A FAST GEOMETRIC ALGORITHM FOR SOLVING THE INVERSION PROBLEM IN SPECTRAL UNMIXING

Rob Heylen, Paul Scheunders

IBBT-Visionlab, University of Antwerp
Universiteitsplein 1, Building N
B-2610 Wilrijk (Antwerp), Belgium

ABSTRACT

A well-known problem in hyperspectral unmixing is the estimation of the abundances once the endmembers are known, while respecting the constraints on these abundances. Recently, we have presented the simplex projection unmixing algorithm for solving this inversion problem, based on the equivalence of the constrained unmixing problem and geometric projection onto a simplex. This algorithm however does not yield the correct solution in all cases, and counterexamples can be easily found. In this paper, we integrate the simplex projection algorithm with an efficient algorithm for validating candidate solutions. When a solution is rejected, the algorithm can be restarted from a better starting point, until a correct solution is found. The results of this validated simplex projection algorithm are shown to be identical to those obtained via other methods, over a wide variety of configurations. Furthermore, we show that this algorithm outperforms the fully constrained least-squares algorithm, except when the number of endmembers is high.

Index Terms— Hyperspectral imaging, spectral unmixing.

1. INTRODUCTION

Spectral unmixing of hyperspectral imagery [1] usually consists of two stages: an endmember extraction stage, followed by an unmixing or inversion stage. The latter process tries to optimally estimate the abundance of each endmember in each pixel, where the used error measure is usually the reconstruction error between the actual pixel and the pixel reconstructed from the endmembers and their respective abundances. In the linear mixing model, this reconstruction error equals the Euclidean distance between the two points. Since one assumes two constraints on the abundances, positivity and sum-to-one, the allowed pixel values will lie in a simplex in spectral space, spanned by the endmembers. This interpretation leads to constrained least-squares techniques, of which the fully-constrained least-squares unmixing (FCLSU) technique presented in [2] is a well-known example, used in many practical applications.

It is shown in [3] that solving the inversion problem is equivalent to a simplex projection operation, where one projects onto the nearest point inside the simplex. Due to this equivalence, one can use a simplex projection algorithm for solving the inversion problem. This resulted recently in the simplex projection unmixing (SPU) algorithm [4], which exploits exactly this equivalence. However, the SPU algorithm uses a slightly flawed method to estimate endmembers with zero abundance, which leads to unmixing results that are sometimes not the true solution. Although in practical data sets the fraction of erroneously unmixed pixels is low (of order 0.3% of all pixels), it is desirable to adapt the algorithm in such a way that the results correspond exactly to those obtained via other methods, such as FCLSU.

In this paper, we present an extended version of the SPU algorithm, where we add additional steps to the algorithm for validating the results, and restarting the algorithm from a better starting point in case a candidate solution is rejected. Checking whether a candidate solution is truly the closest point can be done efficiently via dot products, exploiting the convexity of the simplex. We compare the results obtained by this validated simplex projection algorithm (VSPU), and its computational runtime, with the FCLSU algorithm and the original SPU algorithm on both artificial data, and the well-known Cuprite data set.

2. THE ALGORITHM

Consider a \(d\)-dimensional spectral space \(\mathbb{R}^d_+\) containing \(p\) endmembers, forming the set \(I = \{e_i\}_{i=1,...,p}\). These endmembers span a simplex \(S(I)\), and every point \(y \in S(I)\) can be written as a convex linear combination of the endmembers in \(I\): \(y = \sum_i a_i e_i, \forall i: a_i \geq 0\) and \(\sum_i a_i = 1\). The abundances \(a = (a_1, \ldots, a_p)\) can be considered as normalized barycentric coordinates with respect to the endmembers [5]. Consider a point \(x \notin S\). Solving the inversion problem corresponds to finding the abundances of the point \(y \in S\) which minimizes the reconstruction error, defined as the \(L_2\)-distance between \(x\) and \(y\). This interpretation allows us to define a projection operator \(P_I\), which projects the point \(x\) onto the
simplex $S(I)$: $P_I(x) = y \in S(I) \iff \forall z \in S(I): \|z - x\|_2 \geq \|y - x\|_2$.

The SPU algorithm \cite{4} is a recursive algorithm for solving this simplex projection, based on the following observations (we define $Q_I$ as the operator that orthogonally projects a point onto the hyperplane through the points in $I$):

- Orthogonal projection onto the simplex plane is an invariant operation with respect to the simplex projection operator: $P_I(Q_I(x)) = P_I(x)$.

