noted that, with some exceptions, analogous plots for MSC would qualitatively resemble the plots for PG shown in these figures. The exceptions are due to the fact that the MSC layer coefficient vectors do not satisfy the equality constraints (9) and these vectors are component-wise monotonically decreasing; see the discussion at the end of section 4.3.

5.2 Rubik’s Cube Results

We present numerical results obtained by applying MSC, PG, and SQP to the Rubik’s cube puzzle described in section 2.2. The difficulty of the problem varies with the number of moves with which to mix the cube. The number of moves equals the number of layers L in the correspondence maximization problem. For each L between 2 and 12, we generated 100 realizations of the Rubik’s cube with L random moves applied, and for each realization we solve the associated correspondence maximization problems using the three methods. The initial guesses for each method are the same as in the previous test case; we took the value of the MSC competition parameter to be $\kappa = .2$.

Results are summarized in Fig. 5. By “success ratio”, we mean the ratio of the number of realizations for which the global maximizer was successfully computed to the total number of realizations. From subplot (a) we see that

for each method the success ratio decreases monotonically as the number of layers (i.e., the number of moves applied to the cubes) increases. The success ratios for MSC and PG are quite similar, but the success ratio for SQP is significantly smaller than that of the other two methods.

Subplot (b) of Fig. 5 shows, for each L , the average amount of computation time required for each method to compute a constrained maximizer. For each method, this amount of time increases monotonically as L increases. For L greater than 4, PG and SQP require about the same amount of time, while MSC requires significantly less time.

6 Summary and Conclusions

Our original motivation for this work was to gain a thorough understanding of David Arathorn’s MSC algorithm, which has produced impressive results for difficult problems in visual cognition, but has lacked theoretical underpinnings. With this goal in mind we were able to precisely define a class problems—multilinear constrained maximization problems associated with correspondence maximization—to which MSC had been successfully applied. We then characterized both global and local solutions to this class of problems. Finally, we applied a pair of standard numerical optimization algorithms—projected gradient (PG) and sequential quadratic programming (SQP)—to solve these problems, and we compared their performance to a particular MSC implementation.

What follows are the main conclusions of this work.

1. The structure of the constrained maximization problems is such that global and local constrained maxima lie on the boundary of the constraint set Ω . In the generic case, these maxima are vertices of the polytope that comprises Ω , and any interior critical point must be a saddle point.
2. For the two test problems considered in this paper, our particular MSC implementations were far superior to PG and SQP in terms of computational efficiency.
3. While MSC makes use of gradient information and superficially resembles a gradient ascent method like PG, there are significant differences between MSC and PG. In particular, PG monotonically increases the cost functional while maintaining the constraints. This guarantees that in the generic case, PG iterates will converge to a local constrained maximizer. However, even if the MSC iterates are rescaled

to maintain the equality constraints (nonnegativity constraints are automatically satisfied by MSC), the cost functional may still decrease with MSC.

Conclusion 3 indicates that we have so far failed to rigorously prove that MSC iterates converge in the generic case to a (scalar multiple of a) vertex which is a local constrained minimizer. Numerical evidence presented in this paper and practical experience of Arathorn and his collaborators strongly suggests that this is the case. Arathorn views MSC as a scheme in which most of the coefficients in the superposition of transformations are iteratively “pruned”, or reduced to zero. The manner in which this is carried out makes use of gradient information, but it does so in a way that resembles no numerical optimization technique that we are aware of. On the more positive side, conclusion 3 indicates that there does exist a method (PG) which, for almost every initial guess, can be proved to yield a local constrained maximizer.

There are a number of open questions that have not been addressed in this paper. Perhaps the most important are the following: How can one characterize the class of correspondence problems (or equivalently, the class or correspondence arrays c ; see eqn. (5)) for which there is a unique local (and hence global) constrained maximizer for c_{MSC} ? Given that algorithms like PG will find a local constrained maximizer in the generic case, this question is tantamount to asking for conditions on c under which PG will converge to the global maximizer. If such conditions can be found, is MSC also then guaranteed to find the global maximizer?

