Fast Approximate Construction of Best Complex Antipodal Spherical Codes

Miguel Heredia Conde, Member, IEEE, and Otmar Loffeld, Senior Member, IEEE

Abstract—Compressive Sensing (CS) theory states that real-world signals can often be recovered from much fewer measurements than those suggested by the Shannon sampling theorem. Nevertheless, recoverability does not only depend on the signal, but also on the measurement scheme. The measurement matrix should behave as close as possible to an isometry for the signals of interest. Therefore the search for optimal CS measurement matrices of size \( m \times n \) translates into the search for a set of \( n \) \( m \)-dimensional vectors with minimal coherence. Best Complex Antipodal Spherical Codes (BCASCs) are known to be optimal in terms of coherence. An iterative algorithm for BCASC generation has been recently proposed that tightly approaches the theoretical lower bound on coherence. Unfortunately, the complexity of each iteration depends quadratically on \( m \) and \( n \). In this work we propose a modification of the algorithm that allows reducing the quadratic complexity to linear on both \( m \) and \( n \). Numerical evaluation showed that the proposed approach does not worsen the coherence of the resulting BCASCs. On the contrary, an improvement was observed for large \( n \). The reduction of the computational complexity paves the way for using the BCASCs as CS measurement matrices in problems with large \( n \). We evaluate the CS performance of the BCASCs for recovering sparse signals. The BCASCs are shown to outperform both complex random matrices and Fourier ensembles as CS measurement matrices, both in terms of coherence and sparse recovery performance, especially for low \( m/n \), which is the case of interest in CS.

Index Terms—Best Complex Antipodal Spherical Codes, BCASC, Best Spherical Codes, BSC, codes, Compressive Sensing, CS, coding theory, coherence optimization, Welch bound.

I. INTRODUCTION

A question that naturally arises by simply observing the nature is what is the best packing of \( n \) points in an \( m \)-dimensional (hypo-)sphere, where \( m \) is typically either two or three. Best packing is to be understood as the arrangement that maximizes the minimum distance between points. Regular polygons, ubiquitous present in nature, are the best packing of 2-dimensional points in the sphere of corresponding dimensionality, that is, the circle. Probably the most paradigmatic example is the hexagonal shape of honeycombs or paper wasp nests. Also the compound eyes of insects are packed hexagonally. Also some 3-dimensional packing schemes seem to be present, e.g., in the arrangement of the places of exit on the surface of spherical pollen grains. It was the Dutch biologist Pieter M. L. Tammes [1] who conjectured that, for a given size of the pollen grain and area required for each exit place, the arrangement was such that the number of exit places was maximized, i.e., it was the best packing of 3-dimensional points on the sphere. In honor of Tammes, the problem of finding this optimal arrangement of points on a spherical surface or, equivalently, finding how many spherical caps of a given radius can be placed on a unit sphere without overlapping, is classically known as the Tammes' problem or hard-spheres problem [2]. This problem has been thoroughly analyzed and solutions exist for different numbers of points [3], [4]. A monograph on the pursuit of best packings with real life examples can be found in [5].

Note the close relationship between Tammes’ problem and the so-called Thomson’s problem, namely, determining a stable distribution of electrons able to move freely on the surface of a 3-dimensional sphere, motivated by the atom model first suggested about 1900 by Lord Kelvin and formally described later in [6] by Sir J. J. Thomson. Several methods exist for accurately approaching solutions to Thomson’s problem, e.g., based on random walk [7], on steepest descent [8], on constrained global optimization (CGO) [9], using genetic algorithms [10], a generalized simulated annealing algorithm [11], [12] (available as general-purpose software package [13]) and Monte-Carlo-based simulations [14], [15].

At the light of the amount of research on this subject along the twentieth century, one is tempted to consider the problem of finding the best packing points on the sphere as solved. Nevertheless, such resolution is to be questioned when taking into consideration that this problem can be seen as a very specific instance of the more general problem of packing subspaces in Grassmann manifolds. A Grassmann manifold or Grassmannian refers to a space parametrizing all linear subspaces of a vector space (which could be \( \mathbb{R}^m \) or \( \mathbb{C}^m \)) of a given dimension. If the dimension of the subspaces is chosen to be one, the corresponding Grassmannian is the space of lines originating at the origin of the vector space. If the vector space is \( \mathbb{R}^m \), this Grassmannian will be denoted as \( G(\mathbb{R}^m, 1) \) and finding the best packing of \( n \) subspaces (lines) into it is equivalent to the problem of finding the optimal arrangement of \( n \) \( m \)-dimensional points on the surface of the \( m \)-hypersphere. The general problem of subspace packing has been object of study since the 1960s [16], motivating a number of works which cannot be included here for space restrictions and the culmination of which can be seen in the detailed numerical study by J. H. Conway [17].

Nowadays best subspace packings find multiple application areas, being of special interest in Multiple-Input Multiple-Output (MIMO) and Code Division Multiple Access (CDMA) wireless systems. In fact, communication strategies based on quantization via subspace packings have been incor-
porated into the IEEE wireless metropolitan area network (WirelessMAN™) standard [18]. In general, it is clear that best packings are useful in any application where codes with low mutual coherence are required, since maximizing the minimal distance between codewords leads low mutual coherence. More specifically, codes with optimal or close-to-optimal mutual coherence have earned renewed importance thanks to the advent of the groundbreaking theory of Compressive Sensing (CS). According to this novel mathematical theory, many real-world signals can be exactly recovered from a number of measurements that is much lower than that suggested by the Shannon sampling theorem. CS takes profit of the fact that most real signals exhibit some structure and can be sparsely represented in an appropriate basis, in other words, they can be represented in a domain where their effective dimensionality is much lower than the original signal dimensionality. Nevertheless, some requirements are to be observed to attain successful signal reconstruction from few measurements, one of them being that the columns of the measurement matrix have to exhibit as low mutual coherence as possible. The measurement matrix models the linear transformation from the $n$-dimensional domain where the sparse signal lives to the $m$-dimensional measurement domain, with $m \ll n$. If intercolumn coherence is to be minimized, we face the problem of finding a set of $n$ $m$-dimensional codewords constituting an optimal code in terms of coherence.

In coding terminology, if the codewords lie on the surface of the unit sphere, they are known as Spherical Codes (SC). We leave the details on the different classes of SCs and construction schemes for forthcoming sections. In general, an SC is called Best Spherical Code (BSC) if the point arrangement it describes maximizes the minimum distances between points. At the time being, the sphere packings obtained by N. J. A. Sloane [19] are often considered putative solutions to the problem, i.e., the definitive BSCs in real spaces. Nevertheless, as pointed out in [20], for a given problem dimensionality, the vector set of minimal coherence is given by a Best Complex Antipodal Spherical Code (BCASC). In other words, the optimal CS measurement matrix in terms of coherence and supposing absence of a priori information is given by a BCASC. Consequently, methods for generating BCASCs in an efficient manner are invaluably precious from a CS point of view.

In this work we propose an approximate implementation of the algorithm for constructing BCASCs proposed in [20] which provides a considerable reduction of the complexity of the algorithm. This way, our approach can be used to generate BCASCs or, equivalently, CS measurement matrices, of very large dimensionality, thus paving the way to their implementation in real applications. We will also show that our approximation not only does not degrade the quality of the resulting BCASCs, but yields slightly better codes in terms of coherence for large $n$. The rest of the paper is structured as follows: Section [I] reviews some of the related work, pointing out their main peculiarities. In Section [II] we briefly provide the fundamentals we build our work upon, namely, Compressive Sensing, Spherical Codes and the Approximate Nearest Neighbor (ANN) search. Our approximate BCASC construction method is outlined and explained in Section [III]. The results of an experimental evaluation of the performance of our approach are presented in Section [IV] where we compare to the original method in [20]. The paper ends with the conclusions given in Section [V].

II. RELATED WORK

The methods for construction BCASCs can be roughly divided in two categories: analytical approaches and numerical approaches. In principle, analytical approaches should be preferred over numerical methods, but in practice they can only provide solutions for specific combinations of number of codewords and codeword dimensionality. Additionally, analytical approaches may impose some further structure on the codewords, thus artificially reducing the degrees of freedom available to optimize the codes. If the aim is to generate BCASCs of arbitrary size, with special focus on large sizes, numerical methods are the only feasible way. In this work we build upon the recent BCASC construction algorithm proposed in [20]. Nevertheless, there has been a number of previous works on numerical methods for approaching BCASCs. Note that many of the construction schemes that will be introduced in the following are not restricted to find BCASCs, but a MWBE (Maximum Welch Bound Equality) codebook or, in general, solutions to the subspace packing problem in Grassmannian manifolds, which is the case in works related to MIMO communications. Nevertheless, for a subspace dimension of one, the latter general formulation boils down to the problem of packing Grassmannian lines, i.e., finding the best packing of points on the hypersphere.

In [21], the authors draw from previous research on tight frame construction by R. Balan and I. Daubechies, who proposed constructing the tight frames by selecting the first $m$ components of the $n$-dimensional vectors of an $n \times n$ discrete Fourier transform (DFT). While this may be directly used for constructing $n$ $m$-dimensional complex codes for relatively low values of $m$, it is clear that when $m \to \infty$ the coherence between contiguous codewords tends to one. The novelty in [21] is that the selection of the $m$ DFT rows is not deterministic, but the result of an optimization process. The optimization consists in a random search that looks for the set of frequencies which yields minimal coherence between resulting codewords. In their simulations they consider $m = 8$ and up to $n = 2209$ codewords.

The authors of [22] provide both an analytic construction of optimal codes approaching the Welch bound and a numerical search method for those cases for which analytical construction does not apply. The numerical method is based on Lloyd’s algorithm. Lloyd’s algorithm [23], also known as Voronoi relaxation, is a well-known method for obtaining evenly spaced sets of points in subsets of Euclidean spaces. The original goal of the method was to attain an optimal finite quantization scheme, where optimality was meant in the sense of minimal average quantization error power. Given a probability density function, the quantization error power for each quantum partition takes the shape of a second order moment thereof, restricted to the domain covered by the partition.
Consequently, if the quantum partitions are fixed, their optimal quantum values are their centers of mass, while if the set of quanta is fixed, the best partition for each quantum is given by those points of the domain for which the Euclidean distance with respect to the quantum value is lower than those obtained with respect to any other quantum. An iterative formulation with two alternative steps of center of mass calculation and construction of the partitions (Voronoi cells) converges to an optimal quantization scheme. For completeness, recall that well-known clustering approaches, such as \(k\)-means, are fully based on the Lloyd’s algorithm, where the probabilistic distribution of data points is not given analytically, but as a set of particles. The K-SVD algorithm [24] for dictionary learning is often regarded as an extension of \(k\)-means clustering and, consequently, also known as generalized Lloyd’s algorithm (GLA). The idea in [22] was to use every quantizer codebook generated at each Voronoi iteration as candidate MWBE codebooks, supposing an uniform distribution of points on the unit hypersphere in \(C^n\). From all candidates, which could also come from many executions of Lloyd’s algorithm, the complex codeword with smallest coherence was chosen as approximate MWBE codebook. In posterior work [23] the authors adopt the GLA for generating the codebooks and provide lower bounds on the rate-distortion performance. Another evolution of Lloyd’s algorithm intended to generate optimal codebooks is proposed in [20], where the objective is matrix quantization to minimize capacity loss.

