Fully constrained least-squares spectral unmixing by simplex projection.

Rob Heylen*, Dževdet Burazerović†, Paul Scheunders‡
IBBT-Visielab, University of Antwerp
Universiteitsplein 1
2610 Wilrijk, Belgium

* Email: rob.heylen@ua.ac.be. Tel: +32(0)3 265 2478. Fax: +32(0)3 265 2245
† Email: dzevdet.burazerovic@ua.ac.be. Tel: +32(0)3 265 2478. Fax: +32(0)3 265 2245
‡ Email: paul.scheunders@ua.ac.be. Tel: +32(0)3 265 2473. Fax: +32(0)3 265 2245

Abstract—We present a new algorithm for linear spectral mixture analysis, capable of supervised unmixing of hyperspectral data while respecting the constraints on the abundance coefficients. This simplex-projection unmixing algorithm is based upon the equivalence of the fully constrained least-squares problem and the problem of projecting a point onto a simplex. We introduce several geometrical properties of high-dimensional simplices, and combine them to yield a recursive algorithm for solving the simplex projection problem. A concrete implementation of the algorithm for large data sets is provided, and the algorithm is benchmarked against well-known fully-constrained least-squares unmixing techniques, on both artificial data sets and real hyperspectral data collected over the Cuprite mining region. Unlike previous algorithms for fully-constrained least-squares unmixing, the presented algorithm possesses no optimization steps and is completely analytical, severely reducing the required processing power.

EDICS Category: SAM-SENS

I. INTRODUCTION

Spectral unmixing of hyperspectral imagery is an important subject in the field of remote sensing and image processing, and has received significant attention in the past decades. Spectral unmixing concerns the decomposition of a pixel spectrum into its constituent spectra, the motivation being that the surface area of a single pixel contains a low number of “pure” materials, each with a given fraction [1]. Usually both the spectra of the pure materials, called endmembers, as well as their abundances in each pixel, are considered unknown. Most algorithms for spectral unmixing therefore need to perform at least two tasks, one concerned with finding the unknown endmember spectra, and a second one of determining the corresponding abundances of each endmember in each pixel. The former task is labeled as endmember extraction, while the latter is labeled as unmixing, the inversion problem, or spectral mixture analysis. By combining an endmember extraction algorithm (EEA) with a spectral mixture analysis algorithm, one yields a holistic approach to the spectral unmixing problem. Some approaches (e.g. [2], [3], [4]) integrate both tasks in a single algorithm (unsupervised unmixing), while most algorithms first look for the endmembers, after which these endmembers are used in a supervised unmixing algorithm for determining the abundances.

Most spectral unmixing algorithms start from the linear mixing assumption [1], or the assumption that an observed spectrum is a linear combination of a limited number of endmember spectra, where the linear coefficient of each endmember is its abundance. These abundances have to obey two constraints, derived from their physical interpretation: They have to be positive, since no endmember material can have a negative presence in a pixel’s field of view, and they have to sum to one, indicating that the spectrum of every pixel can be completely decomposed into the contributions of the endmembers, and nothing else. This linear model, together with the constraints on the abundances, has proven to be very successful in practice, and many algorithms for spectral unmixing exist that start from these assumptions. See [5] for a detailed taxonomy of linear spectral unmixing algorithms, and [6], [7] for recent surveys of unmixing algorithms.

Most of the linear unmixing algorithms exploit the fact that due to the linear mixing equation, the mixed pixels have to lie inside the convex hull of the endmembers [8]. This convex hull forms a simplex in the spectral space, with the endmembers as spanning vertices. Some algorithms then try to look for the largest volume embedded simplex (NfindR [9], simplex growing algorithm [10]), or try to identify extreme points in the data cloud (e.g. pixel purity index [8], orthogonal subspace projection [11], convex cone analysis [12], vertex component analysis [13]). Under the assumption that the endmembers are present in the data as pure pixels, these strategies are able to locate these pure pixels, hence identifying the endmembers. Many other algorithms exist, and many of them do not require the endmembers to be present in the data [6].

Most of these algorithms are mainly concerned with endmember extraction. Once the endmembers have been found, the estimation of the abundances requires the inversion of the linear mixing equation, since one only has the data points and the endmembers available, and one would like to estimate the linear coefficients (the abundances). Due to the presence of noise in the data, and possibly small errors in the estimated endmember spectra, this problem has no straightforward analytical solution. Furthermore, one would like to obey the restrictions imposed on the abundance coefficients (positivity and sum-to-one), further complicating the matter.

Over time, many algorithms have been proposed for in-
verting the linear mixing equation, effectively unmixing the data points into their abundances. Originally, one used simple least-squares linear regression for solving the over-determined system of linear equations [14]. Unfortunately, these solutions did not obey any of the constraints on the abundances. However, it proved easy to also incorporate the sum-to-one constraint, leaving the problem to be solved as a non-negatively constrained least-squares problem [2]. Further improvements could then be reached via a zeroing procedure, where negative abundances were replaced with zero, and normalization of the resulting positive abundance vectors. Orthogonal subspace projection lead to unmixing algorithms via subspace projection methods [15]. In [16] a weighted least-squares method was proposed, and here it was also observed that solving the unmixing problem with all constraints obeyed would require quadratic programming. None of these techniques however is able to find the optimal solution.