7 Acknowledgements

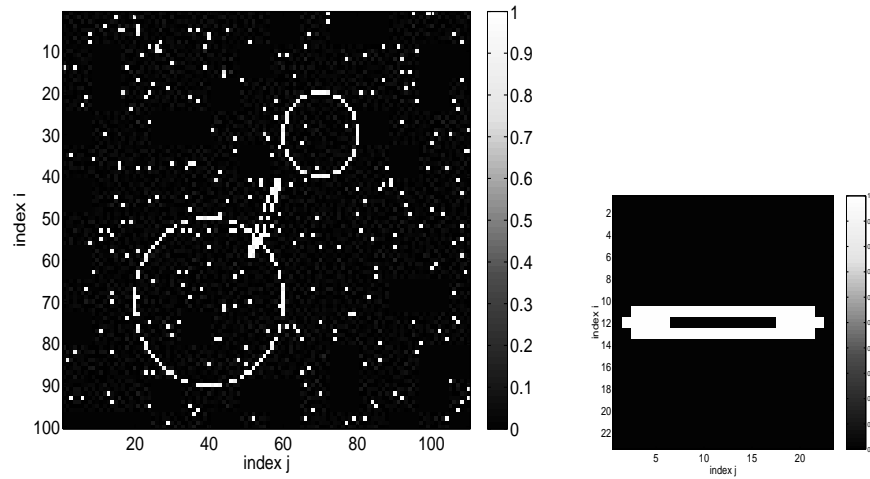
The work of Shaun Harker was partially supported by an NSF-EPSCoR graduate fellowship administered through Montana State University. Tomas Gedeon was partially supported by NSF-BITS grant 0129895, NIH-NCRR grant PR16455, NSF/NIH grant W0467, and NSF-CRCNS grant W0577. Curtis Vogel was partially supported by the Data Driven Modelling and Analysis Group at Los Alamos National Laboratory (US DOE); the Center for Adaptive Optics at UC Santa Cruz through grant AST-9876783; and the NIH Bioengineering Research Partnership at the University of Rochester Center for Visual Science through grant EY014365.

The authors also wish to thank David Arathorn for his assistance and valuable insight.

References

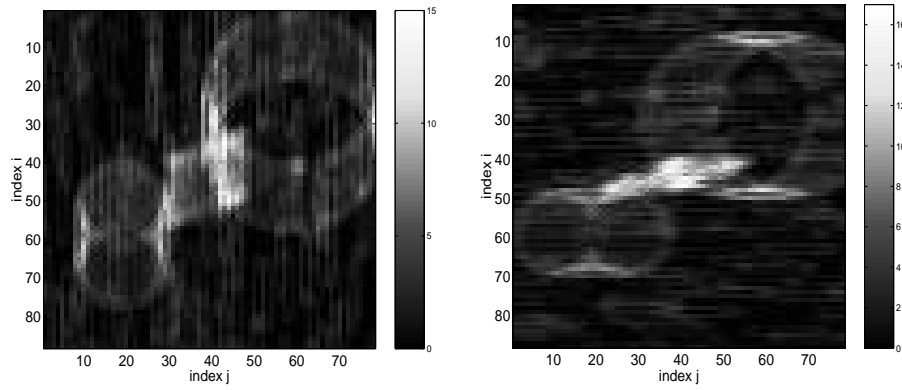
- [1] A. ANGELUCCI, B. LEVITT, E. WALTON, J. M. HUPE, J. BULLIER, AND J. LUND, *Circuits for local and global signal integration in the primary visual cortex*, Journal of Neuroscience, **19** (2002), pp. 8633–8646.
- [2] D. W. ARATHORN, *Recognition under transformation using ordering property of superpositions*, IEE Electronics Letters, **37**, pp. 164–166 (2001).
- [3] D. W. ARATHORN, *Map-Seeking Circuits in Visual Cognition: A Computational Mechanism for Biological and Machine Vision*, Stanford University Press, 2002.
- [4] D. W. ARATHORN, *Computation in higher visual cortices: Map-seeking circuit theory and application to machine vision*, Proceedings of IEEE Applied Imagery Pattern Recognition Workshop (2004), pp. 73–78.
- [5] D. W. ARATHORN, *From wolves hunting elk to Rubik’s cubes: Are the cortices compositional/decompositional engines?*, Proceedings of AAAI Symposium on Compositional Connectionism (2004), pp. 1–5.
- [6] D. W. ARATHORN, *Memory-driven visual attention: An emergent behavior of map-seeking circuits*, in Neurobiology of Attention, Eds. L. Itti, G. Rees, and J. Tsotsos, Academic Press/Elsevier, 2005.
- [7] D. W. ARATHORN, *A cortically plausible inverse problem solving method applied to recognizing static and kinematic 3-D objects*, Advances in Neural Information Processing Systems, Volume 18, MIT Press, 2005
- [8] D. W. ARATHORN, *A cortically plausible inverse problem solving method applied to complex perceptual and planning tasks*, Proceedings SPIE Defense and Security Symposium, 2006.
- [9] P. H. CALAMAI AND J. J. MORÉ, *Projected gradients methods for linearly constrained problems*, Mathematical Programming, **39** (1987), pp. 93-116.
- [10] T. GEDEON AND D. W. ARATHORN, *Convergence in map finding circuits*, Journal of Mathematical Imaging and Vision, submitted.

