A Decoder for the Color Code with Boundaries

Skylar Turner · Josey Hanish · Eion Blanchard · Noah Davis · Brian La Cour

Received: March 27, 2020/ Accepted: date

Abstract We introduce a decoder for the 3D color code with boundaries, which is a variation of the restriction decoder introduced by Kubicka and Delfosse. Specifically, we adapt the lift procedure to efficiently find a correction on qubits adjacent to a boundary. We numerically estimate a threshold of $1.5\% - 3\%$ for $X$ errors, and a threshold of $0.1\% - 0.2\%$ for $Z$ errors. Our work is a first step towards characterizing the performance of Bombin's recently proposed "colorful quantum computing."

Keywords Quantum error correction · Color code · Measurement-based quantum computing

1 Introduction

Quantum computers promise to solve certain problems faster than their classical counterparts [1,2], but the quantum systems used to build one tend to be highly sensitive to noise. Using error-correcting codes it is possible to build
a fault-tolerant quantum computer, one in which noise is corrected before it can degrade the computation. In such a code, a logical qubit is encoded in many physical qubits, introducing redundancy that makes it possible to correct errors from noise on the physical qubits. In order to correct noise, a decoding algorithm must interpret syndrome information in order to identify a correction operator to remedy the error that has occurred.

The leading candidate for a fault-tolerant universal quantum computing scheme is the surface code with defect braiding and magic state distillation [3, 4]. The surface code has a high noise threshold (near 10% [5]), below which the likelihood of errors can be made arbitrarily small by using larger numbers of physical qubits. The threshold of the 2D color code has been found to be comparable, also near 10% [6]. However, the surface code can transversally implement only a limited set of quantum gates and requires costly magic state distillation to implement a universal gate set. These drawbacks inspire research into alternative fault-tolerant schemes. “Colorful quantum computing” [7] is one such promising alternative.

In this paper, we investigate quantum error correction on the 3D color code with boundaries, also known as the “tetrahedral color code,” which is a necessary structure for “colorful quantum computing” [7]. In Section 2, we introduce an algorithm for decoding $X$ and $Z$ errors on the tetrahedral color code. This algorithm was adapted from the “restriction decoder,” an algorithm for decoding color codes with periodic boundaries [6]. In Section 3, we characterize the performance of this decoder on independent, identically distributed $X$ and $Z$ errors and present thresholds for error-correction such that for noise below the threshold, the probability of logical errors can be made arbitrarily small by using larger size codes. Finally, we interpret our results in relation to colorful quantum computing and other recent proposals for fault-tolerant quantum computing in Section 4.

1.1 Colorful quantum computing

Colorful quantum computing uses a 3D color code that can be implemented on a 2D lattice of physical qubits using “just-in-time decoding.” This scheme achieves universality with only transversal gates, circumventing the Eastin-Knill Theorem [8] by relying on non-local classical computing [7]. Transversal gates, which operate “qubit-wise” on physical qubits, are highly desirable for fault-tolerant codes because they propagate errors only locally [9]. Colorful quantum computing is fault-tolerant, though the noise threshold for fault-tolerance has only been investigated theoretically [7] and is very very low. However, thresholds found using direct simulation of the error-correction process are usually much higher.

The tetrahedral color code admits the following transversal logical gates [9]: The logical Controlled-X (CNOT) gate can be applied by applying CNOT pairwise between corresponding physical qubits of two logical qubits. The Controlled-Z ($cZ$) gate is applied analogously, but by matching qubits on only
one facet of a primal tetrahedron to qubits on one facet of another primal tetrahedron. Most importantly, the $T$ gate is transversal. The logical $T$ gate is applied by applying $T$ to a set of the physical qubits and $T^\dagger$ to the remainder, such that the two sets are a bipartition of the lattice. The logical $T^2$ gate is also transversal, as are the logical $X$ and $Z$.

When supplemented with measurements in the $Z$ basis $\{|0\rangle, |1\rangle\}$, $X$ basis $\{|\pm\rangle = |0\rangle \pm |1\rangle\}$, and $X + Y$ basis $\{|0\rangle \pm e^{i\pi/4} |1\rangle\}$, this set of gates for the tetrahedral color code becomes universal in measurement-based quantum computing (MBQC). An example of how to implement the Hadamard gate in such a scheme can be found in Refs. [10,7]. MBQC is equivalent to the circuit model, but implemented differently - all entanglement is present in a resource state at the beginning of the computation. Gates are simulated by qubit measurements on a highly entangled cluster state, and a classical computer processes the Pauli frame [3]. Universality requires a cluster state of at least two dimensions and the ability to measure qubits in the $X$, $Z$, and $X + Y$ bases. Generating a cluster state requires a source of $|\pm\rangle$ states and the ability to apply the $cZ$ gate between neighboring qubits [11].