Three different methods are provided in [27] for generating evenly-distributed constellations of points on the complex unit hypersphere. The first is a greedy technique in which the constellation is constructed sequentially, while the second generates the entire constellation at once. The unaffordable computational cost of the latter alternative for large constellation sizes motivated a third approach, which consists of two sequential steps: generation of a low-size constellation using the latter approach and generation of the final constellation fusing several rotations of the initial one. The common denominator of the three approaches is that they all exploit the smooth geometry of the Grassmann manifold and use derivative-based techniques to optimize a smooth cost function that approximates the original non-differentiable objective, in this case the pairwise chordal Frobenius norm, which is to be maximized. Similarly, in [28] the coherence, which is a non-smooth function to be minimized, is approached by a family of potential functions that are smooth. A smoothing approximation has also been used in [29], where the Euclidean distance is approximated by an hyperbolic function for solving the classical Tammes’ problem. Inspired by the objective-smoothing approach in [27], a construction method is proposed in [30] that uses a sequence of smooth approximations to the objective function. Since the smooth approximation functions have continuous first and second order derivatives, conventional smooth optimization techniques can be used to solve the problem. Several smooth approximations to the maximum distance (to minimize) and to the Fubini-Study distance are presented, highlighting the usual compromise between convergence rate and stability in the latter case.

The construction method first introduced in [31] and further developed and analyzed in [32] exploits the fact that unitary transforms representing any point on the Grassmann manifold can be represented by the exponential of any element of the tangent space at the identity point. For clarity, in the rank one case we deal with in this paper this would mean an exponential of an \(n\)-dimensional vector with only \(n-1\) nonzero components. This is due to the fact that we pack subspaces of dimension one (lines) on the Grassmannian, which yields a tangent space of dimensionality \(n-1\). The core idea of the method is to design a coherent codebook in the tangent space, which yields an optimal non-coherent one on the Grassmannian. Further work exploiting the exponential parametrization of the Grassmannian can be found in [33], where different rotated lattices are used as initial codes in the tangent space.

Provided that all rotations of a spherical code (or set of subspaces, in the case of non-unitary subspace dimensionality) around the origin are equivalent, in [34] each configuration of subspaces is associated with a block Gram matrix. Note that the off-diagonal elements of the Gram matrix are related to the distances between pairs of subspaces. The novelty in [34] is the use of an alternating projection scheme for approaching a solution to the best packing. The algorithm works directly on the Gram matrix and alternates two steps for enforcing both structural conditions and spectral conditions. In practice, the algorithm keeps two estimates of the Gramm matrix: one strictly satisfying the structural conditions and the other strictly satisfying the spectral conditions. The alternating scheme aims to minimizing the Frobenius norm of the difference, thus making both estimates recursively approach to each other. Clearly, the solution lies in the intersection of the two constrain sets where the temporal estimates live. The importance of [34] does not only lie in the novelty of the approach, but in the underlying connection to CS theory. One of the structural conditions is an upper bound on the norm of off-diagonal elements of the Gram matrix. In the specific case of line packing, i.e., subspaces of unit dimensionality, this is equivalent, in CS terms, to an upper bound on the coherence of the sensing matrix.

An expansion-compression algorithm (ECA) is proposed in [35] for finding packings in Grassmann manifolds. The ECA scheme seems to be motivated by the fact that using the chordal distance yields degenerated constellations if a simple max-min (maximization of the minimum distance between codewords) scheme is applied. To overcome this issue, an alternating scheme between a step of max-min and a subsequent step of min-max is proposed. The former step is called expansion and the latter compression. The authors observe that using the Fubini-Study distance as a metric degenerated constellations do not occur and one can do with a conventional max-min scheme, thus avoiding the compression step.

Building upon M. Elad’s method for obtaining optimized projections for CS [36], the authors of [37] make use of frame theory concepts in order to construct tight frames that are the nearest to the obtained optimized projections. This results in significant mutual coherence reduction. In the sequel [38] the authors extended this work with an averaged projections version of the algorithm instead of the initial alternating
projections formulation. The difference is that in the new version the temporal estimates of the Gram matrix obtained after applying each of the steps of the algorithm to the Gram matrix that resulted from the previous iteration are stored and the Gram matrix for the next iteration is obtained as the average of these stored projections. A further step into the very desirable overlap between optimal packings and CS was taken in [39]. The author explicitly points out the equivalence between optimal unit-norm frame design and optimal packing of points on the unit sphere. Differently from the methods in [34] or [37], which operate on the Gram matrix, a method called Iterative Decorrelation by Convex Optimization (IDCO) is proposed that works directly on the frame being optimized, i.e., on the code itself. The algorithm tries to find at each iteration a frame with lower coherence than the previous, while constraining the new frame to be close to the previous one. From an sphere packing point of view, the closeness constraint avoids that the spherical code being constructed rotates endlessly around the origin, precluding convergence, while the coherence minimization (a min-max scheme) forces the packing to be as uniform as possible. The performance of the algorithm was demonstrated creating frames of fairly large dimensionality.

Recently, a numerical method for computing BCASCs has been proposed in [20]. The method can be regarded as an extension of the BASC construction in [40] to the complex case, which, in turn, is based on Lazić’s early works on the construction of BSCs [41], [42]. As it will be explained in Section III-B, from a computational point of view, the novelty in [20] is the introduction of an integral over all complex rotations of each temporal codeword that influence a specific codeword when computing the resulting force of the former over the latter. This is necessary to transfer the concept of antipodality to the complex case.

Some of the methods presented so far were either exclusively or, at least, initially designed to construct codes, packings, frames, sensing matrices, etc. in the real space. In some cases, complex extensions are available and, in fact, some of the previous methods were designed to work natively in complex space. Nevertheless, in such cases the computational cost is higher and the methods are only evaluated for relatively low numbers of codewords of low dimensionality. There is, indeed, a lack of work where constructions of large close-to-optimal complex codebooks are presented. To the best of our knowledge, the tables in [20] provide the most complete comparative benchmark so far and are limited to codes with \( m \leq 64 \) codewords of dimensionality \( n \leq 4 \) (or only \( m = 16 \) for \( n = 5 \)).

III. FUNDAMENTALS

In the following we will provide the fundamentals of Compressive Sensing (CS) theory (Section III-A), a new sensing paradigm that aims to overcome the explosion in the volume of data acquired by high resolution sensors operating at the Nyquist rate. The exposition of the CS sensing model will immediately lead to the necessity of a matrix with low intercolumn coherence, that is, sets of vectors with low mutual coherence. The search of the set of vectors with lowest coherence brings the link to spherical codes, which will be presented and analyzed in detail in Section III-B. The methods for obtaining such set increase their ability to approach the theoretical lower bound on coherence, they also get more complex in computational terms and approximate methods become necessary. In Section III-C we will give an overview of Approximate Nearest Neighbor (ANN) searches, the core idea for speeding up the construction of BCASCs.

A. Compressive Sensing

The Shannon sampling theorem is the keystone of classical sampling theory and states that a continuous signal is completely determined by a number of equidistant samples acquired at a rate that is twice the maximum frequency contained in the signal. While this is true and actually the basis of almost every signal acquisition system, it implies the generation of huge data volumes when the signals to sense are allowed to have large bandwidth. Such volumes of data are too large to be stored directly, often too large to be even transmitted, and thus a process of data compression right after sensing becomes necessary. In this scenario, compressive (or compressed) sensing (CS) arises as a novel sensing paradigm, in which the compression is performed at sensing and not immediately afterwards, thus obtaining fewer but more informative measurements.

While the Shannon sampling theorem requires the signal to be bandlimited, CS theory imposes the more general requirement of being sparse in some known representation basis. More specifically, CS states that a finite-dimensional signal that admits a sparse or compressible representation in some known basis or tight frame can be exactly recovered from few non-adaptive measurements if certain conditions are satisfied. We restrict our attention to discrete signals with real or complex coefficients. Let \( \vec{x} \in \mathbb{C}^n \) be the vector corresponding to a discrete signal. Then, the \( l_0 \) norm of \( \vec{x} \) is defined as:

\[
\|\vec{x}\|_0 := \lim_{p \to 0} \|\vec{x}\|_p = |\text{supp} (\vec{x})| \tag{1}
\]

that is, the cardinality of the support of \( \vec{x} \), and \( \vec{x} \) is called an \( s \)-sparse signal if:

\[
\|\vec{x}\|_0 \leq s \tag{2}
\]

in other words, if \( \vec{x} \) has, at maximum, \( s \) non-zero elements. Provided that we know that the signal obeys a sparsity constraint, the challenge is to reconstruct it from a reduced number of linear measurements \( m \ll n \). Thus, the classic CS measurement model is a severely underdetermined linear system of the form:

\[
\vec{y} = \mathbf{A}\vec{x} \tag{3}
\]

where \( \mathbf{A} \in \mathbb{C}^{m \times n} \) denotes the measurement matrix, which explains how the vector of measurements \( \vec{y} \in \mathbb{C}^m \) relates to the signal \( \vec{x} \). This matrix is often the composition of a sensing matrix, which models the real sensing process and a
representation basis or dictionary, which enables recovering signals that are not sparse, but admit a sparse representation. Clearly, provided that $m \ll n$, the challenge is to design $A$ in such a way that preserves the information in $\tilde{x}$, despite the dimensionality reduction, in other words, that condenses a limited information content living in a high-dimensionality space into a space of lower dimensionality.

The linear system in Eq. 3 is massively underdetermined and, therefore, cannot be solved directly. Solving Eq. 3 via least squares would spread the power in the measurement vector over all coefficients of $\tilde{x}$, yielding dense solution that does not fit our a priori knowledge on the sparsity of $\tilde{x}$. Therefore, we should rather look for the solution with minimal $l_1$ norm, between all the possible solutions satisfying Eq. 3. Unfortunately, it is well known that finding a solution to a constrained $l_0$ minimization is, in general, NP-hard [48]. In the CS literature there are two main approaches to overcome this difficulty: either to relax the $l_0$ minimization into an $l_1$ minimization, or to use a greedy algorithm to find the sparse support of the signal in an incremental fashion. Recovery guarantees are typically given under the assumption of solving Eq. 3 by $l_1$ minimization, which yields:

$$\hat{x} = \arg\min_{\tilde{x}} ||\tilde{x}||_1 \text{ subject to } y = A\tilde{x}$$  \hspace{1cm} (4)

Eq. 4 can be efficiently solved as a linear program [49]. A fundamental question in CS is under which conditions Eq. 4 is equivalent to the desired $l_0$ minimization [50]. The success of the signal recovery process, provided that the sparsity constraint is satisfied, depends on whether $A$ possesses some properties. The most intuitive property is the so-called Null Space Property (NSP) [51], which is based on the analysis of the null space of $A$. There exist several formulations of the NSP, being the following one obtained in the common case of an upper bound on the recovery error of compressible signals that are not sparse, but admit a sparse representation.