- [11] D. JOYNER, *Adventures in Group Theory Rubik's Cube, Merlin's Machine, and Other Mathematical Toys*, Johns Hopkins University Press, Baltimore, 2002.
- [12] MATLAB is a registered trademark of The MathWorks, Inc.
- [13] J. J. MORÉ AND G. TORALDO, *On the solution of large quadratic programming problems with bound constraints*, SIAM Journal on Optimization, **1** (1991), pp. 93–113.
- [14] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, Springer Verlag, New York, 1999.
- [15] R. SNIDER AND D. W. ARATHORN, *Terrain discovery and navigation of multi-articulated linear robot using map-seeking circuits*, Proceedings SPIE Defence and Security Symposium, 2006.
- [16] C. R. VOGEL, D. W. ARATHORN, A. ROORDA, AND A. PARKER, *Retinal motion estimation in adaptive optics scanning laser ophthalmoscopy*, Optics Express, **14** (2006), pp. 487-497.



(a)

(b)



(c)

(d)

Figure 1: Illustration of the discrete image matching problem. Subfigure (a) show the 100 pixel \times 110 pixel input image I ; Subfigure (b) shows the 23 pixel \times 23 pixel template image M ; Subfigures (c) and (d) show slices of the correspondence array $c(i, j, k)$ corresponding to $k = 1$ and $k = 21$, respectively.

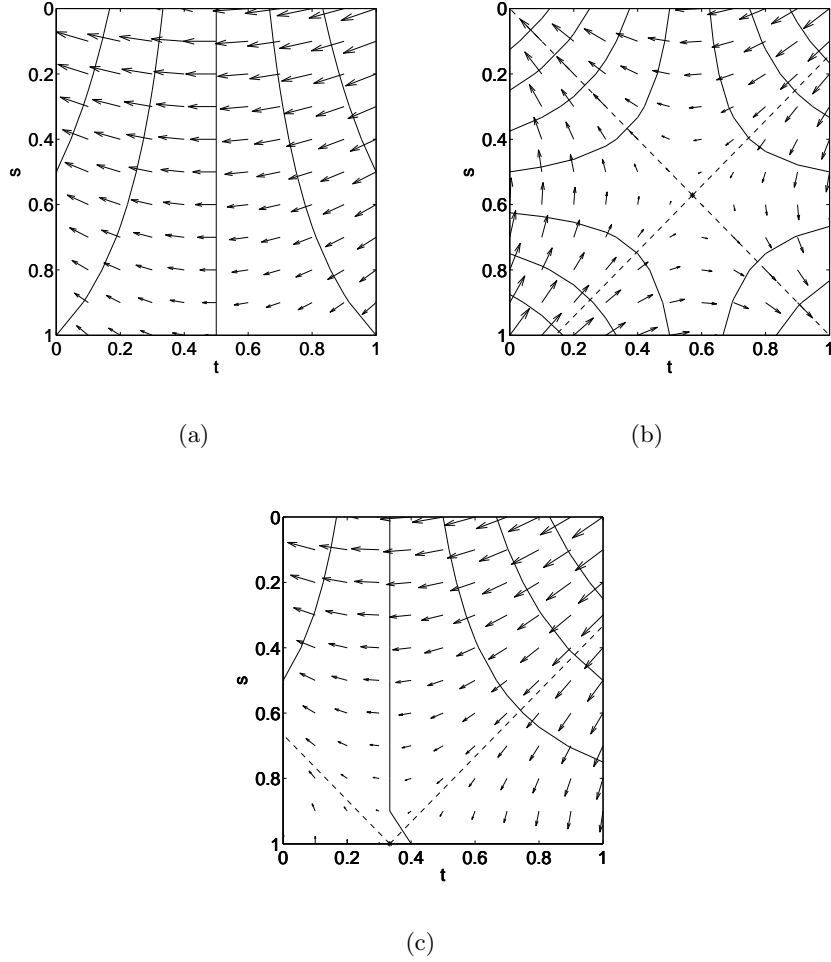
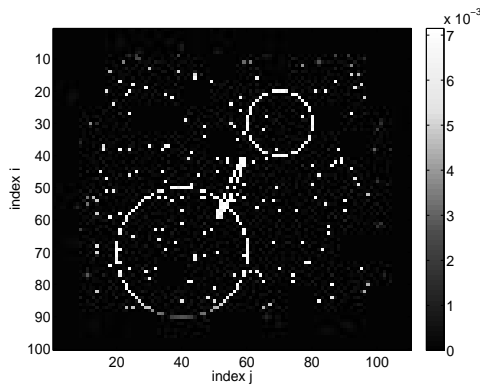
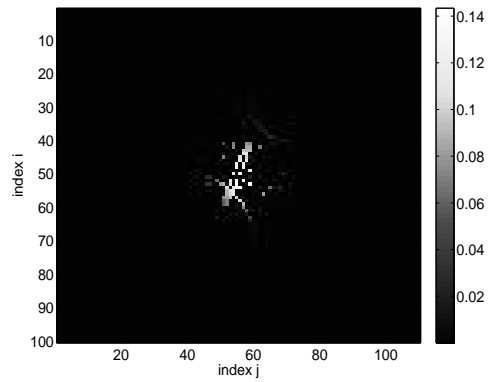