Then, in [2] a technique was proposed that first turns the unmixing problem into a non-negatively constrained least-squares problem by introduction of a Lagrangian, after which this problem is solved using the techniques developed by Lawson and Hanson (the non-negatively constrained least-squares (NNLS) problem [17]). This was the first algorithm that finds the least-squares solution that obeys both the positivity and sum-to-one constraint, yielding the fully-constrained least-squares unmixing (FCLSU) algorithm. Soon after, other algorithms followed that also allowed to find this solution, however through some other approach. In [18] for instance, one first notes the equivalence between solving the FCLSU problem and the simplex projection problem, after which a quadratic programming algorithm is used for solving the latter problem in the context of spectral unmixing. Quadratic programming has also been successfully used in other unmixing algorithms, e.g. [19], [20]. In [21] one transforms the formulation of the least-squares problem into a least-distance problem, which subsequently can be solved by the Lawson-Hanson NNLS algorithm. In [2], [3] a combined algorithm is proposed, capable of simultaneous fully-constrained least-squares unmixing and endmember extraction by extremizing reconstruction errors. More recently, a hierarchical Bayesian model has been proposed [4], [22] for simultaneous endmember extraction and solving the inversion problem, again allowing one to find the optimal solution to the FCLSU problem. Another approach treats the problem as a blind source separation problem, and uses Bayesian techniques [23], [24], [25] or non-negative matrix factorization [26], [27] to find the endmembers and abundances. In [28] it has been shown that also fuzzy set theory and fuzzy logic can be successfully used for solving the inversion problem.

Most of the approaches that provide the exact FCLSU solution use an optimization or maximization procedure, gradient or steepest-descent methods, or require a random sampler from some distribution. E.g. The Lawson-Hanson NNLS algorithm and quadratic programming techniques are both optimization techniques, requiring appropriate starting vectors and step sizes. Bayesian techniques depend on random samplers. Although the results obtained with these algorithms can be made arbitrarily accurate by modifying the algorithmic parameters, they do not provide an analytically correct solution based upon some mathematical principles, but resort to optimization strategies.

In this paper, we present a new supervised unmixing algorithm, the simplex-projection unmixing (SPU) algorithm, based upon the equivalence between the FCLSU problem and the simplex projection problem [18], [29]. We present an algorithm that is capable of finding the projection of a point onto a generic simplex, without resorting to quadratic programming or NNLS techniques. To our knowledge, the algorithms for simplex projection present in the literature either employ quadratic programming [30], [18], or project only on canonical [31] simplices. The solution obtained via this simplex projection algorithm almost always minimizes the least-squares error, while strictly obeying the constraints on the abundances. The algorithm is a recursive algorithm that reduces the dimensionality of the problem by one with each recursion, until an appropriate abundance vector is found. At each step of the recursion, we identify an endmember with zero abundance, and perform an orthogonal projection onto a hyperplane of one dimension less than the previous recursion. Because it does not resort to some optimization or maximization procedure, it has a very high computational performance, and in real spectral unmixing situations, the produced results are comparable to the exact FCLSU solution.

The outline of this paper is as follows: In section II we define the problem of spectral unmixing, and explain the equivalence between the fully constrained least-squares problem and the simplex projection problem. In section III we present the reasoning that lead to the algorithm, along with the various mathematical concepts needed. We then formally present the algorithm, followed by an explanation of every step, and a concrete implementation for large data sets. In section IV we present the results obtained by the algorithm, and compare the computational performance of this algorithm with the well-known FCLSU algorithm presented in [2], both on a synthetic data set, and on the AVIRIS Cuprite data set. Section V contains the conclusions and future work.

II. FCLSU AND SIMPLEX PROJECTION.

Consider a pixel \( \mathbf{x} \) in a hyperspectral image with \( d \) spectral bands: \( \mathbf{x} = (x_1, x_2, \ldots, x_d) \). In the linear mixing model, one assumes that every pixel consists of a linear combination of \( p \) endmembers \( \{e_1, \ldots, e_p\} \), and added noise. We assume that \( p < d \):

\[
\mathbf{x} = \sum_{i=1}^{p} a_i e_i + \epsilon
\]  

(1)

The coefficients \( a_i \) represent the abundances of each endmember in pixel \( \mathbf{x} \). The two constraints on these abundance coefficients, positivity and sum-to-one, then become:

\[
\forall i : a_i \geq 0 \quad \sum_{i=1}^{p} a_i = 1
\]  

(2)  

(3)
The spectral unmixing problem tries to solve the following problem: If the endmembers \( \{e_i\}_i \) are known, find the abundances \( \{a_i\}_i \) so that equation (1) holds. When the noise equals zero \( (\epsilon = 0) \), this problem becomes trivial, since then we have an over-determined system of \( d \) linear equations in \( p \) variables for every pixel \( x \), with \( d > p \). However, when the noise is non-zero, one usually cannot satisfy equation (1) for any set of abundances \( \{a_i\}_i \).

