Visualization for Quantum Computing: Current State and Future Directions

Feng Liang, Xiaolin Wen, Shixian Zhou, Shaolun Ruan, Yuxuan Du, Qiang Guan, Jun Ye, and Yong Wang

***Abstract*—Visualization has play an essential role in understanding quantum computing (QC), from the Bloch sphere to interactive visual analytics systems. Yet as QC advances and its community grows, there are still no systematic surveys of visualization for quantum computing (VIS4QC) to map the landscape or guide future work. To fill this gap, three questions must be answered from three perspectives: *why* visualizations are created, *what* QC subjects they address, and *how* they are designed. In response, we collect publications and tools from 2015 to 2025, curate 69 out of them, and propose a three-dimensional taxonomy in collaboration with QC experts. It summarizes the research space of VIS4QC along three dimensions. The first captures *why* visualizations are created, grouping them into those for *learning and interpretation* and those for *productivity and utility*. The second captures *what* QC subjects are visualized, spanning *qubits and states*, *gates and circuits*, *machines and systems*, and *algorithm development*. The third examines *how* existing work visualizes QC subjects: computational data in QC requires novel visualization techniques, while operational data can use general-purpose approaches. Examining *why* and *what* together with the source of publications and tools reveals that academia and industry focus on largely different areas: academia concentrates on visualizing quantum states for learning, while industry provides basic circuit diagrams for productivity. Neither side serves the other's needs. Algorithm development and machines/systems receive the least coverage overall, and productivity tools for algorithm development are nearly absent despite the fact that algorithms are where QC delivers practical value. These structural patterns, together with qualitatively distinct challenges at each abstraction level, point to concrete future directions for the field, including visualizations for quantum phase, dynamic quantum circuits, quantum visual debugging and algorithm mechanisms.**

***Index Terms*—Visualization, quantum computing, quantum state, quantum circuit, visual analytics, survey.**

I. Introduction

Quantum computing (QC) promises computational capabilities beyond the reach of classical computing. Quantum algorithms can theoretically solve many classically intractable problems with considerable speedup. For example, Shor's Algorithm [1], a cornerstone of QC research, is exponentially faster than classical alternatives in factoring integers. Since then, numerous algorithms have been developed to harness quantum computational advantages [2], [3]. On the hardware side, current quantum computers, while still noisy and unable to perform fault-tolerant computation, have demonstrated practical applications (e.g., [4], [5]), providing early evidence of quantum utility [6].

As quantum technologies advance and demonstrate increasing real-world utility, the quantum industry has developed various tools and software development kits (SDKs) (e.g., Qiskit [7], Cirq [8], and PennyLane [9]) to make quantum computing more accessible. Most include built-in visualization features that help users (e.g., physicists, students, developers, and engineers) understand various aspects of QC. In academia, visualization has been essential for understanding QC and the underlying quantum physics. For example, the Bloch Sphere [10], invented decades before Feynman's proposal of quantum computation [11], is the earliest visual representation of quantum states and remains fundamental for teaching and understanding QC today. Many more visual representations, tools, and systems have since been developed for different QC use cases.

As QC continues to evolve, its growing user base will lead to increasing needs for visualizations for understanding, debugging, and optimizing QC systems. Despite this importance, to the best of our knowledge, there are still no systematic surveys of Visualization for Quantum Computing (VIS4QC) that map the research landscape or guide future work. Because VIS4QC lies at the intersection of visualization and QC, a clear taxonomy is essential for researchers to navigate both fields, which, however, still does not exist. Without these, newcomers lack a structured entry point into the field, and researchers cannot systematically identify which problems remain underexplored.

A useful taxonomy for VIS4QC must address three fundamental questions. First, *why are visualizations created?* Different purposes, such as teaching newcomers versus supporting expert workflows, lead to distinct design requirements. Second, *what QC subjects are visualized?* Quantum computing spans multiple abstraction levels, from individual qubits to full algorithms, each with distinct data types and visualization needs; yet, existing work concentrates on only a few. Third, *how are QC subjects visualized?* Existing techniques face different challenges at each level, from scalability limits for quantum states to the lack of visualization support for complete algorithms. Fig. 1 overviews the taxonomy.

To provide both a comprehensive survey and a systematic taxonomy that answers these questions, we collected and analyzed existing VIS4QC publications and tools (PTs). We

Feng Liang, Xiaolin Wen, Yuxuan Du, and Yong Wang are with Nanyang Technological University. E-mail: {feng011, xiaolin004}@e.ntu.edu.sg, {yuxuan.du, yong-wang}@ntu.edu.sg. Yong Wang is the corresponding author.

Shixian Zhou is with Hangzhou Dianzi University. E-mail: 241330022@hdu.edu.cn.

Shaolun Ruan is with Singapore Management University. E-mail: slruan.2021@phdcs.smu.edu.sg.