Colorful quantum computing is a type of MBQC that encodes the logical qubits of the cluster state in a tetrahedral color code. Arranged in this way, the initial state forms a 3D lattice of qubits. The cluster state is initialized using $cZ$ gates, $|\pm\rangle$ states, and ancilla qubits. Afterwards, the logical qubits form a cluster state up to known single-qubit Pauli errors, i.e. the Pauli frame. An $X$ decoder is used to determine the Pauli frame and the locations of errors are stored in a classical memory. The measurement pattern is then enacted by measuring the logical qubits in the appropriate basis, which requires either an $X$ decoder, a $Z$ decoder, or both. If the logical measurement is in the $X + Y$ basis, a transversal $T$ gate is applied before measuring in the $X$ basis, which requires a $Z$ decoder to interpret. This scheme does not require magic state distillation because the $cZ$ and $T$ gates are transversal, as are measurements in the Pauli bases. The ability of colorful quantum computing to realize fault-tolerant, universal quantum computing without magic state distillation motivates our development of decoders for the color code with boundaries.

1.2 Error correction in the stabilizer formalism

3D color codes are a type of error-correcting stabilizer code. In the stabilizer formalism, states are described not by a wavefunction but by stabilizer operators. The state described by a set of stabilizer operators $S$ is the $+1$ eigenstate of each operator, $S|\psi\rangle = |\psi\rangle, S \in S$. An $n$-qubit state described by $k$ independent, commuting stabilizers inhabits a $2^{n-k}$ dimensional subspace. Operators that commute with the set of stabilizers are denoted $Z$, and non-trivial logical operators are $Z \setminus S$. In the context of stabilizer codes, the code space is the $2^{n-k}$ dimensional $+1$ eigenstate of the code stabilizers. To define a basis for the $n - k$ logical qubits, one chooses $n - k$ logical $\tilde{Z}_i$ operators $\tilde{Z}_i$ from $Z \setminus S$. Then one can choose logical $\tilde{X}_i$ operators such that $\tilde{X}_i \tilde{Z}_j = (-1)^{\delta_{ij}} \tilde{Z}_j \tilde{X}_i$
An error, $E$, will anticommute with at least one stabilizer if the error is correctable. This results in a $-1$ measurement outcome when one of those stabilizers is measured. Thus, we measure a generating set of stabilizers to find the syndrome corresponding to the error $E$, which we then decode to obtain the corresponding correction operator, $\tau$. Finally, we apply this correction operator to bring the system back into the code space. Error correction fails when the combination of errors and correction operators is a logical operator; i.e., $\tau E \notin Z \setminus S$.

1.3 Structure of the color code

3D color codes are topological stabilizer codes. These codes are defined on lattices in which stabilizers correspond to topological objects, such as loops or surfaces. An accessible introduction to color codes (of all dimensions) may be found in Ref. [12]. Here, we investigate tetrahedral color codes, which are 3D color codes with non-periodic boundary conditions [13].

We define the tetrahedral color code on a dual lattice $L^*$ such that tetrahedra [3-simplices, denoted $\Delta_3(L^*)$] specify the physical qubits, vertices [0-simplices, $\Delta_0(L^*)$] generate the set of $X$ stabilizers, $S_X$, and edges [1-simplices, denoted $\Delta_1(L^*)$] generate the set of $Z$ stabilizers, $S_Z$. The tetrahedral color code depicted in Fig. 1 has 15 qubits, four $X$ stabilizers, and 18 $Z$ stabilizers.

Each vertex of $L^*$ is assigned a color from the set $C = \{r, g, b, y\}$ such that no adjacent vertices share a color. Each edge is labeled by the two colors that are absent from the vertices it connects. For example, a $by$-colored edge connects an $r$-colored vertex and a $g$-colored vertex.

We refer to the four boundary vertices of the code as quasivertices. The quasivertices do not correspond to $X$ stabilizers, and the tetrahedron formed by connecting all the quasivertices does not correspond to a physical qubit. However, the edges and tetrahedra adjacent to the quasivertices are $Z$ stabilizers and physical qubits, respectively.

For the purposes of physical implementation, it is simpler to define the tetrahedral color code differently. Before, we used the dual lattice description $L^*$, and now we introduce the primal lattice description $L$. In the primal lattice, qubits are vertices, $Z$ stabilizers correspond to faces, and $X$ stabilizers are cells. The four boundaries now appear as triangular facets.