Since the noise \( \epsilon \) is usually assumed to be independent and identically distributed Gaussian noise with mean zero and a covariance matrix proportional to the unity matrix, the simplest method for approximating the solution of an over-determined linear system that cannot be solved exactly is by simple least-squares linear regression, yielding least-squares unmixing (LSU). This method minimizes the sum of squared differences between the observed spectral values \( x_i \), and the reconstructed values predicted by the linear approximation, without any constraints on the abundance coefficients. With \( \hat{a}_i \), the estimate for \( a_i \), we find:

\[
\{\hat{a}_i\}_{LSU} = \text{argmin}_{\{\hat{a}_i\}} \|x' (\hat{a}_1, \ldots, \hat{a}_p) - x\|_2
\]

where we introduced the reconstructed data point

\[
x' = \sum_{i=1}^{p} \hat{a}_i e_i
\]

Stated otherwise, the LSU solution provides us with the unique linear combination of endmembers with smallest \( \mathcal{L}_2 \)-distance to the data point \( x \). If we define the reconstruction error as

\[
\Delta = \|x' - x\|_2
\]

one can also state that the LSU solution minimizes the reconstruction error. The solution to the LSU problem can be easily found via the pseudo-inverse of the endmember matrix. With \( E = [e_1, e_2, \ldots, e_p] \):

\[
\hat{a} = (E^T E)^{-1} E^T x
\]

With this, we can write the reconstructed data point \( x' \) as

\[
x' = E\hat{a} = E(E^T E)^{-1} E^T x
\]

Hence the solution is provided by a projection operator \( P_{LSU} = E(E^T E)^{-1} E^T \), which orthogonally projects onto a subspace spanned by the columns of \( E \), which are the endmembers: \( x' = P_{LSU}(x) \).

The main problem with the LSU solution is that the constraints on the abundances (2) and (3) are not necessarily obeyed. Adding some constraints on the allowed values for the abundances corresponds to looking for a reconstructed point in only a subspace of the linear space spanned by the endmembers. If both constraints have to be obeyed exactly, this subspace will be the convex hull of the \( p \) endmembers, which is the simplex spanned by these endmembers. Indeed, the positivity and sum-to-one constraint effectively turn the abundances into normalized barycentric coordinates with respect to the endmembers, yielding a simplex lying on a \( (p-1) \)-dimensional plane in the \( d \)-dimensional spectral space.

The reconstructed point \( x' \) corresponding to the FCLSU solution now equals the point inside this endmember simplex that minimizes the reconstruction error \( \Delta \), or \( \mathcal{L}_2 \)-distance, with the actual data point \( x \). If we define the simplex as

\[
y \in S \iff \exists a_1, \ldots, a_p \in [0, 1] : \left\{ y = \sum_{i=1}^{p} a_i e_i \sum_{i=1}^{p} a_i = 1 \right\}
\]

then we can introduce an operator \( P \), projecting a point \( x \) onto the closest point inside the simplex \( S \), as

\[
x' \in S : x' = P(x) \iff \forall y \in S : \|x - y\|_2 \geq \|x - x'\|_2
\]

The operator \( P \) is idempotent \((\forall x : P(P(x)) = P(x)\), making it a projection operator. Due to its construction, the projected point \( x' \) lies in the simplex \( S \), and by solving \( x' = \sum_i \hat{a}_i e_i \), which now is an exactly solvable over-determined linear system, we can find the corresponding abundance coefficients \( \hat{a} \). Hence, there exists a clear equivalence between the FCLSU problem and projecting a point onto a simplex: The reconstructed point from the FCLSU solution equals the projected point onto the simplex. Similarly, the abundance coefficients of the projected point, which are normalized barycentric coordinates in the endmember coordinate system, are the same as the FCLSU solution. In the next section, we present an algorithm for performing the projection operation \( P \).

III. THE SIMPLEX-PROJECTION UNMIXING ALGORITHM

A. Prerequisites

The presented method for projecting a point \( x \) onto a simplex \( S \), and finding the corresponding abundance coefficients, rests on several observations. We list these observations here, together with a proof, and present the reasoning leading to the algorithm.

**Lemma 3.1.** Orthogonal projection onto the simplex plane (the hyperplane through the endmembers) leaves the simplex projection invariant.

**Proof:** Consider a simplex \( S \) in \( \mathbb{R}^d \) spanned by \( p \) endmembers \( \{e_1, \ldots, e_p\} \). Let \( y \) be the orthogonal projection of a point \( x \) onto the plane through the endmember points \( \{e_1, \ldots, e_p\} \), and let \( s \) be any point inside the simplex \( S \). We then have that \( \|x - s\|_2^2 = \|x - y\|_2^2 + \|y - s\|_2^2 \) because the vectors \( x - y \) and \( y - s \) are orthogonal. Since \( \|x - y\|_2 \) is constant, the \( s \) that minimizes \( \|x - s\|_2 \) will also minimize \( \|y - s\|_2 \). Hence \( P(x) = P(y) \), and orthogonal projection onto the simplex plane is an invariant operation for the simplex projection operator \( P \).

A consequence of this lemma is that we can always restrict ourselves to points lying in the \( (p-1) \)-dimensional simplex plane, instead of in the full \( d \)-dimensional space, by replacing \( x \) by its orthogonal projection onto this simplex plane. The coordinates of the points however are still \( d \)-dimensional.
The next lemma states that the projection of a point outside a simplex always has at least one abundance coefficient zero:

**Lemma 3.2:** Consider a simplex \( S \) in \( \mathbb{R}^{p-1} \) spanned by \( p \) endmembers \( \{e_1, \ldots, e_p\} \), and a point \( x \notin S \). The projection \( x' = P(x) \) lies on the boundary of the simplex \( S \), and hence has at least one abundance coefficient \( a_i = 0 \).

**Proof:** Suppose \( x' \) has only non-zero abundance coefficients. Then \( x' \) will lie in the interior of \( S \). The line-segment connecting \( x' \) with \( x \) will intersect the boundary of the simplex \( S \) in a point \( y \) between \( x' \) and \( x \), and \( \|x - y\|_2 < \|x - x'\|_2 \), proving that \( x' \) cannot be the projection of \( x \) onto \( S \).