A matrix $A$ is said to satisfy the RIP of order $k$ if:

$$\|\tilde{x}_{\Omega_k}\|_2 \leq \|\tilde{x}\|_2 \leq C_0 \frac{\sigma_k(\tilde{x})}{\sqrt{k}} = C_0 \frac{\|\tilde{x}_{\Omega_k}\|_1}{\sqrt{k}}, \tilde{x} \in \mathcal{N}(A),$$ \hspace{1cm} (5)

where $\mathcal{N}(A)$ denotes the null space of $A$, $\Omega_k$ denotes any set of $k$ support indices, and $\Omega_k$ the complement set of $\Omega_k$, so that $\tilde{x}_{\Omega_k}$ denotes the vector $\tilde{x}$ with support restricted to $\Omega_k$, with $|\Omega_k| = k$. $\sigma_k(\tilde{x}) = \|\tilde{x}_{\Omega_k}\|_1$ is the so-called best $k$-term approximation error. The NSP prescribes that vectors in $\mathcal{N}(A)$ should not be too concentrated in a small subset of indices [52], i.e., should be as dense as possible. Satisfaction of the NSP of order $2s$ can be linked to the satisfaction of an upper bound on the recovery error of compressible signals and exact recovery of $s$-sparse signals.

A different condition for recoverability is given by the well-known Restricted Isometry Property (RIP), also known as Uniform Uncertainty Principle (UUP), first introduced in [53] and further analyzed in [54]. A matrix $A$ is said to satisfy the RIP of order $k$ if there exists a constant $\delta_k \in (0, 1)$ such that:

$$(1 - \delta_k) \|\tilde{x}\|_2^2 \leq \|A\tilde{x}\|_2^2 \leq (1 + \delta_k) \|\tilde{x}\|_2^2, \forall \tilde{x} \in \Sigma_k$$ \hspace{1cm} (6)

where $\Sigma_k$ is the subset of all $k$-sparse vectors and $\delta_k$ is known as the $k$-restricted isometry constant. The RIP ensures that the sensing matrix is close to an isometry for $k$-sparse vectors, i.e., that the transformation preserves the $l_2$ distances between pairs of $k$-sparse vectors to some extent. If $A$ satisfies the RIP of order $2s$ with $\delta_{2s}$ low enough, e.g., $\delta_{2s} < \sqrt{2} - 1$ from [54] or $\delta_{2s} < \frac{3}{4\sqrt{2}}$ from [55], then successful recovery of the $s$-sparse vector $\tilde{x}$ via $l_1$ minimization is guaranteed. The RIP of order $k$ implies that any set of $k$ columns of $A$ chosen at random should behave approximately as an orthonormal system [56]. In an orthonormal system, every basis vector is perfectly incoherent with all others, thus yielding null coherence. For completeness, we provide the following definition of coherence for the case of an arbitrary set of $n$ vectors of dimension $m$, which are to be identified with the columns of $A$ and in CS is often named matrix coherence [57] of $A$:

$$\mu(A) = \max_{u \leq v \leq n} |\langle \tilde{a}_u, \tilde{a}_v \rangle| \frac{\|\tilde{a}_u\|_2 \|\tilde{a}_v\|_2}{\|\tilde{a}_u - \tilde{a}_v\|_2}$$ \hspace{1cm} (7)

where $\langle \cdot, \cdot \rangle$ denotes (complex) scalar product. Clearly, in the typical case of normalized vectors, $\mu(A)$ is equivalent to the maximum scalar product between vector pairs and, consequently, only dependent on the angles between them. In other words, finding the optimal CS measurement matrix with normalized columns is equivalent to finding the arrangement of points on the unit sphere that maximizes the minimal angle between them when regarded as vectors. In terms of coherence, we face a minimization problem, which, in combination with Eq. 7 yields the following min-max formulation:

$$\mu_{\min}(A) = \min_{A \in \mathbb{C}^m \times n} \mu(A) = \min_{A \in \mathbb{C}^m \times n} \max_{u \leq v \leq n} |\langle \tilde{a}_u, \tilde{a}_v \rangle| \frac{\|\tilde{a}_u\|_2 \|\tilde{a}_v\|_2}{\|\tilde{a}_u - \tilde{a}_v\|_2}$$ \hspace{1cm} (8)

The remaining question is how lower coherence translates into a better RIP. As observed in [52], one can make use of the Geršgorin circle theorem (Theorem 2 of [58]), which states that the eigenvalues of a matrix $G \in \mathbb{R}^{n \times n}$ lie in the union of $n$ discs, $\bigcup_{i=1}^n d_i(c_i, r_i)$, centered at $c_i = g_{ii}$ and with radius $r_i = \sum_{j \neq i} |g_{ij}|$. Bounding the eigenvalues of the partial $k \times k$ Gram matrix $G_{\Omega_k} = A_{\Omega_k}A_{\Omega_k}^*$ obtained from the columns of $A$ indexed by an arbitrary set of indices $\Omega_k$ with $|\Omega_k| = k$, one can directly obtain the restricted isometry constant for Eq. 6. More specifically, if the columns of $A$ are of unit norm, one can show that $A$ satisfies the RIP of order $k$ with $\delta_k$ given by:

$$\delta_k = (k - 1) \mu(A), \forall k < 1/\mu(A)$$ \hspace{1cm} (9)

B. Spherical Codes and Optimal Coherence

1) Minimum Coherence: If the goal is to obtain a set of vectors with optimal, i.e., lowest, coherence, as defined above, then a fundamental question is what is the value of this minimum coherence. Several lower bounds on the coherence exist in the literature that depend only on the problem dimensionality, that is, on the number of vectors $n$ and their dimensionality $m$. Obviously, for $n \leq m$ one can select the $n$ vectors from an $m$-dimensional orthonormal
basis, thus attaining zero coherence. Lower bounds are only meaningful for $n > m$. A well-known lower bound for the coherence of $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the Welch bound [59], named after L. Welch, also known as simplex bound, and given by

$$\mu(\mathbf{A}) \geq \sqrt{\frac{n-m}{m(n-1)}}$$

(10)

The Welch bound can also be considered to have been implicitly given in [60] for the case of real vectors, despite the bound in Theorem 5 of [60] was originally an upper bound for mean square value of the sine of the angle between vectors in the set. When rewritten in terms of cosine it would yield a lower bound on the mean square coherence, while the coherence defined in Eq. 7 is rather a maximum coherence. In principle one would like to fulfill Eq. 10 with equality or as close as possible to equality. Two different criteria exist to determine whether the Welch bound is satisfied with equality. One possibility considers the root mean square (RMS) value of the inner product between vectors in the set as coherence and names the set as Welch Bound Equality (WBE) sequence [62]. Alternatively, if one adopts the more strict definition of coherence given by Eq. 7 sets of vectors satisfying Eq. 10 with equality receive the name of Maximum Welch Bound Equality (MWBE) sequences [62]. In this work and for coherence with CS literature we restrict our attention to the maximum coherence defined in Eq. 7 and, therefore, we focus on finding MWBE sequences. Provided that the coherence defined in Eq. 7 has to be necessarily greater than or equal to the RMS counterpart, MWBE sequences are a subset of WBE sequences. The validity of the Welch bound is restricted to $n$ not too large with respect to $m$. More specifically, the following necessary conditions can be derived from the absolute bounds for $A$-sets given in [63] (second and third rows of the Table II for the real and complex case, respectively):

$$n \leq \frac{m(m+1)}{2}, \text{ for } \mathbf{A} \in \mathbb{R}^{m \times n}$$

$$n \leq m^2, \text{ for } \mathbf{A} \in \mathbb{C}^{m \times n}$$

(11)

If $n$ is greater than the upper bounds in Eq. 11 the $n$ columns of $\mathbf{A}$ cannot form an equiangular system [64] and Eq. 10 no longer applies. In such cases, one can adopt the orthoplex bound (initially stated in [17]) in terms of chordal distance for the case of real-valued vectors and extended later to the complex case [65], given by:

$$\mu(\mathbf{A}) \geq \frac{1}{\sqrt{m}}$$

(12)

Similarly to the Welch bound, the orthoplex bound can only be achieved for some values of $n$ that depend on $m$, namely if the following bounds hold:

$$\frac{m(m+1)}{2} < n \leq (m+1)(m + 2), \text{ for } \mathbf{A} \in \mathbb{R}^{m \times n}$$

$$m^2 < n \leq 2(m^2 - 1), \text{ for } \mathbf{A} \in \mathbb{C}^{m \times n}$$

(13)

For cases of very large $n$, when Eq. 13 cannot be fulfilled, one can still adopt the lower bound developed by Kabatiansky and Levenshtein [66], [67] for $n \rightarrow \infty$:

$$\mu(\mathbf{A}) \geq \left\{ \begin{array}{ll}
\sqrt{\frac{n-m}{m(n-1)}}, & \text{for } n \leq m^2 \\
\frac{1}{\sqrt{m}} \sqrt{\frac{2n-m^2-m}{(m+1)(n-m)}}, & \text{for } m^2 < n \leq 2(m^2 - 1) \\
\max\left\{ \sqrt{\frac{2n-m^2-m}{(m+1)(n-m)}}, 1 - 2n^{-\frac{1}{m-1}} \right\}, & \text{for } n > 2(m^2 - 1) \\
\end{array} \right.$$  

$$\forall \mathbf{A} \in \mathbb{C}^{m \times n}$$

(14)

Alternatively, for large $n$ one can also make use of the lower bound given in Eq. 15 originally derived in [68] for the case of low-coherence beamformer codebooks and explicitly presented as a lower bound on the coherence in [22].

$$\mu(\mathbf{A}) \geq 1 - 2n^{-\frac{1}{m-1}}$$

(15)

Taken all previous theoretical bounds into consideration, one can formulate a composite lower bound on the coherence [20], which for the complex case reads:

Note that the given bounds on coherence often derive from more general upper bounds on the minimum of some distance (be geodesic, chordal, etc.) between subspaces packed in some Grassmann manifold. The relatively simple expressions we provide for the bounds are for the case of packing subspaces of unit dimensionality (lines) and thus do not show dependency on this parameter.

2) Best Spherical Codes: In this work we deal with a particular case of the Grassmannian subspace packing problem, namely, finding the best packing of $n$ one-dimensional subspaces in $\mathbb{R}^m$ or $\mathbb{C}^m$. This is known as the Grassmannian line packing problem, where the corresponding Grassmannians are denoted by $G(\mathbb{R}^m, 1)$ and $G(\mathbb{C}^m, 1)$, respectively, often simplified to $G(m, 1)$ in the real case. It is known that in this case the metrics used to determine the best packing (e.g., geodesic, chordal) are equivalent [17] and thus its selection does not influence the result. Solving this problem is equivalent to solving the optimization in Eq. 8 and yields the desired MWBE sequences.