- If $Q_I(x) \notin S(I)$, then $y = P_I(x) = \sum_i a_i e_i$ will lie on the boundary of the simplex $S(I)$, and there will be at least one abundance $a_j = 0$.

- Let $I_j = I/\alpha e_j$, and let $y = P_I(x) = \sum_i a_i e_i$. If $a_j = 0$, then $P_I(x) = P_I(x)$.

This leads to the following recursive algorithm for the operation $P_I(x)$: Determine $y = Q_I(x)$. If $y \in S$, we have found the projection, yielding the abundance vector $a$ with respect to the endmembers in $I$. Otherwise, since there is at least one abundance $a_j$ zero, determine which one, and remove the corresponding endmember $e_j$ from the set of endmembers $I$, yielding $I_j$. The other abundances $\{a_i\}_{i \neq j}$, can be found by recursively calculating $P_{I_j}(x)$.

To determine which endmember has zero abundance in the projection of a given point $x$, the following technique was suggested: Determine the incenter $c$ of the simplex, and draw a line from $c$ to $Q_I(x)$. This line will cut a single sub-simplex $S(I)$. The corresponding abundance $a_i$ of endmember $e_i$ is assumed to be zero.

This assumption works well in most practical cases, but is not always valid, and one can easily create or generate counter examples. One can also show that replacing the incenter $c$ with some other internal point will never yield an algorithm that gives the correct solution for all cases. Therefore, for improving the SPU algorithm, we have elected to add an additional validation phase to the original algorithm.

To check whether the relation $y = P_I(x)$ holds, one can use the following observation: Because every point inside the simplex is a convex combination of the endmembers, and $y$ should be the point closest to $x$, the angle between the vector $x - y$ and $e_i - y$ should be larger than or equal to $\pi/2$ for every endmember $e_i$. Furthermore, equality arises if and only if the endmember has a non-zero abundance. Hence we have

$$y = P_S(x) \iff \forall i: (x - y) \cdot (e_i - y) \leq 0 \quad (1)$$

See also Fig. 1 for an illustration of this relation with a three-dimensional simplex. If an endmember is found that violates this relation, we know that the obtained solution is wrong. To find the correct solution, we can then create a new simplex to restart the algorithm, spanned by the endmembers with non-zero abundance and those endmembers that violate relation (1). As long as the resulting set of endmembers is smaller than the initial set, the algorithm will eventually find the correct solution. The resulting validated SPU algorithm then becomes:

1. Let $y$ be the orthogonal projection of $x$ onto the hyperplane through the endmembers in $I$. Determine whether $y$ lies inside the simplex $S(I)$. If so, go to step 4. Otherwise, go to the next step.

2. Calculate the internal point $c$, and determine the sub-simplex $S(I/e_i)$, that cuts the line between $c$ and $y$.

3. Remove $e_i$ from $I$, set $a_i$ to zero, and go to step 1.

4. Validation: We have found a candidate solution $y$ after a sequence of orthogonal projections. Determine the set $J = \{e_j\}_j$, for which $(e_j - y) \cdot (x - y) > 0$. If $J \neq 0$, let $I = I \cup J$ and go to step 1. Otherwise, end the algorithm.

Furthermore, we know that no matter the choice for the internal point $c$, there will be cases where the validation phase of the algorithm will reject solutions. Internal points that are computationally easy to calculate, such as the barycenter, will yield more erroneous solutions after step 3 of the algorithm (see experimental section), but might result in an algorithm that is overall faster when compared to using e.g. the computationally expensive incenter. From our tests, we have found that the barycenter of the simplex yields very good results. We have not encountered a situation where the algorithm does not finish in a finite number of iterations, or gives a solution that differs from the one obtained with FCLSU.

Finally, while the presented description of the algorithm is for unmixing a single point, it is easy to implement a version
that can deal with an entire data set at once: At each iteration, the algorithm splits up the input data set into at most \( p \) subsets, each containing the pixels associated with one of the \( p \) sub-simplices. One then recursively calls the algorithm for each subset, with the number of endmembers decreased by one. The resulting recursion tree algorithm is computationally much more efficient than a version that treats point-per-point, since many operations (e.g. calculation of projection matrices) need to be executed only once for the entire data set.