The aim next is to be able to identify the abundance coefficient that has to be zero. For this, we first introduce the incenter of a simplex:

**Definition** The incenter \( c \) of a simplex \( S \) spanned by \( \{e_1, \ldots, e_p\} \) is defined as the intersection of all \( (p - 2) \)-dimensional planes that bisect the dihedral angles between the faces of the simplex. This is also the center of the largest possible hyper-sphere one can inscribe in the simplex. See Fig. 1.

The coordinates of the incenter \( c \) can be found via the \( (p - 2) \)-dimensional volumes of the \( p \) faces of the simplex. Let \( V_i \) be the volume of the sub-simplex spanned by \( \{e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_p\} \). Then we have [32]

\[
a_i^c = \frac{V_i}{\sum_{i=1}^p V_i} \quad \text{for} \quad i = 1, \ldots, p \tag{11}
\]

yielding a vector \( a^c \) containing the barycentric coordinates of the incenter \( c \). The Euclidean coordinates of the incenter are then given by

\[
c = E a^c \quad \text{for} \quad c \in S \tag{12}
\]

Consider a point \( x \) lying in the simplex plane. The ray starting from the incenter \( c \) through \( x \) will intersect a certain face of the simplex. We define the set of all points intersecting a given face as the bissective cone of that face:

**Definition** The bissective cone \( Z_i \) of a simplex \( S \) in \( \mathbb{R}^{p-1} \) spanned by \( \{e_1, \ldots, e_p\} \) is the set of points defined by:

\[
x \in Z_i \iff \exists b_1, \ldots, b_p \geq 0 : \left\{ \begin{array}{l}
x = c + \sum_{j=1}^p b_j(e_j - c) \\
b_i = 0
\end{array} \right. \tag{13}
\]

with \( c \) the incenter of \( S \).

In Fig. 1 the concept of bissective cones is illustrated for a simplex in the 2-dimensional plane. It is clear that these cones form a partitioning of the \( (p - 1) \)-dimensional simplex plane:

\[
\bigcup_i Z_i = \mathbb{R}^{p-1} \tag{14}
\]

These bissective cones allow one to estimate the abundance coefficient that has to be zero. We first present the following theorem, of which we will use an extension to generic simplices later:

**Theorem 3.3:** Consider the canonical simplex \( S_c \) in \( \mathbb{R}^p : v \in S_c \iff (\sum v_i = 1 \text{ and } \forall i : v_i \geq 0) \), and a point \( x \) in the simplex plane, with \( x \notin S \). The projection of \( x \) on the simplex \( S_c \) is \( x' = P(x) \), with barycentric coordinates \( a_1, \ldots, a_p \) (abundances). We then have

\[
x \in Z_i \Rightarrow a_i = 0; \tag{15}
\]

**Proof:** First of all, remark that barycentric coordinates equal Euclidean coordinates in the canonical simplex. To prove the theorem, we use the following lemma proven in [33], and also used in [34]:

**Lemma 3.4:** Consider the canonical simplex \( S_c \) in \( \mathbb{R}^p : v \in S_c \iff (\sum v_i = 1 \text{ and } \forall i : v_i \geq 0) \). Let \( y \in S_c \) be the vector that minimizes \( \|y - x\|_2 \) for a given \( x : y = P(x) \). Let \( i \) and \( j \) be two indices such that \( x_i > x_j \). We then have that \( y_i = 0 \Rightarrow y_j = 0 \).
In the context of simplex projection, we know (see lemma 3.1) that any point \( x \) can be replaced with its orthogonal projection on the endmember plane without affecting the projection onto the simplex. Hence we can restrict ourselves to points \( x \) lying in this simplex plane. We also know that if a point \( x \) lies in the simplex plane, but outside the simplex \( S_c \), there has to be at least one barycentric coordinate (or abundance coefficient) zero (lemma 3.2). A corollary of above lemma is then that the smallest \( x_i \) certainly corresponds to a zero barycentric coordinate:

\[
x \notin S_c, \ x_i = \min(x_1, \ldots, x_p) \Rightarrow y_i = 0 \tag{16}
\]

See Fig. 2 and 3 for an illustration of this. Now, the bissective cones \( Z_i \) correspond exactly to those subsets where a given coordinate \( x_i \) becomes minimal. Hence, a consequence of this corollary is that, at least for the canonical simplex, every point \( x \notin S_c, x \in Z_i \), will have its corresponding barycentric coordinate \( \hat{a}_i = 0 \).

In the proposed algorithm, we extend the previous theorem to generic simplices, where we replace the origin with the incenter. Hence, consider a simplex \( S \) in \( \mathbb{R}^{p-1} \) spanned by \( p \) endmembers \( \{e_1, \ldots, e_p\} \), and a point \( x \notin S \). The projection of \( x \) is \( x' = P(x) \), with barycentric coordinates \( \hat{a}_1, \ldots, \hat{a}_p \) (abundances). We then assume that

\[
x \in Z_i \Rightarrow \hat{a}_i = 0; \tag{17}
\]

This assumption is not true in general: Although for \( p = 3 \) the assumption seems to hold, one can generate counter-examples for any \( p > 3 \). However, these counter-examples usually involve simplices that are not well-shaped: Their volume is very low compared to the smallest inter-vertex distance, and some dihedral angles are very obtuse. Such situations will not occur often in real hyperspectral data. Furthermore, for higher values of \( p \) it becomes very hard to find counter-examples, and this assumption can be used in practice.