In the simplest possible primal lattice, shown in Fig. 1, each face is adjacent to four qubits and each cell is adjacent to eight qubits. It is harder to see, but in the corresponding dual lattice each edge and vertex is adjacent to four or eight tetrahedra, respectively. We analyze a family of codes built in the dual on the body-centered cubic, or bcc, lattice. In the corresponding primal lattice each cell in the bulk is a bitruncated cube.

Every tetrahedral color code encodes a single logical qubit in a larger number of physical, or code, qubits. A common basis for a single logical qubit is one in which the logical operators $\bar{X}$ and $\bar{Z}$ are tensor products of $X$ and $Z$ operators, respectively, on each of the physical qubits.
2 Decoding algorithms

2.1 $X$ errors — loop-like syndromes

We implement the cellular automaton-based restriction decoder described in [6], adapting it to a lattice with a tetrahedral boundary (as opposed to a lattice with periodic boundaries). The syndrome of an $X$ error appears as a collection of edges that bound the set of tetrahedra corresponding to the qubits that have errors. Given a set of edges $\sigma \subset \Delta_1(\mathcal{L}^*)$, we find the set of tetrahedra $\tau \subset \Delta_3(\mathcal{L}^*)$ such that $\sigma = \partial_3,1(\tau)$, where $\partial_{n,m}$ denotes the projection of an $n$-dimensional set of objects to its $m$-dimensional boundary.

The restriction decoder achieves this via two routines: the “sweep” decoder and the “lift” routine. One of the four lattice colors is chosen as the lift color; the rest are sweep colors. For each sweep color, the decoder considers the subset of edges in $\sigma$ labeled with the particular sweep color (i.e. only syndrome edges not incident to a sweep-colored vertex are considered). The decoder then runs the sweep decoder [14], a toric-code decoder, on this subset of edges to obtain a set of faces whose boundary is the set of edges having been considered. The faces resulting from each sweep color are taken in union as $\gamma \subset \Delta_2(\mathcal{L}^*)$. Then $\partial_2,1(\gamma) = \sigma$.

Next, considering the subset of vertices assigned the lift color, the decoder identifies the set of tetrahedra $\tau \subset \Delta_3(\mathcal{L}^*)$ such that $\partial_{3,2}(\tau) \supset \gamma$ and for each lift vertex $v$, $\partial_{3,2}(\tau)|_v = \gamma|_v$, where $\gamma|_v$ is the subset of $\gamma$ containing $v$. We note the naïve treatment of boundary vertices in the lift process. Lifting each bulk (interior) vertex runs in constant time since there are $2^{12}$ possible subsets of tetrahedra incident to each bulk vertex (a consequence of the lattice geometry). With non-periodic boundaries as in our code lattice, however, the tetrahedral neighborhoods of boundary vertices become increasingly dense at a quadratic asymptotic rate $O(d^2)$, where $d$ is the code distance. So our naïve approach runs in at-worst exponential time: $O(2^{d^2})$.
To overcome this computational hurdle, we adapt the “peel” algorithm described in [15] to lift boundary vertices more efficiently. (See Fig. 2.) In fact, on code lattices with small distance, this is more efficient than the naïve lift on even the bulk vertices. The peel algorithm proceeds on each lift vertex \( v \) by taking the set of tetrahedra \( \tau \mid v \) incident to \( v \) and considering \( \partial_3 (\tau \mid v) \setminus (\partial_3 (\tau \mid v) \mid v) \). That is, we consider the set of faces which are incident to \( \tau \mid v \) but disjoint from \( v \) itself (i.e. faces whose vertices are in the link of \( v \)); these faces form a topological sphere when \( v \) is a bulk vertex and form a triangular facet when \( v \) is a boundary vertex.

We consider the subset of “intermediate syndrome” faces which is output from the sweep decoder that are incident to \( v \): \( \gamma \mid v \) (brown faces in Fig. 2a). We project this set of faces \( \gamma \mid v \) into a set of edges (purple in Fig. 2b) on the aforementioned “peeling surface” (topological sphere or tetrahedral facet). The peel algorithm finishes by identifying the set of faces (Fig. 2c) on the surface whose edge-boundary matches these projected edges, which can be done efficiently following Ref. [15], then projects the triangle faces back up to tetrahedra, with the associated lift vertex \( v \) as the fourth vertex for each tetrahedra. The resulting set of tetrahedra from all lift vertices is the desired \( \tau \). The peel subroutine runs in asymptotically constant time for bulk vertices. Its runtime on boundary vertices is now \( O(d^2) \), a vast improvement over the original lift procedure. In addition, decoding is done locally.