Without loss of generality, we restrict our attention to codes with normalized codewords, in other words, to points on the surface of the unit sphere. Thanks to the normalization factors in the definition of coherence (Eq. 7), arbitrary scaling of the vectors has no effect on the coherence. Already in Section 1 it was introduced that a Spherical Code (SC) refers to a code whose codewords lie on the surface of the unit sphere. We deal with the general case of codewords with complex entries and will drop the reference to the full space $\mathbb{C}^m$ out of the notation. Thus a SC of $n$ codewords in $\mathbb{C}^m$ is denoted by $C_S(m, n)$. An SC is called Best Spherical Code (BSC) and denoted

$$\mu(\mathbf{A}) \geq \left\{ \begin{array}{ll}
\sqrt{\frac{n-m}{m(n-1)}}, & \text{for } n \leq m^2 \\
\frac{1}{\sqrt{m}} \sqrt{\frac{2n-m^2-m}{(m+1)(n-m)}}, & \text{for } m^2 < n \leq 2(m^2 - 1) \\
\max\left\{ \sqrt{\frac{2n-m^2-m}{(m+1)(n-m)}}, 1 - 2n^{-\frac{1}{m-1}} \right\}, & \text{for } n > 2(m^2 - 1) \\
\end{array} \right.$$  

$$\forall \mathbf{A} \in \mathbb{C}^{m \times n}$$

(16)
A mapping of the form $\alpha_{\text{ERSW}}$, which expresses the equilibrium of mutual forces called Equilibrium of Rescaled Differences of (code) Words $\text{BSC}$ into itself. The first and most intuitive form is the so-called Equilibrium of Rescaled Differences of (code) Words (ERDW), for instance. The solution can be expressed by the method of Lagrange multipliers. We omit here the $\parallel \text{BSC} \parallel$ already pointed out in [41] and yields the mapping

$$F_{\alpha} \left[ C_{S} (m, n) \right] = \left\{ \tilde{c}_{u} + \alpha \tilde{f}_{u} \right\}_{u=1}^{n}$$

which in the case of real-valued coefficients yields:

$$C_{S} (m, n)^{(k+1)} = F_{\alpha} \left( C_{S} (m, n)^{(k)} \right), \quad k \geq 0$$

Due to the antipodal codewords, a BASC has a number of codewords that is actually twice the number of columns of the desired optimal measurement matrix, i.e., $2n$. Nevertheless, half of the codewords are given by the other half and, additionally, if the starting point is an antipodal SC, the antipodal symmetry of Eq. [23] ensures that antipodality will be preserved through iterations. In other words, the underlying number of degrees of freedom of the problem remains $n$ (still, in an
algorithm is an iterative integration procedure in which the integration region is divided into subintervals and at each iteration the subinterval responsible for the largest integration error is bisected, so that the error gets reduced. While this procedure yields accurate approximations, it might become too expensive in computational terms. For this reason we also focus here on the approximate summations, since that in the real-valued case, so that the composite lower bound in Eq. 16 (lowest) lower bounds get extended to greater values of $\nu$ and not $Kn$ or $\infty$, for the same reason that in the real-valued ASCs we wrote $n$ and not $2n$, despite the antipodals. Using Eq. 24 (or Eq. 25 for sufficiently large $K$) in the mapping of Eq. 21 yields a CASC that arbitrarily approaches the BCASC as $\nu \to \infty$. Considering the complex antipodals enables that the resulting BCASC $C_{BCAS}(m, n)$ can be directly considered as the optimal CS measurement matrix $A \in \mathbb{C}^{m \times n}$ in terms of coherence. In general the BCASC will not be attained exactly, but closely approximated, and thus we obtain a close-to-optimal solution rather than the absolute optimum.

**C. Approximate Nearest Neighbors**

Any of the formulas for calculating the forces acting over each codeword of the different SCs introduced in Section III-B exhibit the same general form: a summation whose terms have a distance between two $m$-dimensional vectors to the power of $\nu$ as denominator, with $\nu$ typically large. Regardless of any underlying physical meaning of the calculus, it becomes clear that such a denominator makes summands with low distances between vectors have a much more relevant contribution to the sum than summands with larger distances, whose contribution might be negligible. This observation relates to the core of our approximate method for constructing BCASCs and will be further developed in Section IV. For now it only important to realize the interest of an approach for finding the vectors in some given vector set, say $V : |V| = n$, $V \subset \mathbb{R}^m$, that are within a certain neighborhood of some query vector $\vec{v}_q \in \mathbb{R}^m$, where $m$ can be relatively large and $n > m$.

The problem of searching the Nearest Neighbors, from now on NN, to a given query point in high-dimensional spaces appears often in computer vision problems, most specifically
when trying to match visual features between images or between an image and some database of features. Visual features are, in general, points or regions of an image that are considered to be easily recognizable or descriptive of the image content. Feature matching requires computing a vector that describes each feature, known as its descriptor, and the problem reduces to finding the best match (or some candidate set of low cardinality) for it. Typical descriptor dimensionalities are \( m = 128 \) (as originally considered for SIFT [71]) or even larger (as studied for SURF [72]). When matching against feature descriptors from another image, the number of candidates, say \( n \), can reach hundreds and easily get over a million when matching against large databases or feature maps.

This minimalistic reference to the feature matching problem is not just intended to highlight the necessity of efficient methods for performing NN searches in high-dimensional spaces, but also to clarify the connection to our case of study. State-of-the-art methods for generating BCASCs [20] cannot cope with large values of \( m \) and \( n \), due to the square dependency of the complexity on the dimensionality. Similarly, regarding feature matching, it is turns out that there is often no algorithm available that is able to find the nearest neighbor with a computational complexity that is lower than a linear search. Unfortunately, the cost of a linear search often becomes excessive for large values of \( n \) and \( m \), especially if real-time requirements are to be fulfilled, as it is the case in many computer vision applications. In this context algorithms for performing approximate NN (ANN) searches arise as a feasible alternative to performing an exact NN search. When properly tuned these methods are able to speed up the NN searches several orders of magnitude with respect to a linear search. There has been a large amount of research on algorithms for performing ANN searches, a review thereof falls out of the scope of this paper. A common objective is avoiding the linear dependency of the algorithm complexity on \( n \) [73], e.g., aiming for a logarithmic dependency. The parameter \( \varepsilon \) is referred to [74-76] for comparisons between different ANN search approaches.

For completeness, finding the \( K \) NN of a query point \( \vec{v}_q \in \mathbb{R}^m \) in the set of vectors \( V \subset \mathbb{R}^m \) means finding the subset \( V_{NN} \subset V \) with \( |V_{NN}| = K \) satisfying:

\[
d(\vec{v}_q, \vec{v}_q) \leq d(\vec{v}_q, \vec{v}_u), \forall \vec{v}_u \in V_{NN}, \vec{v}_q \in V \setminus V_{NN}
\]  

where \( d(\cdot, \cdot) \) denotes some distance function between two points. In this work we restrict our attention to the Euclidean distance, but considering other distances might be a subject of future work. If we are interested in retrieving \( K \) NN, we implicitly allow that the obtained set of neighbors, say \( V_{ANN} \) with \( |V_{ANN}| = K \), is not exactly equivalent to the set of exact NN \( V_{NN} \). An appropriate measure of the quality of the approximation would be the ratio between the cardinality of the intersection between \( V_{ANN} \) and \( V_{NN} \) and the cardinality of these sets, namely \( K \). This can be used to define \( V_{ANN} \) by means of an lower bound on it, which would read:

\[
\frac{|V_{ANN} \cap V_{NN}|}{K} \geq \varepsilon
\]  

where \( 0 < \varepsilon \leq 1 \) is a custom parameter that sets the approximation error we can tolerate, typically close to 1. Once some approximation parameter is set, the challenge is to design a search structure that allows finding a subset \( V_{ANN} \) satisfying Eq. 28 as fast as possible. Provided that the NN are defined in terms of distances to the query point, good search structures will be those clustering the points in \( V \) using the same metric \( d(\cdot) \). The most widely-used NN search structure is the \( k \)-d tree [77], which is able to obtain search times of \( O(\log n) \), being the computational cost of the tree construction \( O(n \log n) \). The \( k \)-d tree construction was modified in [78] to allow for ANN searches. If \( K = 1 \) (or, in general, low), i.e., if only a single NN is required, the bound in Eq. 28 is meaningless and the \( \varepsilon \)-ANN of \( \vec{v}_q \) is given by the point \( \vec{v}_{ANN} \) satisfying:

\[
d(\vec{v}_{ANN}, \vec{v}_q) \leq (1 + \varepsilon)d(\vec{v}_{NN}, \vec{v}_q)
\]  

where \( \vec{v}_{NN} \) denotes the true NN of \( \vec{v}_q \) and \( \varepsilon \) should be close to zero. The corresponding ANN search locates first the leaf cell containing \( \vec{v}_q \) (in \( O(\log n) \) time) and then the leaf cells are visited in increasing order of distance from \( \vec{v}_q \). The parameter \( \varepsilon \) allows interrupting the search when the distance between \( \vec{v}_q \) and the current leaf cell exceeds \( d(\vec{v}_{ANN}, \vec{v}_q)/(1 + \varepsilon) \), where \( \vec{v}_{ANN} \) denotes the closest point found so far, and \( \vec{v}_{ANN} = \vec{v}_{NN}^* \) is delivered as solution. A similar modification of the \( k \)-d tree for ANN search is described in [79], where the stopping criterion is given by the maximum number of leaf nodes to examine, which yields better performance than the upper bound on the distance error used in [78].

When the dimensionality \( m \) is large, a \( k \)-d tree requires an excessive number of points \( (n = 2^m) \) in order to have splitting nodes covering every dimension. For lower \( n \) the performance of the tree may be far from optimal. As an alternative, multiple randomized \( k \)-d trees can be created from different rotations of the data [80, 81] or, alternatively, randomly selecting the dimension along which the data is divided from a set of few dimensions in which the data exhibits high variance. This way, each \( k \)-d tree covers a different set of dimensions in terms of splitting. Additionally, searches on the different trees can be considered as independent [81], thus yielding an exponential decrease of the error with the number of searched nodes.

In [82] the authors propose to construct the search tree by recursive \( k \)-means clustering of the data, where \( k \) denotes the branching factor or number of disjoint clusters to establish at each node of the tree. A similar idea underlies the Geometric Near-neighbor Access Tree (GNAT) proposed in [83], which avoids the multiple distance computations required to determine the mean point of the clusters taking data points as the cluster centers. The authors also propose adapting the branching factor or degree of each node to the number of points it contains, still keeping a constant average degree. A hierarchical \( k \)-means tree similar to that in [82] is proposed in [84].

Other search structures and evolutions of the previous exist, but are not included for brevity. Even restricting our attention to the few classical approaches highlighted above, the truth is that there is no structure that can be considered best in absolute terms, since their performance largely vary with the nature of
the data, its dimensionality \( m \), the number of points in the database \( n \), and the specific values of the algorithm parameters. The nature of the data relates to the underlying dimensionality of the subspace (or union thereof) where the data lives. If the entire dataset is contained into some \( k \)-dimensional subspace with \( k < m \), the performance will be similar to that obtained using \( k \)-dimensional data, rather than \( m \)-dimensional data. Fortunately, a procedure for automatic selection of the optimal algorithm and its parameters is presented in [55].

The authors consider the algorithm itself one more parameter of a generic NN search routine, whose optimal value for some dataset is to be found. At the light an evaluation of different alternatives for constructing the trees, two methods repeatedly showed the best performance, namely, the hierarchical \( k \)-means tree and the randomized \( k \)-d trees. The automatic algorithm selection and tuning procedure only considers these two alternatives and requires from the user the value of two parameter that control the importance of the tree build time relative to the search time and the importance of the memory overhead compared to the time overhead, respectively. This algorithm was made available in the nowadays famous Fast Library for Approximate Nearest Neighbors (FLANN) [86], and is the general-purpose ANN search algorithm we implement in our approach.

### IV. Methodology

The method in [20] was shown to produce BCASCs that closely approach and eventually meet the theoretical lower bounds on coherence. The algorithm is a numerical search whose structure naturally derives from the definition of BCASCs. In spite of approximating the integrals over complex rotations that needs to be evaluated to obtain the forces by discrete summations, the computational load of the method becomes intractable for large numbers of codewords of large dimensionality. This is due to the fact that the complexity of each iteration for a fixed \( \nu \) scales asymptotically with \( \mathcal{O}(m^2n^2K) \), that is, the computational cost grows linearly with the number of discrete complex rotations considered in the approximate integration \( K \), but quadratically with the codeword dimensionality \( m \) and the number of codewords \( n \). The quadratic dependency on \( n \) follows directly from the fact that, for each of the \( n \) codewords, the difference with respect to each of the \( (n-1)K \sim nK \) complex rotations of the other codewords are to be computed. The quadratic dependency on \( m \) is due to implementation details and can be circumvented. For a reference on execution times, the reader is referred to Table 6.3 of [88], where the method introduced in [20] with approximate integration requires over an hour to construct a BCASC for \( n = 64 \) codewords of dimensionality \( m = 4 \). Without implementing the approximation the original method took nearly 18 hours for constructing the same BCASC.