Since this assumption now allows us to estimate which abundance has to be zero, we can construct a recursive algorithm: Suppose \( x \) lies outside a simplex \( S \), but in the simplex plane. Since a certain known endmember \( e_j \) will not contribute to the projection \( x' \) of \( x \), we know that \( x' \) has to lie on the sub-simplex \( S_i \) spanned by all endmembers except \( e_j \). Hence projecting onto this sub-simplex \( S_i \) instead of the full simplex \( S \) will yield the same abundance coefficients with respect to the endmembers spanning \( S_i \). Orthogonal projection onto the plane of \( S_i \) reduces the implicit dimensionality of the problem by one, and one can recursively continue until one either finds a projected point lying inside a simplex, or one finishes with a trivial one-dimensional problem. In the next section, we formally present this algorithm.

B. The algorithm

The observations in previous section lead to the following algorithm for projecting a point onto a simplex, and finding the corresponding abundance coefficients. Suppose one has a point \( x \in \mathbb{R}^d \) one wants to project onto a simplex \( S_I \) spanned by the \( p \) endmembers in the set \( I = \{e_1, \ldots, e_p\} \).

1) Orthogonally project the point \( x \) onto the simplex plane, determined by the endmembers in \( I \), yielding \( y \).
2) If \( y \) lies in the simplex \( S_I \), go to step 4. Else, go to the next step.
3) Determine the cone \( Z \), that contains \( y \). Remove \( e_i \) from the set \( I \), yielding \( I' = I / \{e_i\} \), and a new sub-simplex \( S_I' \) of one dimension less. Put the abundance coefficient \( \hat{a}_i = 0 \). Set \( I = I' \), \( x = y \) and go to step 1.
4) Determine the abundance coefficients by writing \( y \) as a linear combination of the remaining endmembers in \( I \). This is now an over-determined and exactly solvable system of linear equations, respecting the constraints on the abundances. All abundances that correspond to an endmember not in \( I \) are zero.

The first step of the algorithm is to find the orthogonal projection \( y \) of a point \( x \) onto the simplex plane of \( S_I \). This projection can be obtained by adapting equation (8), describing projection onto the subspace spanned by the endmember vectors:

\[
\hat{E} = [e_2 - e_1, e_3 - e_1, \ldots, e_p - e_1] \tag{18}
\]

\[
y = \hat{E}(\hat{E}^T \hat{E})^{-1} \hat{E}^T (x - e_1) + e_1 \tag{19}
\]

Above equations are also valid for any permutation of the \( p \) endmembers. This projection corresponds to a translation of \(-e_1\), which puts the endmember \( e_1 \) at the origin, then projecting the translated point \( x - e_1 \) onto the space spanned by the \((p-1)\) other translated endmembers \( e_2 - e_1, \ldots, e_p - e_1 \) (this is the plane through these translated endmember and the origin), and performing the inverse translation from the origin to \( e_1 \), yielding the projected point \( y \).

This equation (19) also allows us to determine whether a point lies inside a simplex or not, which is needed in the second step of the algorithm. Furthermore, it allows us to find the abundances of a point inside a simplex as well, as needed in the fourth step of the algorithm. First, we define the partial abundance coefficients \( v \) as

\[
v = (\hat{E}^T \hat{E})^{-1} \hat{E}^T (x - e_1) \tag{20}
\]

If all \( v_i, i = 1, \ldots, p - 1 \) are positive, and \( \sum_i v_i \leq 1 \), the orthogonal projection \( y \) of the point \( x \) lies in the simplex, and the corresponding abundance coefficients \( \hat{a} \) are given by the concatenation of \( \hat{a}_i = 1 - \sum_i v_i \) and the vector \( v \):

\[
\hat{a} = [1 - \sum_i v_i, v] \tag{21}
\]

In the third step we need to determine the cone \( Z_i \) that contains \( x \). To determine this cone, we must first calculate the location of the incenter \( c \), which can be done via equations (11) and (12). We must still calculate the volumes of the \( p \) sub-simplices however, where the vertices are given as \( d \)-dimensional coordinate vectors. A straightforward method for calculating these volumes that is independent of the dimensionality \( d \) of the data is by using the Cayley-Menger determinant. Let \( d_{ij} \) be the Euclidean distance between points \( e_i \) and \( e_j \). The volume \( V \) of a simplex spanned by endmembers
\(e_1, \ldots, e_p\) can then be found via
\[
(-1)^p 2^{p-1}((p-1)!)^2 v^2 = \det(C_{1,2,\ldots,p}) \tag{22}
\]
\[C_{1,2,\ldots,p} = \begin{pmatrix}
D_{1,2,\ldots,p} & 1 \\
1 & 0
\end{pmatrix} \tag{23}
\]
with \(D_{1,2,\ldots,p} = [d_{ij}^2]_{i,j=1,2,\ldots,p}\). This equation is valid for any number of endmembers \(p\), and has no explicit dependence on the dimensionality of the data \(d\).

Once we have determined the incenter \(c\), we can find the cone \(Z_i\) that contains \(x\). If we define
\[
x^c \equiv x - c \tag{24}
\]
\[e_i^c \equiv e_i - c \tag{25}
\]
\[E_i^c = \begin{bmatrix} e_1^c, \ldots, e_{i-1}^c, e_{i+1}^c, \ldots, e_p^c \end{bmatrix} \tag{26}
\]
then the linear system of equations
\[
x^c = E_i^c b_i \tag{27}
\]
will have a solution for every index \(i \in \{1, \ldots, p\}\). Because of the definition of bisejective cone (13), we find:
\[
x \in Z_i \iff \forall j : b_j^i \geq 0 \tag{28}
\]
These observations result in the following algorithm for determining the cone \(Z_i\) that contains \(x\):

1) Calculate the incenter \(c\) and the entities (24)-(26).
2) Solve equation (27) for every \(i\), yielding a set of vectors \(\{b_i\}_{i=1,\ldots,p}\).
3) The set \(b_i\) with only positive entries identifies the cone that contains \(x\).