2.2 Z errors — point-like syndromes

The syndrome of Z errors is a set of vertices corresponding to X stabilizers that returned a −1 measurement outcome. We use a minimum-weight perfect matching (MWPM) subroutine to find edges that connect these vertices. First, we restrict the dual lattice to include only vertices of two colors; that is, for \( C \) as above, vertices assigned color \( \kappa_i \) or \( \kappa_j \) are removed, while vertices assigned \( \kappa_k \) or \( \kappa_l \) remain (for \( \kappa_i \neq \kappa_j \neq \kappa_k \neq \kappa_l \)). We then match the syndrome vertices on each twice-restricted lattice and take the union of all returned sets of edges. Those edges correspond to the syndrome that would occur if X errors had occurred on the qubits actually affected by Z errors. Finally the X error decoder, described above, is called as a subroutine to identify the error qubits. This is the approach used in [16], which can be thought of as a specific implementation of the more general process described by the restriction decoder [6].

To handle the boundary, we adapt the MWPM algorithm to allow for vertices to match to the nearest quasivertex that remains in the twice-restricted lattice. We add edges between each vertex and its nearest quasivertex, which can be either color. In addition, we add weight-0 edges between quasiverses so that the solutions that do not include a quasivertex-vertex edge are considered to be matchings. This is the same process that is used to decode the surface code using MWPM [17], and it is not computationally any more difficult than decoding without boundaries. Our decoding scheme corrects the smallest set
Fig. 2: (Color online) Illustration of the efficient lift procedure on a boundary quasivertex. In a), lift must find which tetrahedra in the neighborhood of the red quasivertex have a boundary of faces that match the brown highlighted ones. b) We project the marked faces onto the facet nearest the red quasivertex, marking a purple edge for each face’s intersection with the facet. c) On the facet, it is possible to quickly find the 2D boundary of these edges marked in black. These faces have a one-to-one correspondence with the error qubits adjacent to the red quasivertex.

of error qubits that could have caused the given syndrome. This is a justified choice since the probability $p^n$ of an $n$-qubit error decreases as $n$ increases; smaller errors are more likely. Because of this, the decoder has a tendency to match bulk vertices to the boundary quasivertices, while an observer who knows the positions of the original error qubits would expect that vertex to be matched to a bulk vertex. This limits successful error-correction of $Z$ errors, as the smallest-weight logical errors of tetrahedral color codes are $Z$ errors on the $d$ qubits that connect two quasivertices by the edges between two facets. It may be possible to weight edges that connect quasivertices to bulk vertices to avoid this situation, similar to the flag qubit scheme used in Ref. [18]. We leave more sophisticated ways to handle $Z$ boundary errors for future research.
The $Z$-error decoder is necessarily less successful than the $X$-error decoder: there are fewer $X$ stabilizers than there are $Z$ stabilizers, and $X$ stabilizers have higher weight than $Z$ stabilizers, so $Z$-error syndromes contain less information than $X$-error syndromes. The $Z$-error decoder also has a worse runtime than the $X$-error decoder because of the repeated MWPM subroutines.

3 Characterization of performance

We numerically tested both the $X$ decoder and the $Z$ decoder under an assumption of independent and identically distributed local noise. Each data point was found using between $10^4$ and $10^7$ Monte Carlo simulations. Error bars represent the standard deviation of the mean, given by $\sqrt{p_f(1-p_f)/N}$, where $p_f$ is the decoding failure probability and $N$ is the number of Monte Carlos simulations used.

\[ \text{Decoding failure rate} \]

\[ \text{Error rate, } p \]

Fig. 3: (Color online) Performance of the $X$ decoder under local noise.

The data found from testing of the $X$ decoder is shown in Fig. 3. We find that smaller code sizes definitely outperform larger ones above $p = 3.3\%$, and larger code sizes perform better than smaller ones below $p = 2.5\%$. Because the error bars overlap in between these points, we cannot pinpoint a precise threshold. We therefore conclude that the threshold for the $X$ decoder is between 2.5\% and 3.3\%.
The $Z$ decoder data is shown in Fig. 4. Sub-threshold data, where larger codes outperform smaller codes, appears at or above $p = 0.1\%$. The different code sizes start to perform similarly near $p = 0.02\%$ and exhibit some crossover at $p = 0.01\%$. We therefore estimate the threshold for the $Z$ decoder to be between $0.01\%$ and $0.02\%$.