3. ARTIFICIAL DATA

To test the obtained results and the performance of the algorithm, we have created artificial data sets with a given number of endmembers \( p \), number of bands \( d \), and \( N \) data points \( \{x_i\}_{i=1,...,N} \) to unmix. Each spectral band of each endmember was chosen randomly from a standard Gaussian distribution, as well as all bands of all data points \( x \). The majority of the data points will in this case not lie within the simplex formed by the endmembers.

The first experiment displays the effect of the choice of the internal point \( c \) on the number of rejected points in the validation phase of the algorithm. In Fig. 2 we have plotted the percentage of rejected points as a function of the number of endmembers \( p \). For each value of \( p \), we have generated 100 random endmember configurations in a spectral space with \( d = p - 1 \) dimensions. For each configuration, we have unmixed 100 random points, and counted the number of rejected points in the validation step of the algorithm. The displayed values are the averages over these data sets.

From this figure, it is clear that using the incenter for \( c \) will yield a much larger percentage of points that are correctly projected when the algorithm enters the validation phase, compared to using the barycenter. Calculating the incenter however is a much more involved operation, and the runtime of the algorithm when using the incenter is longer than when the barycenter is used. The final results obtained by both algorithms are virtually identical to those obtained by FCLSU, and differ only due to insignificant numerical errors.

The difference in runtimes can be clearly seen in Fig. 3, where we have plotted the runtime of the presented algorithm for the two cases (with barycenter and incenter as internal point). For comparison, we have also added the runtimes of the original SPU algorithm, and those obtained by the FCLSU algorithm. From this figure it is clear that the validated SPU algorithm which uses the barycenter has a computational performance that is comparable to the original SPU algorithm, and significantly outperforms the variant that uses the incenter. Both algorithms are also much faster than FCLSU as long as the number of endmembers \( p \) stays relatively low, but will show much larger runtimes than FCLSU if \( p \) becomes large.

This effect is mostly due to the construction of the artificial data sets: Since the endmembers are chosen randomly from a Gaussian distribution, as well as the data points, the majority of data points will lie relatively far away from the simplex in spectral space, and hence the SPU algorithm will require many recursions before finishing for larger values of \( p \). This is not the case in real data sets, where the mixed data points will usually lie close to the simplex spanned by the endmembers.

4. THE CUPRITE DATA SET

To test the performance of the VSPU algorithm in practical applications, we have unmixed the well-known AVIRIS Cuprite data set for different values of the number of endmembers. A subset of \( 300 \times 364 \) pixels was used, with 51
spectral bands in the IR range (1.98-2.48 μm). This data set is depicted in Fig. 4 as an approximated true-color image.

For a given value of $p$, we have used the N-FindR algorithm [6] to extract $p$ endmembers from the data set. These endmembers were subsequently used to invert the linear mixing equation, using the VSPU, SPU and FCLSU algorithms. For the VSPU algorithm, the version that uses the barycenter was used, since the tests on the artificial data already show that this version is significantly faster than the version that uses the incenter.

The abundances obtained by the VSPU and FCLSU algorithm were identical, while the SPU algorithm had a low percentage of wrongly unmixed pixels. The runtimes of both algorithms are plotted in Fig. 5, as a function of the number of endmembers $p$. For all the values of $p$ treated, the runtimes of the SPU and VSPU algorithm are significantly lower than the FCLSU algorithm. Only for larger values of $p$ will the (V)SPU algorithm perform worse than the FCLSU algorithm.

5. CONCLUSION

We have presented an improved version of the SPU algorithm, which includes an efficient validation step, and can restart the algorithm when a candidate solution is rejected. Since this validated simplex projection unmixing (VSPU) algorithm depends on the choice of an internal simplex point, two possibilities are explored: The incenter, also used in the original description of the SPU algorithm, and the barycenter. The latter possibility is shown to yield a higher rejection percentage in the validation phase of the algorithm, but results overall in a computationally more performant algorithm, caused by the computational simplicity of calculating the barycenter, compared to the incenter. Since the unmixing results produced are identical to those obtained with the FCLSU algorithm, we focus mainly on the runtime of the algorithm when applied to real data sets. It is shown that the VSPU algorithm significantly outperforms the FCLSU algorithm on the Cuprite data set, as long as the number of endmembers is not too large.

6. REFERENCES