To solve equation (27) for \(b_i\), it suffices to consider only a \((p-1) \times (p-1)\) linearly independent sub-matrix of \(E_i^c\) instead of the full \(d \times (p-1)\) matrix, and use matrix inversion.

In the case \(p = 2\), the algorithm could be simplified by observing that the incenter is located in the middle between the two endmembers making up the simplex.

**C. Implementation for large data sets**

In previous section we have described how the algorithm functions for projecting a single point onto a simplex. In practice however, one usually needs to perform this operation for a large number of points. Since many steps of the algorithm contain operations involving only the endmembers, a large performance boost can be gained by processing all data points at the same time. We provide a concrete implementation here for the simplex projection algorithm executed on large data sets. Hence, consider a data set of \(N\) points \(x_n \in \mathbb{R}^d, n = 1, \ldots, N\), and \(p\) endmembers \(\{e_1, \ldots, e_p\}\). We want to project all points \(x_n\) onto the simplex spanned by these \(p\) endmembers, yielding the projected points \(x'_n\) and their corresponding abundance vectors \(\alpha_n = (a_{n1}, \ldots, a_{np})\). The projected or reconstructed points \(x'_n\) are found via \(x'_n = E \tilde{\alpha}_n\).

As a simple illustration, we have projected a number of points on the unit circle onto a random simplex with this algorithm in Fig. 4.

---

**Algorithm 1: simplex_project**

**input:** \(x, E\)  
\(x\): \(d \times N\) matrix, the \(N\) points to unmix  
\(E\): \(d \times p\) matrix, the \(p\) endmembers  
**output:** \(a\)  
\(a\): \(p \times N\) vector, the abundances

**begin**

if \(p = 1\) then
    \(a = 1\);
return;

\(I = \{\}\);  
for \(\forall n \in [1, \ldots, N]\) do
    Calculate partial abundance \(v^n\) with (20);
    if \(\forall j : v^n_j \geq 0\) and \(\sum_j v^n_j \leq 1\) then
        \(a_n = [1 - \sum_j v^n_j, v^n]\);
    else
        Project \(x_n\) onto simplex plane with (19);
        Add \(n\) to \(I\);

Calculate endmember distance matrix \(D\);  
for \(\forall i \in [1, \ldots, p]\) do
    Calculate volume \(V_i\) of sub-simplex \(E_i = [e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_p]\) with (22);
Calculate the incenter \(c\) with (11) and (12);
for \(\forall i \in [1, \ldots, p]\) do
    \(I_i = \{\}\);
    for \(\forall j \in I_i\) do
        Solve \(x_j = E_i^c b_i^j\) for \(b_i^j\);
        if \(\forall k : b_k^j \geq 0\) then
            Add \(j\) to \(I_i\);
    if \(I_i \neq \{\}\) then
        \(a^r = \text{simplex_project}(x, E_i)\);
        \(a([1, \ldots, i-1, i+1, \ldots, p], I_i) = a^r\);
        \(a(i, I_i) = 0\);

Fig. 4. 60 points on the unit circle (circles) are projected onto a randomly chosen simplex (triangle). The points are connected to their projections (in stars) with dotted lines for clarity.
IV. Performance of the Algorithm

A. Complexity

The algorithm always finishes in a finite number of recursions, where every recursion contains a finite number of steps. The number of recursions needed however is highly dependent on the data. In the worst-case scenario, the number of recursions needed equals \(\min(N(p - 1), 2^p - 2)\), where \(N\) is the number of data points, and \(2^p - 2\) is the number of sub-simplices in a simplex spanned by \(p\) endmembers. In the spectral unmixing problem however, most pixels lie inside or close to the endmember simplex, significantly reducing the number of iterations required.

Within a single recursion of the algorithm, the main computational cost of the different steps of the algorithm are:

- Simplex plane projection requires a matrix inversion and multiplication of a \(p \times p\) matrix, and a matrix multiplication of a \(p \times d\) with a \(d \times N\) matrix.
- Determining the location of the incenter involves the calculation of \(p\) determinants of symmetric and positive definite matrices of size \((p - 1) \times (p - 1)\).
- Determining the vectors \(b_i^t\) for a given cone \(Z_i\) involves a single inversion of a \((p-1) \times (p-1)\) matrix, and a matrix multiplication of a \((p-1) \times (p-1)\) and a \((p-1) \times N\) matrix. This has to be executed \(p\) times.
- All other steps are steps involving index management, or have a negligible computational complexity.

We did not list the step where the inter-vertex distances are calculated, since this can be executed once at the beginning of the algorithm, and can be passed between the recursive calls, since this matrix does not change. The total complexity of the algorithm depends strongly on the implementation details and optimizations used, and is hard to determine explicitly. We rather quantify the dependence of the runtime on the dimensionality of the problem in a practical way, by explicitly running the algorithm for several (simulated and real) data sets, and determining the runtime and number of recursive calls needed for completion.