Our goal is to achieve the excellent results in [20], but drastically reducing the execution time that the original algorithm requires for large values of \( m \) and \( n \), so that the range of applications gets broader. In CS it is often the case that, despite the underlying dimensionality of the problem to solve (say, sparsity or, in general, signal complexity) might be low, the ambient dimension might be very large, e.g., in case of a fine discretization of the whole ambient space. In such cases, the method in [20] cannot be directly applied to generate close-to-optimal sensing matrices because of the square dependency on \( n \). Furthermore, the number of measurements \( m \) to attain successful reconstruction depends on \( \log n \) and will be also affect the execution time quadratically. Additionally, it is a known issue of most methods for generating spherical codes or, in general, best packings, that the performance degrades as the number of points to pack increases, while the computational effort grows [34]. In order to extend the applicability of the BCASCs as CS sensing matrices for real-world problems of large \( n \), we need to get rid of the squares in \( m \) and \( n \), so that the complexity of each iteration reduces to \( \mathcal{O}(mnK) \), while still preserving the adequate algorithmic structure presented in [20].

An analysis of the original program for generating BCASCs with approximate summation whose performance is evaluated in [20], [88] reveals that the square dependency of the complexity on \( m \) is implementation-dependent rather than specific to the algorithm. We observe that an optimization of the code easily reduces the square dependency to a linear dependency, thus yielding complexity \( \mathcal{O}(mn^2K) \) per iteration. We would like to stress that this optimized version of the original code does not imply an approximation by any means, since the mathematical operations leading to the construction of the code remain the same and, consequently, the resulting codes are identical up to machine precision. From now on we take this improved implementation as reference and it will be named simply as the BCASC construction algorithm, while any reference to the original implementation will be made explicit.

The square dependency on \( n \) is indeed specific to the algorithm, since all other codewords contribute to generate the force that move the codeword under consideration from one iteration to the next. Therefore, this dependency cannot be softened without incurring an approximation. More specifically, for each codeword one would need to select few other codewords or, better said, complex rotations thereof, that actually contribute to the calculus of the force that actuates over the codeword under consideration, having the rest no effect in this regard. This is the core idea of this work: how to select few (eventually complex-rotated) codewords that are responsible for the greatest contributions in the force calculus step in such a way that the quality of the final BCASC does not degrade with respect to the reference approach, without this approximation. To this end, we propose using an ANN search to decide which summands in the double summation of Eq. 25 are to be included in the force calculus and which ones can be neglected.

In this work we establish the nearest neighbors to some query codeword \( \vec{c}_q \in \mathbb{C}^m \) using the Euclidean distance and use FLANN to retrieve them. Note that we deal with complex-valued vectors, while FLANN is implemented for
the real-valued case. For simplicity, let \( \tilde{c}_{u,v,k} \in \mathbb{C}^m \) denote the difference vector between the codeword \( \tilde{c}_u \) and the \( k \)th complex rotation of some other codeword \( \tilde{c}_v \). If we denote by \( \tilde{c}_{u,v,k}^R = R(\tilde{c}_{u,v,k}) \in \mathbb{R}^m \) the vector of real components and by \( \tilde{c}_{u,v,k} = \Im(\tilde{c}_{u,v,k}) \in \mathbb{R}^m \) the vector of real components, we can pack the initial complex vector \( \tilde{c}_{u,v,k} \in \mathbb{C}^m \) into a real vector of double dimensionality \( \tilde{c}_{u,v,k}^R = \left[ \tilde{c}_{u,v,k}^T ; \tilde{c}_{u,v,k}^T \right]^T \in \mathbb{R}^{2m} \). This is just an operational detail that only affects the way the data is stored, the tree construction and the ANN searches, but not the algorithm itself. Observe that Euclidean distances in this induced \( 2m \)-dimensional real space are equivalent to those in the original \( m \)-dimensional complex space:

\[
\| \tilde{c}_{u,v,k}^R \|_2^2 = \sum_{w=1}^m [\Re(\tilde{c}_{u,v,k,w})]^2 + [\Im(\tilde{c}_{u,v,k,w})]^2 = \| \tilde{c}_{u,v,k} \|_2^2
\]

where \( \tilde{c}_{u,v,k,w} \) denotes the \( w \)th component of \( \tilde{c}_{u,v,k} \). This domain change implies that in practice we construct the FLANN index with vectors in \( \mathbb{R}^{2m} \). The number of points in the index is not \( n \), but \( Kn \) due to the double summation in Eq. (32). The set of all complex rotations of all codewords will be denoted by \( V \). From now on we exclusively denote the number of summands effectively used in the calculus of the forces, that is, \( K = \vert V_{\text{ANN}} \vert \) with \( V_{\text{ANN}} \subseteq V \), and no longer denotes the number of discrete rotations considered when approximating the integral on complex rotation domain by a summation, which we will further denote by \( n_{\text{rot}} \). Then \( |V| = n_{\text{rot}}K = nm_{\text{rot}} \). Note that in \( m_{\text{rot}} \) is defined according to this redefinition, while we will see that \( K \leq n_{\text{rot}} \) in our approach.

For the sake of generality, we consider two possible ways of finding the ANN, namely, by means of a radius search and by means of a KNN search. In the exact search case, the KNN delivers the subset \( V_{\text{NN}} \) of cardinality \( K \) satisfying Eq. (27) in the approximate case, we obtain a set \( V_{\text{ANN}} \) for which the middle term of Eq. (28) might be lower than 1. In such cases one may require that Eq. (29) holds for each of the \( K \) neighbors. A radius search delivers a set \( V_{\text{NN}} \) such that:

\[
\begin{align*}
    d(\tilde{v}_u, \tilde{v}_q) &\leq r, \forall \tilde{v}_q \in V_{\text{NN}} \\
    d(\tilde{v}_u, \tilde{v}_q) &> r, \forall \tilde{v}_q \in V \setminus V_{\text{NN}}
\end{align*}
\]

where \( r \) denotes the radius, i.e., delivers the set of points that are contained in the ball of radius \( r \) centered at the query point. This way the resulting \( 0 \leq K < n_{\text{rot}} \) depends on \( 0 \leq r \leq 2 \). Regardless of whether a radius or a KNN search is performed, one ends up with some set of \( K \) \( \text{NN} V_{\text{ANN}} \) and Eq. (25) boils down to a single \( K \)-term summation:

\[
\left\{ \tilde{f}_u \right\}_u = \sum_{\tilde{c}_u \in V_{\text{ANN}}(\tilde{c}_u)} \frac{\tilde{c}_u - \tilde{v}_q}{\| \tilde{c}_u - \tilde{v}_q \|_2} = \left\{ \frac{\tilde{c}_u - \tilde{v}_q}{\| \tilde{c}_u - \tilde{v}_q \|_2} \right\}_u
\]

where \( V_{\text{ANN}}(\tilde{c}_u) \) simply denotes the set of ANN obtained for the query point \( \tilde{c}_u \), whose cardinality \( K \) may vary with \( u \) if the radius search case and is fixed to \( K_u = K \) in the KNN search case. At this point we are prepared to understand the algorithm for ANN-BCASC construction, which is given in Algorithm 1.

**Algorithm 1 Approximate Nearest Neighbor - Best Complex Antipodal Spherical Code (ANN-BCASC)**

**Initialize:** \( \alpha = \alpha_0 \), \( \nu = \nu_0 \), \( C_S(m, n) = \{ \tilde{c}_u^n \}_{u=1}^{n} \in \mathbb{C}^{m \times n} \)

1. \( \nu < \nu_{\text{max}} \) do

2. **FixedPoint** \( u = false \), \( \forall 1 \leq u \leq n \)

3. \( \tau = 0 \)

4. Initialize damping factors: \( \alpha_n = \alpha, \forall 1 \leq u \leq n \)

5. while \( (\nu < \nu_{\text{max}}) \) and \( \left( \left( \frac{\nu}{\nu_{\text{rot}}} FixedPoint_u = false \right) \right) \)

6. FLANN index: \( \mathcal{I}(V) \), \( V = \{ \tilde{c}_u e_{\text{rot}}^{\nu k} \}_{1 \leq u \leq n} \)

7. Find ANNs in \( \mathcal{I}(V) \): \( V_{\text{ANN}}(\tilde{c}_u) \), \( \forall 1 \leq u \leq n \)

8. for \( u = 1 \) \( \rightarrow u = n \) do

9. Calculate forces: \( \tilde{f}_u = \sum_{\tilde{c}_u \in V_{\text{ANN}}(\tilde{c}_u)} \frac{\tilde{c}_u - \tilde{v}_q}{\| \tilde{c}_u - \tilde{v}_q \|_2} \)

10. if \( \left( \left( \| \tilde{f}_u \|_2 < \varepsilon \| \tilde{f}_u \|_2 \right) \right) \) and \( (2\alpha_u < 1) \) then

11. \( \alpha_u := \frac{\alpha_u}{2} \)

12. else

13. \( \alpha_u := \frac{\alpha_0}{\nu + 1} \)

14. end if

15. Apply forces: \( \tilde{c}_u = \tilde{c}_u + \alpha_u \tilde{f}_u \)

16. if \( \left( \| \tilde{f}_u - \tilde{c}_u \|_2 < \varepsilon \right) \) then

17. \( \text{FixedPoint}_u = true \)

18. end if

19. end for

20. \( \tau := \tau + 1 \)

21. end while

22. Update \( \nu: \nu := 2\nu \)

23. Update \( \alpha: \alpha := \frac{\alpha_0}{\nu + 1} \)

24. end while

**Return:** \( C_{\text{BCASC}}(m, n) = \{ \tilde{c}_u^n \}_{u=1}^{n} \)

The initial value of \( \alpha \) is relatively close to one, e.g., \( \alpha_0 = 0.9 \), so that the codewords are allowed to experience large changes when applying the forces (line 15). Complementary, the initial value of the exponent \( \nu \) has to be low, typically \( \nu_0 = 2 \), which is the minimum feasible value. At the end of each iteration of the outer while loop (line 1), \( \nu \) is doubled (line 22) and \( \alpha \) is conveniently decreased (line 23), so that \( \alpha \rightarrow 0 \) as \( \nu \rightarrow \infty \). Before starting the inner while loop (line 6) the damping factor to be used for each codeword is the same for all codewords and equal to \( \alpha \). This loop is executed until all codewords are considered a fixed point or some maximum number of iterations \( \nu_{\text{max}} \) is reached. Within the loop, first the search structure is created using FLANN. It can be either a randomized \( k \)-d tree or a \( k \)-means tree, depending on the data to index \( V \), and is denoted by \( \mathcal{I}(V) \). In line 9 Eq. (32) is used to compute the approximate force acting on each codeword. We preserve the idea of *accelerating on
straight lines from \([20]\): if the direction of \(\vec{f}_u\) did not change between consecutive iterations, the corresponding damping factor \(\alpha_u\) is increased even within the inner loop (line 10). The threshold \(\varepsilon_\Delta \vec{f}_u\) regulates the sensitivity of the direction change detection. A codeword is considered a fixed point if it does not differ from the force acting over it. In practice, a threshold \(\varepsilon\) on the \(l_2\) norm of the difference is applied (line 16).