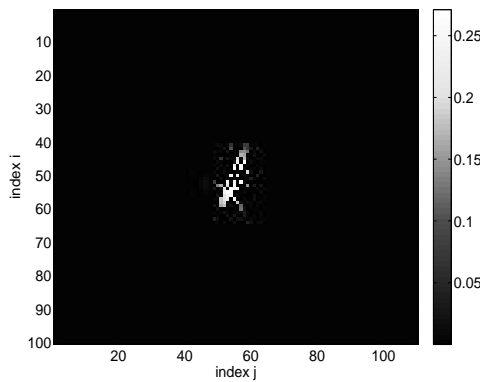
Figure 2: Reduced cost functions $f_k(s, t)$ for the example in Section 3. Subplot (a) shows contours and the gradient field for f_1 ; subplots (b) and (c) show contours and gradients for f_2 and f_3 , respectively. The dashed lines passing through the critical point in subplots (b) and (c) are its stable and unstable manifolds, and the critical points at the intersection of these manifolds are denoted by asterisks (*).



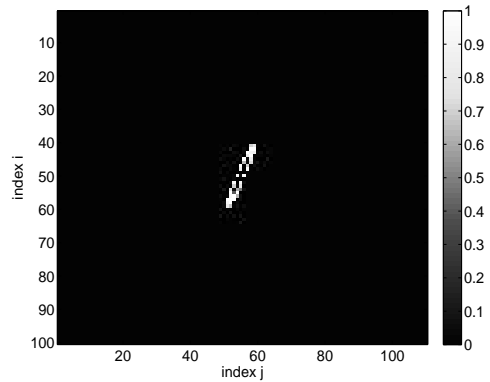
(a) PG iteration 0



(b) PG iteration 1



(c) PG iteration 3



(d) PG iteration 6

Figure 3: Image matches at various PG iterations. Subfigures (a)-(d) show the pixel-wise product of I and $U(M)$ for PG iteration 0 (initialization), iteration 1, iteration 3, and iteration 6 (final PG iteration).