4 Conclusion

The highest possible values for correction of $X$ errors and $Z$ errors in a 3D color code with periodic boundaries is $27.6\%$ and $1.9\%$ respectively [19]. Our decoder achieves thresholds an order of magnitude below this. Previous tests of the restriction decoder on the $bcc$ lattice without boundaries found a threshold of $0.077\%$ for the $Z$ decoder, and estimated (though not numerically tested) a threshold of $13.1\%$ for the $X$ decoder [5]. The $Z$ decoder threshold of $0.046\%$ found in [20] using a “clustering decoder,” to our knowledge, is the only previous threshold found for a 3D color code with boundaries (though it was not tested on the $bcc$ lattice). The role of boundaries is unclear; on one hand, we do not expect the thresholds for two codes which only differ in their boundaries to be much different, because for large code sizes almost all qubits lie in the bulk of the lattice. For the toric and the surface codes, this is true. However, for the family of 3-dimensional color codes where the number of physical qubits scales
as $d^3$ it is difficult to test large enough code sizes that boundaries become unimportant. Indeed, 31.2% of qubits in the largest code we test, $d = 17$, lie on a boundary. It may be an interesting question to pursue how much an experimentally-determined threshold depends on the size of the codes used in numerical simulation.

While our decoders do not perform as well as previous ones, we are able to adapt the restriction decoder to boundaries. The success of adapting the restriction decoder into a fault-tolerant protocol for the 2D color code [18] inspires hope that similar approaches may be used to improve performance of the tetrahedral color code. Tetrahedral color codes are the building blocks of a particularly elegant proposal for fault-tolerance called colorful quantum computing, which can be implemented using only transversal gates and a 2D lattice of physical qubits. Our work is a first step towards being able to numerically assess the performance of colorful quantum computing.

References

1. P.W. Shor, Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM pp. 124–134 (1994)
2. L.K. Grover, Proceedings, 28th Annual ACM Symposium on the Theory of Computing p. 212 (1996)
3. R. Raussendorf, J. Harrington, Phys. Rev. Lett. 98, 190504 (2007). DOI 10.1103/PhysRevLett.98.190504. URL https://link.aps.org/doi/10.1103/PhysRevLett.98.190504
4. A.G. Fowler, M. Mariantoni, J.M. Martinis, A.N. Cleland, Phys. Rev. A 86, 032324 (2012). DOI 10.1103/PhysRevA.86.032324. URL https://link.aps.org/doi/10.1103/PhysRevA.86.032324
5. E. Dennis, A. Kitaev, A. Landahl, J. Preskill, Journal of Mathematical Physics 43(9), 4452 (2002)
6. A. Kubica, N. Delfosse, arXiv e-prints arXiv:1905.07393 (2019)
7. H. Bombin, arXiv e-prints arXiv:1810.09571 (2018)
8. B. Eastin, E. Knill, Physical review letters 102(11), 110502 (2009)
9. H. Bombin, arXiv e-prints arXiv:1810.09575 (2018)
10. P. Webster, S.D. Bartlett, (2019)
11. R. Jozsa, NATO Science Series, III: Computer and Systems Sciences. Quantum Information Processing, From Theory to Experiment 199, 137 (2006)
12. A. Kubica, M.E. Beverland, Phys. Rev. A 91, 032330 (2015). DOI 10.1103/PhysRevA.91.032330. URL https://link.aps.org/doi/10.1103/PhysRevA.91.032330
13. H. Bombin, M. Martin-Delgado, Phys. Rev. Lett. 98, 160502 (2007). DOI 10.1103/PhysRevLett.98.160502. URL https://link.aps.org/doi/10.1103/PhysRevLett.98.160502
14. A. Kubica, J. Preskill, Phys. Rev. Lett. 123, 020501 (2019). DOI 10.1103/PhysRevLett.123.020501
15. A.B. Akshionis, P.K. Sarvepalli, Decoding toric codes on three dimensional simplicial complexes (2019). ArXiv:1911.06056v1
16. A.B. Akshionis, P.K. Sarvepalli, Phys. Rev. A 98, 012302 (2018). DOI 10.1103/PhysRevA.98.012302. URL https://link.aps.org/doi/10.1103/PhysRevA.98.012302
17. D.S. Wang, A.G. Fowler, A.M. Stephens, L.C.L. Hollenberg, arXiv e-prints arXiv:0905.0531 (2009)
18. C. Chamberland, A. Kubica, T.J. Yoder, G. Zhu, arXiv e-prints arXiv:1911.00355 (2019)
19. A. Kubica, M.E. Beverland, F. Brandao, J. Preskill, K.M. Svore, Physical review letters 120(18), 180501 (2018)
20. B.J. Brown, N.H. Nickerson, D.E. Browne, Nature communications 7, 12302 (2016)