If \(\varepsilon\) is sufficiently small and \(\tau_{\max}\) not too restrictive, the set of codewords obtained after execution of Algorithm 1 can be considered a good approximation of the true BCASC.

In Algorithm 1 the initial complex spherical code \(C_S(m, n)\) is randomly generated. Eventual appearance of too close codewords in the initial code can thus occur, yielding an inappropriate seed for the algorithm. One could avoid this situation by implementing some sort of intelligent initialization procedure, e.g., generating candidate complex codewords randomly, but only adding them to the initial code if the distances to the previously selected codewords are greater than a threshold, or, equivalently, if the corresponding scalar products are lower than a threshold, similarly to the initialization in [34]. Both in [20] and in our improvement the forces acting over the codewords tend to zero as the exponent \(\nu\) tends to infinity. It is natural to see here a connection to the cooling process in simulated annealing methods [89], [90]. We would also like to point out a parallelism to the method for optimal frame design in [39], which starts with a loose constraint on the variation between consecutive frames, which gets tightened along iterations, progressively freezing the process.

Interestingly, after designing and evaluating the proposed approach, we found out that the idea of restricting the calculus of the forces to some zone of significant influence around the point under consideration is already contained in [42]. Differently, in [42] the codewords lie in a real space of very low dimensionality, while our approach was conceived to deal with codes living in high-dimensional complex spaces. More specifically, in [42] points are excluded from the force calculus if their influence is weaker than a custom threshold. For a given value of \(\nu\) this restriction immediately translates into a radius constraint and is thus equivalent to an NN radius search. Nevertheless, the equivalent search radius depends on \(\nu\) and, consequently, in the first stages of the algorithm, where \(\nu\) is still very small compared to its final value, the equivalent search radius is very large and most (eventually all) other points are considered in the calculus. This is only motivated by the fact that for small \(\nu\) both close and far points yield comparable influences in terms of forces, while, as \(\nu\) increases, the influence of far points become negligible compared to that of close ones. Nevertheless, the experience demonstrates that this computational overhead is not really necessary. Fixing a relatively small region of influence from the beginning yields better results than even considering all other points, as our results show. This is due to the fact that the crucial question is not whether the influence is negligible or not, but rather whether it is convenient or not. Indeed, for low \(\nu\) the influence of some points might not be numerically negligible, despite not being close points, but their influence can be neglected because it is not a convenient one. Heuristically, we know that only few neighboring points should influence the displacement of the point under consideration at each iteration, while the aggregate influence of all other points can be regarded as some sort of background clutter that should be ignored.

Note that locality, namely, the restriction of the points that will affect a specific codeword to a certain neighborhood of that codeword, is already present in [23], since the Voronoi cells are defined according to the distance to the previously obtained center of mass. Points not belonging to the new cell will not contribute anymore to updating its center of mass. Unfortunately, integrating probability distributions on high-dimensional complex spaces over Voronoi cells defined by multiple (also highly-dimensional) hyperplanes is not a trivial task. Differently, we do not work with the underlying uniform probability distribution on the complex unit sphere whose optimal quantizer codebook is expected to be an excellent MWBE. We work only with the temporal solutions to such quantizers, that is, with the temporal BCASCs, which (progressively) distill the aforementioned underlying distribution by acquiring an even distribution themselves when regarded as particles. Note that this formulation brings locality back into the game, but without the massive growth of computational cost that the integration of an spherically-supported probabilistic distribution over Voronoi cells would require. Note that tree structures are the natural search structures for efficiently implementing locality in the way we do.

Regarding the complexity of each inner iteration of Algorithm 1, note that an ANN search is to be carried out for each of the \(n\) codewords (line 7). Provided that each search in the tree has a reduced complexity \(O(\log nn_{rot})\), this yields a total search complexity of \(O(n \log nn_{rot})\). Despite this varies from one search structure to another, the complexity of building the tree \(\mathcal{T}\) from the data (line 9) can be considered to be \(O(mn_{rot} \log (nn_{rot}))\). The rest of the algorithm has complexity \(O(nK)\) or, more generally, \(O\left(\sum_{u=1}^{n} |V_{ANN}(\vec{c}_u)|\right)\), due to the force calculus (line 9). Note that there is no dependency on \(m\) because all necessary distances have been already computed in the ANN search. In practice a very small \(K \ll n\) is selected and the total complexity of each iteration of the innermost loop of the algorithm can be considered to be \(O(mn_{rot} \log (nn_{rot}))\) for some specific values of \(\nu\) and \(K\), regardless of \(K\).

V. EXPERIMENTAL EVALUATION

In this section we present the results of different experiments destined to evaluate the performance of the ANN-BCASC algorithm. The experiments are divided in three groups. In Section V-A we compare both the performance of our approach to that of the reference algorithm it derives from, namely [20], both in terms of execution time and coherence of the resulting BCASCs. In Section V-B the scope is widened in order to offer a general comparison to other approaches for close-to-optimal complex code construction. Finally, Section V-C evaluates the performance of our BCASCs for recovering sparse signals from few measurements in a typical CS framework.
A. Comparison to Reference BCASC Algorithm

The main advantage of the ANN approximation over the reference BCASC algorithm is the fact that the calculus of the forces acting over each codewords requires some low number of summands, say $K$, rather than $n_{\text{rot}}$, which will be generally large. A fundamental question is how small can $K$ be while still obtaining complex codes with excellent coherence properties. In the case of performing radius searches for determining the ANN, the same question is transferable to the considered radius $r$. The first series of experiments is intended to provide an empirical answer to these questions and studies the evolution of the coherence and the execution time both with $K$ in the case of KNN search and with $r$ in the case of radius search. We aim to generate $n = 64$ codewords of dimensionality $m = 8$ considering $n_{\text{rot}} = 16$ complex rotations. For this case we use a single $k$-d tree as search structure, without imposing any restriction on the maximum number of leaves to visit. This way the exact NNs can be found. The higher computational cost of exact searches gets widely compensated by the lower computational cost of using a single $k$-d tree in this case.

The results of these experiment series are collected in Fig. 1. The plots at the left hand side show the evolution of the time required to generate the codes, both with $r$ in the case of radius search (Fig. 1a) and with $K$ in the case of KNN search (Fig. 1c). Similarly, the plots at the right hand side show the evolution of the coherence of the obtained codes, also with $r$ (in Fig. 1b) and with $K$ (in Fig. 1d). In the radius search experiments, we consider radii $r > 1.14$, up to the maximum $r = 2$, since our approach failed to generate BCASCs for $r < 1.14$. In the KNN experiments we considered all $0 < K \leq n_{\text{rot}}$. In both cases 32 experimental cases are considered. We compare the result of our ANN-based approach to the reference algorithm from [20]. Additionally, we include a complex random matrix and a Fourier ensemble in the coherence evaluation. Both the real and complex components of the elements of the random matrix are drawn from a Gaussian distribution of zero mean and unit standard deviation. The Fourier ensemble of size $m \times n$ is constructed by randomly selecting $m$ rows of a DFT matrix on size $n \times n$. Both matrices are then columnwise normalized. In order to assess the stability of our construction method and obtain some statistically relevant data, we repeat all the experiments ten times. The solid lines in the plots of Fig. 1 are thus average results. The shaded area of the same color around each line depicts the range in which all the results of the individual experiments fall. When the shaded area is not visible, it is because the line width used to plot the corresponding mean values is wider than the width of the range. This is particularly the case for the execution times of the proposed approach, which exhibit negligible variations between executions, and the coherences of the BCASC obtained both via our approach and the reference algorithm.

The first conclusion that can be drawn from Fig. 1 is that, regardless of the search method and the number of NN delivered by the ANN searches, no degradation of the coherence of the generated BCASCs is observed, when comparing to the reference algorithm. Comparing Fig. 1a to Fig. 1c one may think that performing KNN searches is more efficient than radius searches, but the plain region of the curve in Fig. 1a suggests that for radius in that range the NN are all points in the index, thus further increasing the radius search has no computational cost. Consequently, one should compare only the initial region of the curve in Fig. 1d to that in Fig. 1c in which both search options perform equivalently, showing a linear dependency of the execution time on the size of the neighborhood considered. From Fig. 1b and Fig. 1d it becomes clear that BCASCs are superior in terms of coherence, not just to randomly-generated SCs, but to SCs derived from a discrete Fourier basis. In fact, it is remarkable how close BCASC coherences are to the theoretical coherence, given in this case by Eq. 10 regardless of the construction method. The mean coherence of BCASCs generated using our approach is very close to that obtained using the reference algorithm, still always lower than the latter for all experimental cases. In short terms, the ANN-BCASC approximation is able to simultaneously provide a dramatic reduction of the execution time and a slight coherence reduction.

From now on we adopt the KNN search as default search method for our experiments. A closer observation of the ANN-BCASC coherences in Fig. 1d reveals that the minimal coherence is obtained for the lowest $K$, while reaching that obtained using the reference algorithm for $K = n_{\text{rot}}$. This comes at no surprise, since for $K = n_{\text{rot}}$ our approximation is equivalent to the reference. The remaining question is then how much can we push the ANN approximation so that the execution time and eventually the coherence are further reduced, while still being able to successfully generate the codes. In the search of an answer we carried out detail experiments for very low $K$ and unit increments, which showed that successful ANN-BCASC generation was possible with as few as $K = 8$ NN. Fig. 2 shows the execution times and coherences obtained in 32 different experimental cases, from the lowest possible number of NN, $K = 8$, to $K = 39$, both included.

The results in Fig. 2 are highly encouraging. As expected, the execution time exhibits a slow linear increase with $K$, while the coherence still remains slightly below that of the reference algorithm for all cases considered. For instance, adopting the minimal $K$, the execution time is approximately 50 s, instead of the $> 400$ s required by the reference algorithm, i.e., the execution time was reduced by a factor of $> 8$. In terms of coherence, the minimum average coherence, obtained for $K = 20$, was 0.3272, which means a 1% reduction when compared to that obtained with the reference algorithm (0.3763). While this might look negligible at first sight, if differences with respect to the theoretical lower bound are considered, this means a reduction of more than 8%.

Provided the large potential for reducing execution time, one can aim to establish some custom tradeoff between this and an improvement on the quality of the codes, for instance, by considering a larger number of complex rotations in the calculus, i.e., refining the approximate integral. Then it is of interest to study the evolution of the execution time with $n_{\text{rot}}$ and the effect of considering larger $n_{\text{rot}}$ on the coherence
of the resulting codes. We do so for $16 < n_{\text{rot}} \leq 48$ with unit step size, provided that the case $n_{\text{rot}} = 16$ was already studied before. The size of the codes remain $8 \times 64$ and the number of NNs to search for is set to $K = 20$ in this and all the following experiments. Both the execution time and coherence are compared to those obtained with the reference algorithm. As before, all experiments are repeated ten times and the results are given in Fig. 3. Due to the large execution times of the reference algorithm, the cases $n_{\text{rot}} > 32$ were omitted and estimated by extrapolation of a polynomial fit.