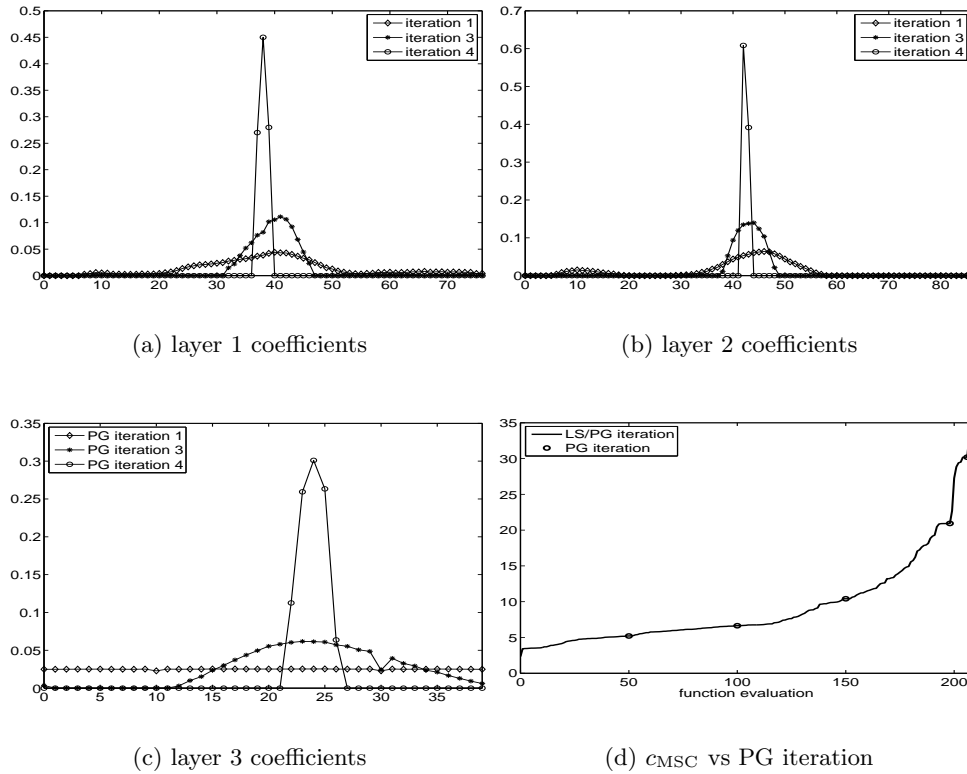


Figure 4: PG performance vs iteration count for the image matching problem. The plots in subfigures (a)-(c) show how the layer coefficient vectors $\mathbf{x}^{(\ell)}$, $\ell = 1, 2, 3$ change with iteration count. Subfigure (d) shows the MSC correspondence functional vs cumulative function evaluations. The circles in subfigure (d) indicate PG iterations, while the solid line between circles indicates line search iterations. Convergence was attained at PG iteration 6.

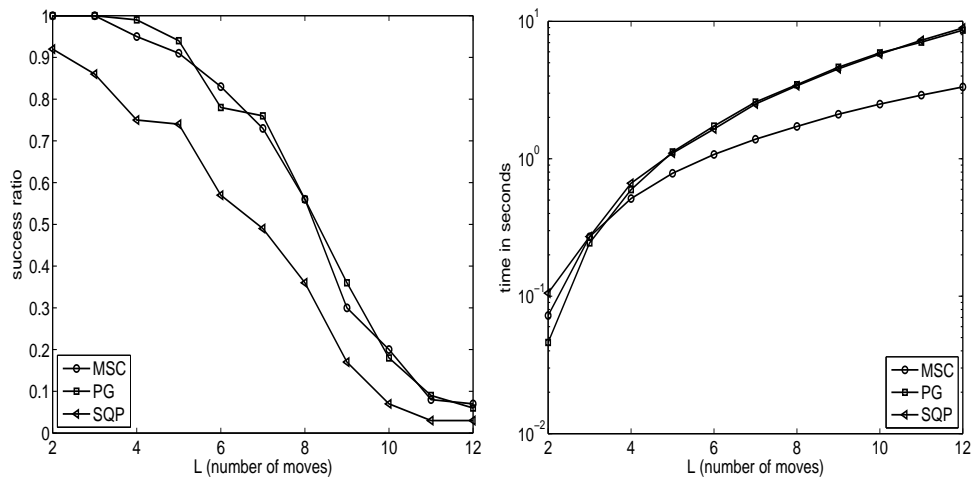


Figure 5: Performance of methods for solving Rubik's cubes. The plots in subfigure (a) show success ratio as a function of the number of moves (equivalently, number of layers) L . Plots in subfigure (b) show average CPU time as a function of L .