B. Execution on an artificial data set

As a first experiment, we test the performance of the algorithm on an artificial data set, representing simulated hyperspectral data that is linearly mixed with known endmembers and abundances. The data has been generated in the following way: Choose the number of points \(N\), the number of endmembers \(p\), the dimensionality of the data \(d\) and a signal-to-noise ratio (SNR). Generate the \(d \times p\) matrix \(E\) of random numbers in the interval \([0, 1]\), containing columnwise the \(p\) endmembers. Then, generate a \(p \times N\) abundance matrix \(A\) where every column is a random sampling from the canonical simplex, hence respecting the positivity and sum-to-one constraints. For each column \(a\) of \(A\), this can be done by sampling \(p\) variables \(a_i\) independently and uniformly in \([0, 1]\), returning \(-\log(a_i)\) (yielding an exponential distribution), and normalizing. Finally, we generate a \(d \times N\) noise matrix \(\eta\) of random Gaussian numbers with a given SNR. According to [11], the SNR is defined as half the reflectance divided by the standard deviation of the noise. The data set \(x\) is then given by:

\[
x = EA + \eta
\]

We compare the proposed SPU algorithm with the FCLSU algorithm presented in [2] for testing the accuracy, and benchmarking the required computational time. Both algorithms are implemented in MATLAB. The implementation of the FCLSU algorithm turns the problem into an equivalent problem where only the positivity constraint needs to be obeyed, and uses the build-in MATLAB function \"lsqnonneg\" for solving this non-negatively constrained least-squares problem with the technique proposed by Lawson and Hanson [17]. This method has become a de-facto standard in linear spectral unmixing for determining the abundances after the endmember extraction phase. We must point out that the SPU algorithm can benefit from the simultaneous treatment of the entire data set, while the FCLSU algorithm treats every data point sequentially and independently from previous runs. These tests were done on a regular desktop machine with a Intel core2 quad 3.0 GhZ processor and 4 Gb of RAM.

First of all, we tested the numerical accuracy by calculating the abundance coefficients for several parameter sets with both algorithms, and comparing them one by one. We found that in a typical situation \((p = 5, N = 10^3, d = 4, SNR = 10)\), about 99.7% of the retrieved abundances will differ by less than \(10^{-7}\) from those obtained by the FCLSU algorithm (averaged over 100 runs). In the other 0.3%, some errors can be attributed to the breakdown of the assumption used in the algorithm, but even then, the typical errors usually stay very low. Similar results can be obtained for other values of \(p\), although it must be noted that for any value of \(p > 3\), examples can be artificially created which have significant errors. Because the results obtained by both algorithms are hence comparable in quality in practical use, we mainly focus on the computational performance.

In Fig. 5, we have plotted the runtime and the number of recursions needed to obtain the solution as a function of the SNR, for \(p = 15\) endmembers, dimensionality \(d = 50\), and \(N = 10000\) data points. We notice that, as can be expected, the runtime of the SPU algorithm scales linearly with the number of recursions. Second, the SPU algorithm has a significantly shorter runtime for all values of SNR. For very low SNR, the number of recursions needed strongly increases, which causes longer runtimes of the SPU algorithm. For larger values of SNR however, which are typical of most practical situations, the SPU algorithm will outperform the FCLSU algorithm by a factor of order 10-50, and often higher.

Next, we have plotted the runtime as a function of the number of endmembers \(p\) in Fig. 6 for both algorithms, with \(N = 10000, SNR = 10\) and \(d = 50\). The required processing time increases exponentially as a function of \(p\) for both algorithms. Again the SPU algorithm performs significantly better than the FCLSU algorithm. The slightly higher slope of the SPU result in Fig. 6 however indicates that for very high values of \(p\), the FCLSU algorithm can have a better performance than the SPU algorithm. Such high values of \(p\) are not often encountered in practice however.
The dependence of the runtime on the dimensionality of the data $d$ is presented in Fig. 7. It can be seen that the runtime of the FCLSU algorithm increases significantly for increasing $d$. The runtime of the SPU algorithm however is mostly insensitive to the dimensionality $d$, as the same value is found for the entire range of $d$. This is mostly due to the fact that the most computationally intensive steps of the SPU algorithm are the calculation of matrix inverses, where the size of each matrix depends only on the number of endmembers $p$, and not the dimensionality $d$. Again we notice that the runtime of the SPU algorithm is drastically smaller than the runtime of the FCLSU algorithm.

Finally, the dependence of the runtime on the number of data points $N$ is presented in Fig. 8. The runtime of the FCLSU algorithm is linear in the number of data points $N$, as can be expected from the fact that the FCLSU algorithm processes point per point. The runtime of the SPU algorithm appears to scale linearly in $N$ as well, although this relation is not so clear from Fig. 8. The runtimes of the SPU algorithm are again much lower than those of the FCLSU algorithm. Furthermore, the slope of the SPU curve is smaller than that of the FCLSU curve, indicating that the difference in computational performance between both algorithms will only increase with increasing $N$.

Execution of both tests with a profiler show that for the FCLSU algorithm, $75\%$ of the runtime is used for calculating matrix pseudo-inverses, while the other $25\%$ of the time is mainly used for memory management and other matrix manipulations. For the SPU algorithm, virtually all time is dedicated to matrix manipulations. A careful optimization based on studying the profiling output, and using a compiled programming language, can yield further significant increases in performance.