Fig. 3a is truly illuminating: even for the largest number of rotations considered ($n_{\text{rot}} = 48$), the execution time required by our approach is three times lower than that required by the reference for $n_{\text{rot}} = 16$. Furthermore, the slope of the polynomial fit of the mean values of execution times is much lower for the ANN approximate method, more specifically, 2.450 s/rot versus 23.11 s/rot, i.e., an order of magnitude lower. Extrapolating the linear fit for our approach one obtains that with the same time budget required by the reference algorithm for only $n_{\text{rot}} = 16$, up to $n_{\text{rot}} = 150$ complex rotations could be considered in the calculus. Also enlightening is Fig. 3b, since it shows that increasing $n_{\text{rot}}$ has in our approach the same effect of further coherence reduction already observed in [20] for the reference algorithm. Furthermore, the shaded areas showing the ranges where the results of each approach live are typically disjoint, that is, in terms of coherence the best code delivered by the reference approach is worse than the worst code delivered by our approach, regardless of the experimental case considered. The mean difference between the quadratic polynomial fits in the considered range is $1.965 \times 10^{-3}$, which means over a 4% reduction in terms of mean differences with respect to the theoretical lower bound. Also important is to notice that the rate at which the mean coherence decreases also decreases with $n_{\text{rot}}$, up to the point that for $n_{\text{rot}} > 32$ the fitted average coherence starts to degrade and further increase of $n_{\text{rot}}$ will only worsen it. This effect was not observed for the reference algorithm, neither in [20] nor in this work and, in fact,
Fig. 2: Detailed evaluation of the execution time (a) needed to generate the BCASCs by means of our ANN approximation for different numbers of NNs and corresponding coherence of the generated codes (b). The execution time is compared to that of the reference algorithm and the coherences of both BCASCs and ANN-BCASCs are compared to the theoretical lower bound and to the coherences of complex random matrices and Fourier ensembles. The solid lines depict the mean of ten experiment runs for each experimental case and the shaded areas are defined between the minimum and maximum values obtained for each case.

Fig. 3: Evolution of the execution time (a) needed to generate the ANN-BCASCs for different numbers of complex rotations $n_{\text{rot}}$ considered in the algorithm and corresponding coherence of the obtained codes (b). The execution time is compared to that of the reference algorithm and the coherences of both BCASCs and ANN-BCASCs are compared to the theoretical lower bound. The solid lines depict the mean of ten experiment runs for each experimental case and the shaded areas are defined between the minimum and maximum values obtained for each case.

Experiments for $n_{\text{rot}} > 32$ revealed an asymptotic decrease of the coherence with $n_{\text{rot}}$ for the reference algorithm. The worsening of the coherence for very large $n_{\text{rot}}$ in our approach is due to the relatively low number of NN per search, set to $K = 20$ for all experiments to obtain ceteris paribus results. Increasing $K$ linearly with $n_{\text{rot}}$ would solve this issue, at the cost of an slightly increased execution time.

Alternatively, for some fixed $n_{\text{rot}}$, considered to be sufficiently high, the time saving that the ANN variant provides can be invested in generating larger codes, i.e., of larger dimensionality and with larger number of codewords. This was indeed one of the main motivations of this work, provided that in CS the measurement matrices can be very wide, due to the fact that $n$ often relates to the step used to discretize some continuous domain and not to the underlying dimensionality of the problem. In order to obtain empirical evidence of the expectedly good behavior of the proposed approach with increasing $m$ and $n$, we carry out independent experiments to check the evolution of the execution time and the coherence with these parameters. For the $m$-sweep experiments we fix $n = 64$ and consider 32 experimental cases for $2 \leq m \leq 64$. Clearly, for $m = n = 64$ an orthobasis is obtained and zero coherence is attained. For the $n$-sweep experiments we fix $m = 8$ and would like to consider the cases from $n = m = 8$ to $n = 128$. Unfortunately, due to the quadratic dependency
of the execution time on $n$ in the reference algorithm, we only consider 32 experimental cases for $8 \leq n \leq 101$ (step 3) and obtain the rest cases up to $n = 128$ by extrapolation of a polynomial fit. As before, all experiments are repeated ten times to obtain mean, minimum and maximum values. It turned out that we could neither afford the computational cost required for running all the $m$-sweep experiments with the costly reference algorithm. For this reason we were forced to run only the first 12 experimental cases. We used appropriate polynomial fits to offer an approximation to the results of the omitted cases. The results of these experiments are summarized in Fig. 4. As before, shadowed areas are used to represent the ranges between minimum and maximum values and solid lines are used for mean values.

The first row of plots in Fig. 4 correspond to the varying-$m$ experiments. Similarly to Fig. 3a, Fig. 4a also reveals the potential of the ANN approximation for obtaining much more from the same or lower computational effort. Indeed, it is clear than even for the maximum $m$ considered our approach is able to deliver a code in a time that is lower than the time the reference algorithm needs to generate the code of lowest $m$. The linear evolution of the execution time of the reference approach with $m$ is due to the fact we use our own optimized (yet exact) reimplementation, but was originally quadratic. Note the accuracy of the linear fits for both approaches and observe the large ratio between their slopes. The slope of the reference algorithm linear fit is ten times greater than that of our approach. This means that our approach is suitable for generating BCASCs of large $m$, while the reference soon requires unrealistically high execution times. Further individual experiments revealed that for $m > 54$, i.e., for $m \rightarrow n$ the reference algorithm is able to find fixed points in a reduced amount of time, abruptly breaking the linearity of the regression in Fig. 4a. This is due to the fact that it becomes easier to optimally pack the $n$ codewords in an hypersphere of such a high dimensionality. Unfortunately, the case $m \rightarrow n$ is of no interest in CS. Regarding the coherence of the codes, Fig. 4b shows how tightly BCASCs approach the theoretical lower bound, regardless of the approach used to generate them. A detailed view reveals that the line corresponding to our approach is always below the one corresponding to the reference algorithm, as one could already expect from previous experiments. When $m < n$ the coherence globally approaches to one, while tending to zero as $m \rightarrow n$. Despite the coherence of the Fourier ensemble also follows this pattern and yields an orthobasis for $m = n$, it is far from the lower bound for relatively low $m$, which is the case of interest in CS. The random construction of complex SCs delivers the worst results and is unable to push the coherence under 0.36. Also remarkable is the stability of both our approach and the reference one, both in terms of coherence and execution time.

The second row of plots in Fig. 4 summarizes the results of the varying-$n$ experiments. Fig. 4c confirms both the quadratic dependency of the execution time of the reference algorithm on $n$ and the linear dependency of our approach. Compare the relatively fast increase of the slope of the quadratic fit for the reference algorithm with the low constant slope of the linear fit for our approach. As a consequence, the accurate polynomial fits predict that for a moderate $n = 128$ our approach only needs around 2 min to generate the code, i.e., one order of magnitude less than the 26 min required by the reference algorithm. Fig. 4d shows that the massive speedup does not degrade the coherence and, in fact, one can observe that the points for the mean coherence obtained with our approach are always below those corresponding to the reference approach. Regardless of the method, BCASC approach the theoretical lower bound more tightly than Fourier ensembles or random complex SCs, whose coherence tends to one as $n \rightarrow \infty$. Also in this experiment series the stability of both BCASC generation methods is observed to be very high compared to the other alternatives.

B. General Comparison to Related Work on BCASC Construction

In this section we provide a general comparison between the ANN-BCASC approximation and other approaches for generating close-to-optimal complex codes. In order to provide a complete cumulative comparison, the best option for such comparison is to extend the tables in [20], which are in turn an extension of Table II of [22] and Table II of [30], respectively. These tables provide the values of coherence obtained from complex SCs generated using different methods and are named Table I and Table II homonomously to the corresponding tables in [20]. For both the BCASC and the ANN-BCASC methods we adopt the best result of ten independent runs. We also measure the time that each algorithm needed to generate the codes. For the other algorithms, we preserve the values in the original tables of [20]. We use the reference BCASC algorithm with approximate summation with $n_{\text{rot}} = 32$ summands, in order to have a fair reference for the ANN-BCASC algorithm, for which we adopt the same $n_{\text{rot}}$ and a constant $K = 20$ for all cases considered. As in [20], for both methods $\alpha_0 = 0.9$ and $\tau_{\text{max}} = 10^9$ were used.

In short terms, the proposed ANN approximate approach and the reference algorithm generate BCASC of equivalent quality in terms of coherence. In other words, Tables I and II confirm that the reduction of the algorithm complexity has no significant effect on the quality of the obtained BCASC. In fact, both alternatives often produce codes with equal coherence, closely approaching and eventually meeting the theoretical lower bound. Cases for which our approach meets the theoretical lower bound are $m = 4$, $n = 5$ and $m = 5$, $n = 6$. Furthermore, in some cases the proposed approach outperforms the reference method, which is a remarkable fact, even when the differences are rather negligible in general. The reader might observe that the coherences obtained for the reference method often do not coincide with those given in [20]. This is due to the fact that we use the variant with approximate discrete summation for calculating the integral over complex rotations, in order to enable a fair comparison to the proposed approach, while the results presented in Tables I and II of [20] were obtained using numerical integration.

Regarding the execution time, Table II suggests that the proposed approach provides a moderate speedup when compared to the reference, but not even in every case. Furthermore,
The execution time is compared to that of the reference algorithm and the coherences of both BCASCs and ANN-BCASCs are compared to the theoretical lower bound and to the coherences of complex random matrices and Fourier ensembles of the same size. The solid lines depict the mean of ten experiment runs for each experimental case and the shaded areas define between the minimum and maximum values obtained for each case.

**Fig. 4:** Evaluation of the execution time (left plots) needed to generate the ANN-BCASCs for different code sizes \( m \times n \) and corresponding code coherences (right plots). Plots in the first row show the results for varying \( m \) and in the second row for varying \( n \). The execution time is given in seconds (s).