C. Execution on the AVIRIS Cuprite data set

As we have seen in previous section, the SPU algorithm will in most cases have a significantly lower runtime than the FCLSU algorithm when executed on synthetic data. Next, we
evaluate the performance of both algorithms when executed on real hyperspectral data sets. The data set under consideration is the well-known AVIRIS Cuprite data set, obtained over the Cuprite mining region in Nevada, USA (see e.g. [35] for a detailed description of this data set). Specifically, we used the entire “f970619t01p02_r02_sc04.a.rfl” file, resulting in a $512 \times 614$ image. After elimination of several bands with low SNR, we kept 188 of the 224 spectral bands [36]. An approximate color image of this data set is shown in Fig. 9.

Since the unmixing algorithms are supervised, and hence assume that the endmembers are known, we first have to run an endmember extraction algorithm to locate these endmembers. The algorithm chosen is the NfindR algorithm [9], for its proven performance on this data set. Furthermore, we must determine the number of endmembers $p$ to extract, which is not a trivial task. A too low number will not yield good unmixing results, since the retrieved endmembers themselves will be mixtures of several actual pure materials. A too high number of endmembers will result in many unidentifiable endmembers, mostly corresponding to spectral noise, and multiple endmembers corresponding to the same pure material [37].

Ideally, the intrinsic dimensionality of the data set in spectral space should be determined, from which the necessary number of endmembers can be derived, e.g. via the virtual dimensionality technique [38]. Another method for estimating $p$ is by using ground data, e.g. the number of known different minerals in the region under consideration. Such methods lead to an estimate of the number of endmembers of order $O(10)$. Therefore, we have extracted the endmembers and unmixd the Cuprite data set for all values of $p$ in the range $[8, 15]$. For every value of $p$, the found endmembers correspond to those expected in this data set, e.g. alunite, kaolinite, buddingtonite, calcite or muscovite. Therefore we do not focus on the quality of the endmember extraction, but on the runtimes needed for the subsequent unmixing algorithm. Hence, we use the NfindR algorithm to extract different sets of endmembers, depending on the value of $p$, and use the SPU and FCLSU algorithm to unmix the data with respect to the found endmembers.

A typical unmixing result is presented in Fig. 10, for the alunite endmember. This abundance map can also be found in the literature (e.g. [9]). As can be seen in Fig. 10, both the FCLSU and the SPU algorithm generate qualitatively identical abundance maps. The runtimes of both algorithms as a function of the number of endmembers $p$ is shown in Fig. 11. It is clear that the SPU algorithm has superior performance compared to the FCLSU algorithm on this data set, at least for $p \in [8, 15]$.

V. CONCLUSIONS

We have introduced a new method for supervised linear spectral unmixing of hyperspectral data, based upon the equivalence between solving the fully-constrained least-squares problem, and the projection of a point onto a simplex. This simplex projection unmixing (SPU) algorithm uses a sequence of orthogonal projections on sub-simplices, decreasing the dimensionality of the problem by one with each step. The algorithm can be implemented recursively, and is well fit to deal with large amounts of data because many operations that depend solely on the endmembers need to be executed only once for all data points. A concrete implementation of the algorithm in pseudo-code is given, along with an explanation of all the steps.
We have tested the performance of the algorithm on simulated data, and on the AVIRIS Cuprite hyperspectral data set, and compared the results and the runtimes with those obtained with the well-known FCLSU algorithm from [2]. We have observed that the SPU algorithm and the FCLSU algorithm both yield almost identical results on real data sets. The runtimes of the SPU algorithm however are drastically lower for most practical purposes, and only for very high numbers of endmembers, the SPU algorithm can perform worse than the FCLSU algorithm. Also on the AVIRIS data, the SPU algorithm has a superior computational performance for several values of the number of endmembers.

Future work include a detailed profiling of the different steps of the algorithm, seeking further ways to improve the computational performance. Also, the assumption used concerning the bissective cones in the derivation of the algorithm should be replaced with an alternative that always yields the correct results, and is preferably provably correct. We are also working on a distance-geometry based version of the SPU algorithm, allowing, amongst other, the use of geodesic distance matrices in the unmixing algorithm. Furthermore, the simplex projection problem and the constrained unmixing problem are recurrent problems occurring in many scientific fields. This same algorithm can be used to solve such problems outside the scope of spectral unmixing, and identifying such fields is another task.

ACKNOWLEDGEMENTS

We would like to thanks the anonymous reviewers for providing us with a counter-example to the assumption used, and for interesting suggestions for improving a first version of this document.

REFERENCES


Rob Heylen (M’10) received the B.S. degree, the M.S. degree and the Ph.D. degree in physics, with work in the field of statistical mechanics, from the Katholieke Universiteit Leuven, Leuven, Belgium, in 2001, 2003 and 2008, respectively. In 2009, he became a post-doctoral researcher with the Vision Lab, Department of Physics, University of Antwerp, Belgium. His main areas of research interest are hyperspectral image processing and computational physics.

Dževdet Burazerović (M’00) received his M.S. degree in electrical engineering from Eindhoven University of Technology, in 2000. Since 2009, he has been with the Vision Lab, Department of Physics, University of Antwerp, where he is working as researcher and towards obtaining a PhD degree. His current research interests include hyperspectral image processing and data/pattern analysis.

Paul Scheunders (M’98) received the B.S. degree and the Ph.D. degree in physics, with work in the field of statistical mechanics, from the University of Antwerp, Antwerp, Belgium, in 1983 and 1990, respectively. In 1991, he became a research associate with the Vision Lab, Department of Physics, University of Antwerp, where he is currently a professor. He has published over 120 papers in international journals and proceedings in the field of image processing and pattern recognition. His research interest includes wavelets and multispectral image processing.