**Table I:** Comparison of the coherence of close-to-optimal complex codes obtained via different numerical approaches. Based on Table I of [20].

| \( m \) | \( n \) | Composite Bound | ANN-BCASC | BCASC [20] | Medra et al. [20] | Xia et al. [22] | Love [21] | Execution Time (s) |
|-------|-------|-----------------|-----------|------------|------------------|----------------|-----------|-------------------|
| 2     | 8     | 0.7500          | 0.7953    | 0.7954     | 0.7997           | 0.8216         | 0.8415    | \( 1.768 \times 10^2 \) |
| 3     | 16    | 0.6202          | 0.6490    | 0.6498     | 0.6590           | 0.6766         | 0.8079    | \( 5.839 \times 10^2 \) |
| 4     | 16    | 0.4472          | 0.4486    | 0.4480     | 0.4473           | 0.4514         | 0.7525    | \( 7.253 \times 10^2 \) |
| 4     | 64    | 0.6000          | 0.6878    | 0.6878     | 0.7151           | 0.7447         | 0.7973    | \( 4.859 \times 10^3 \) |

Table II shows lower execution times for the reference algorithm in all cases considered. At first sight this seems to be in conflict with the reduction of the computational complexity of each algorithm iteration that our approach was claimed to bring. Actually there is no such conflict and the (often largely) reduced execution times obtained for the reference algorithm are due to the fact that fixed points are often found within the first iterations, while the proposed method does not find fixed points so easily due to its locally-restricted operation. Another reason is that we use our own optimized implementation of the reference algorithm, which already makes linear the quadratic dependency of the complexity on \( m \). In practice the early discovery of fixed points and thus abnormally low execution times only happen when \( n \approx m \), being both relatively low, which is the case for the experimental cases considered in Tables I and II. Such cases are of no relevance in practical CS scenarios, in which \( n \gg m \) can be very large. In such cases the reference algorithm is of no use due to the quadratic dependency of its complexity on \( n \) (and originally also on \( m \)) and the proposed method arises as a feasible alternative without coherence degradation.
C. Compressive Sensing Performance

In this last section we come back to the main motivation of the work, namely, the construction of close-to-optimal complex CS measurement matrices. The final measure of the success or failure of the proposed approach is to be measured in terms of the capability of the resulting BCASCs to recover high-dimensional sparse signals from a relatively low number of measurements. We are interested in the cases where \( n \) is so large that generating the BCASCs using the reference algorithm becomes unfeasible. In the experiments described in this section the dimensionality of the sparse signal to recover is set to \( n \in \{128, 256, 512\} \). Once \( A \in \mathbb{C}^{m \times n} \), namely, the approximate BCASC is generated, the goal is to recover the \( s \)-sparse \( n \)-dimensional signal \( \tilde{x} \) from the \( m \ll n \) linear measurements given by \( \tilde{y} = A \tilde{x} \). An estimate of the signal \( \hat{x} \) is recovered from \( \tilde{y} \) by \( l_1 \)-minimization in a conventional CS framework using the efficient Chambolle and Pock’s algorithm \[92\]. The normalized \( l_2 \) distance between \( \tilde{x} \) and \( \hat{x} \) is adopted as measure of the recovery error. The real and imaginary components of the nonzero entries of \( \tilde{x} \) are drawn from a Gaussian distribution of zero mean and unit standard deviation and \( \tilde{x} \) is then \( l_2 \)-normalized. The recovery performance obtained using ANN-BCASC measurement matrices is compared to that obtained using both random complex matrices and Fourier ensembles of the same size. In all cases the columns of the matrices are normalized.

We consider different experimental cases for different values of the parameters \( \delta = m/n \) and \( \rho = s/m \), as suggested in \[93\]. More specifically, we consider a complete evaluation of the entire \( \delta - \rho \) plane, i.e., \( 0 \leq \delta \leq 1 \), \( 0 \leq \rho \leq 1 \). For each parameter sweep, 16 equally-spaced discrete steps are considered, yielding 256 cases per value. For speed and provided that the size of \( A \) only depends on \( \delta \), a single matrix is generated for each \( \delta \)-case and used for all \( \rho \). On the contrary, an independent \( s \)-sparse signal \( \tilde{x} \) is generated for every case. Fig. 5 provides the Donoho-Tanner diagrams \((\delta - \rho)\) obtained for each value of \( n \) (by columns) for the three different types of matrices considered (by rows). For fairness of comparison the Gaussian random matrices were used as seeds for generating the ANN-BCASCs of equal size.

Pixels in dark blue in the plots of Fig. 5 correspond to exact recovery, with normalized error values \(<10^{-15}\) in most cases. At first sight it seems that the recovery performance of the three alternatives is equivalent, quite regardless of \( n \). It can be confirmed that the area of exact recovery for the ANN-BCASC matrices is always greater or equal than for the other alternatives. Furthermore, errors in that area are lowest for ANN-BCASC matrices. Unfortunately, the advantage of using ANN-BCASCs is slight and cannot be easily observed in the plots of Fig. 5. For this reason, we carry out a minimalist statistical analysis of the data in each of the \( \delta - \rho \) graphs. Specifically, we generate both a histogram and a survivor function \[90\] for the empirical distribution of recovery errors. In practice, we use the negated logarithm of the error to ease the visualization. The histograms obtained for each value of \( n \) are shown in the first row of Fig. 6 while the survivor functions are given in the second row. In principle it is desirable to have two well-concentrated regions in the histogram: a first one corresponding to failure cases, close to zero, and a second one of success cases, which should be centered as far as possible to the right of the previous. Regarding the survivor functions, the superiority of one matrix class over another translates into almost no cases falling between the two histogram peaks, witness the superiority of the Fourier ensembles and ANN-BCASCs.

The histograms in Fig. 6 exhibit indeed two well-separated regions, as expected. Furthermore, one can observe that these two regions are most separated for ANN-BCASCs and least for random Gaussian matrices, meaning lower recovery error in the (almost) exact recovery cases. It is also remarkable that almost no cases fall between the two histogram peaks, witnessing a sharp transition between failure or rough recovery and exact recovery. Regarding the survivor functions, the random Gaussian matrices yield curves that are significantly below those for Fourier ensembles and ANN-BCASCs, confirming that significantly poorer coherence also translate into significantly poorer recovery performance. Despite the closeness between the survivor functions of the Fourier ensembles and ANN-BCASCs.

\[S_X(\xi) = 1 - F_X(\xi), \quad \text{where } F_X(\xi) = \int_{-\infty}^{\xi} f_X(u)du \text{ is the cumulative distribution function (CDF) of } X.\]
the ANN-BCASCs, the latter are almost always over the former in the decaying region that corresponds to the last peak in the histogram plots, showing that, despite it might not be of practical relevance, there is indeed a slight improvement with respect to Fourier ensembles too.

In Table III we collect the coherence of all the matrices generated for the CS recovery performance evaluation with $m \leq n/2$ (recall that the cases $m \approx n$ are of no interest in CS). In terms of coherence it is clear that ANN-BCASCs widely outperform the other alternative constructions in all cases considered, especially in the interesting cases of $m \ll n$. Furthermore, the coherence of the ANN-BCASCs is relatively close to the theoretical lower bound. To the best of our knowledge, we are the first in generating approximate BCASCs for such large values of $n$.

**VI. Conclusion**

In this paper an approximate method for constructing BCASCs has been presented and evaluated. The method is a modification of the construction procedure proposed in [20] which uses ANN of each complex codeword to select the most relevant summands to include in an approximate summation that mimics an integral over all the complex rotations of all other codewords during construction of the codes. The main motivation for such an approach was the excessive computational complexity of the reference method in [20]. More specifically, each iteration has complexity $O(m^2 n^2 K)$, where $n$ is the number of $m$-dimensional codewords in the BCASC and $K$ the number of summands to consider in an approximate integral. The quadratic dependency on both $m$ and $n$ precludes the generation of BCASCs for problems of large dimensionality. The proposed approach is able to reduce the complexity of each iteration to $O(mn_{\text{rot}} \log (mn_{\text{rot}}))$ (for some fixed number of NN), where $n_{\text{rot}}$ is the number of steps used to discretize the complex rotation domain, which in our case does not equal the number of summands (per codeword) to consider in an approximate integral, as in the original algorithm. In other words, the complexity is no longer quadratic on $m$ and $n$, but linear on both parameters. This yields a large potential for execution time reduction.
which easily reaches several orders of magnitude as $m$ and $n$ grow. Numerical evaluation showed that the method is rather insensitive to the selection of the number of NN to consider and, in fact, a fixed number of NN that is very low compared to $n_{\text{tot}}$. Sufﬁces to generate BCASCs that reach the coherence of those generated via the reference algorithm. Furthermore, we observed that our approach often results in a reduced coherence when compared to the reference, especially for large $n$.

The capability of constructing BCASCs for large values of $m$ and $n$ is of fundamental interest in the context of Compressive Sensing (CS). CS theory shows that most real signals can be recovered from a reduced number of measurements due to the fact that the minimum number of measurements necessary to retain the signal information is directly related to the amount of information contained in the signal rather than to its ambient dimensionality. In fact, the signal dimensionality $n$ can be arbitrarily large, often arising from a ﬁne discretization of a continuous domain. The proposed approach opens a way of generating close-to-optimal CS measurement matrices in such cases, thanks to its reduced computational complexity. We have studied the performance of BCASCs generated using the proposed method as CS measurement matrices in the problem of sparse signal recovery. For comparison, common CS measurement matrices, such as (complex) random matrices and Fourier ensembles of the same size have also been evaluated. The BCASCs have been found to outperform the other alternatives both in terms of coherence and recovery error.

ACKNOWLEDGMENT

The authors would like to thank Dr. H. Zörlein for providing the code of his algorithms for generating BASCs and BCASCs, his readiness to help, and his interest in this work.

REFERENCES

[1] P. Tammes, “On the origin of number and arrangement of the places of exit on the surface of pollen-grains,” Recueil des Travaux Botaniques Néerlandais, vol. 27, pp. 1–84, 1930.
——, “Scalable nearest neighbor algorithms for high dimensional data,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 36, 2014.

[88] H. A. Zörlein, “Channel coding inspired contributions to compressed sensing,” Ph.D. dissertation, Faculty of Engineering and Informatics, 2015. [Online]. Available: [https://doi.org/10.18725/OIPARU-3257](https://doi.org/10.18725/OIPARU-3257)

[89] A. Khachatryan, S. Semenovovskaya, and B. Vainshtein, “Statistical-thermodynamic approach to determination of structure amplitude phases,” Sov. Phys. Crystallography, vol. 24, no. 5, pp. 519–524, 1979.

[90] ——, “The thermodynamic approach to the structure analysis of crystals,” Acta Crystallographica Section A, vol. 37, no. 5, pp. 742–754, Sep. 1981. [Online]. Available: [https://doi.org/10.1107/S0567739481001630](https://doi.org/10.1107/S0567739481001630)

[91] D. J. Love, “Grassmannian subspace packing,” [https://engineering.purdue.edu/~djlove/grass.html](https://engineering.purdue.edu/~djlove/grass.html), 2006, [Online; accessed 06-February-2017].

[92] A. Chambolle and T. Pock, “A first-order primal-dual algorithm for convex problems with applications to imaging,” J. Math. Imaging Vis., vol. 40, no. 1, pp. 120–145, May 2011. [Online]. Available: [http://dx.doi.org/10.1007/s10851-010-0251-1](http://dx.doi.org/10.1007/s10851-010-0251-1)

[93] D. Donoho and J. Tanner, “Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing,” Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, vol. 367, no. 1906, pp. 4273–4293, 2009. [Online]. Available: [http://rsta.royalsocietypublishing.org/content/367/1906/4273](http://rsta.royalsocietypublishing.org/content/367/1906/4273)

Miguel Heredia Conde received the Diploma degree in industrial engineering, specialized in automation and electronics, from the University of Vigo, Vigo, Spain, in 2012. He wrote his Diploma thesis on “Fast and Robust Localization for RGD-D Sensors” at the Autonomous Intelligent Systems group of the University of Freiburg, Freiburg, Germany, within an exchange program with the University of Vigo. After graduation he started to work as researcher in the research group on Photogrammetry and Close Range Teledetection of the University of Vigo.

In 2013, he joined the Center for Sensorsystems (ZESS) and the Research Training Group GRK 1564 “Imaging New Modalities”, University of Siegen, Siegen, Germany, where he pursued his Ph.D. degree with main focus on applying Compressive Sensing techniques to improve the quality of Time-of-Flight (ToF) Imaging systems based on the Photonic Mixer Device (PMD) until its conclusion in 2016. Since then he leads the research group on Compressive Sensing for the Photonic Mixer Device at ZESS.

Miguel Heredia was one of the recipients of the 2006 Academic Excellence Prices, awarded by the Government of Galicia, Spain, in acknowledgment of outstanding academic performance.

His current research interests include computer vision, depth sensing, with focus on ToF imaging sensors, PMD technology, Compressive Sensing, coding strategies, and different aspects of signal processing. He is a member of the IEEE and ITG/VDE professional societies.