Qiang Guan is with Kent State University. E-mail: qguan@kent.edu.

Jun Ye is with Institute of High Performance Computing, A*STAR. E-mail: yej@a-star.edu.sg.

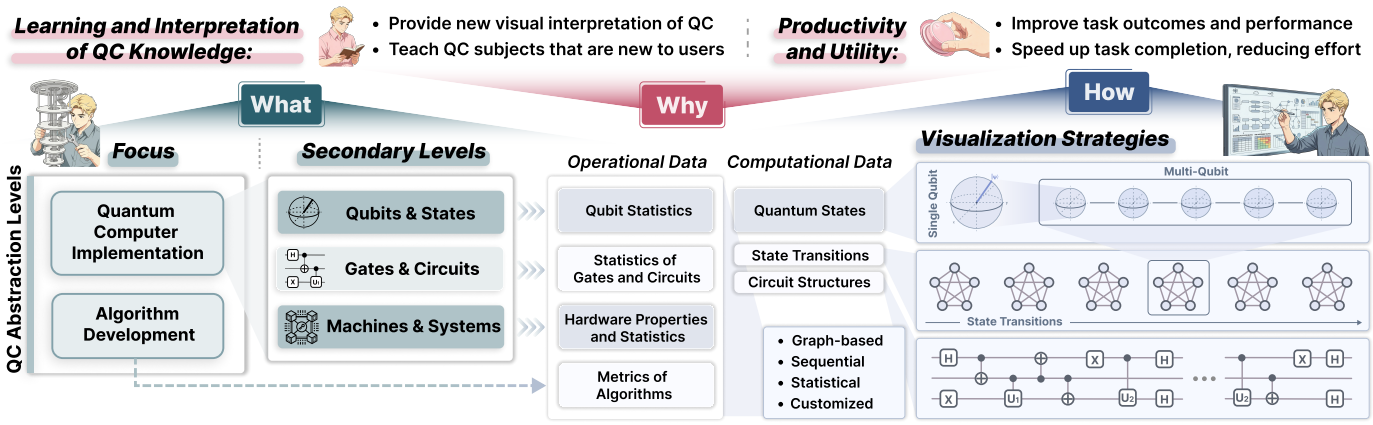


Fig. 1. Overview of our three-dimensional taxonomy for VIS4QC. *Why* (top) distinguishes visualizations created for learning and interpretation from those for productivity and utility. *What* (left) organizes QC subjects by abstraction level, from qubits and states through gates and circuits to machines and systems, alongside algorithm development. *How* (right) separates data into operational and computational categories and surveys the visualization techniques used for each, with selected examples of single-qubit, multi-qubit, state-transition, and circuit-structure visualizations.

performed an extensive search of academic journals and conferences spanning QC, Visualization (VIS), Computer Graphics (CG), Human-Computer Interaction (HCI), and Software Engineering (SE), covering 41666 publications from 2015 to 2025, as shown in Table IV.1. We also used the recent agentic search tools (e.g., Perplexity Deep Research) to discover relevant PTs from the broader Internet. Additionally, we performed recursive snowballing to ensure that earlier work was not missed. After filtering out irrelevant PTs by reading their summaries, we compiled 69 core PTs (Table IV.2). We then manually analyzed them, distilled a taxonomy, and coded them accordingly.

Our taxonomy characterizes the research space of VIS4QC along three dimensions: *Usage and Purpose*, *Abstraction Levels of QC*, and *Visualizations*. The first two define the *problem space*. *Usage and Purpose* capture **why** a visualization is created, either for learning and interpretation, where visualizations help users understand QC concepts, or for productivity and utility, where visualizations help experienced users perform QC tasks more efficiently. *Abstraction levels of QC* capture **what** QC subjects are visualized, spanning from low-level qubits and states, through gates and circuits, to QC machines and systems, as well as algorithm development at a higher conceptual level. The third dimension surveys the *solution space*: **how** existing work visualizes QC subjects. It distinguishes techniques for data intrinsic to the computation from those for operational data and examines the visualization techniques employed.

Examining **why** and **what** , together with the source of PTs, reveals that academia and industry focus on largely different areas: academia concentrates on visualizing quantum states for learning (30 out of 31 related PTs are from academia), while industry provides basic circuit diagrams for productivity (17 out of 27 related PTs are industry tools). Neither side serves the other's needs. Algorithm development and machines/systems receive the least coverage overall, and productivity tools for algorithm development are nearly absent (3 PTs) despite being where QC delivers practical value. With real-world quantum applications emerging in areas such as finance [4] and drug discovery [5], these underexplored regions represent significant opportunities for visualization researchers. We further identify visualization challenges that differ qualitatively across abstraction levels, from fundamental scalability limits at the state level, through evolving conventions for dynamic circuits and transpilation, to the need for mechanism-based rather than state-based visualization at the algorithm level. Each of them informs distinct future directions, such as novel quantum state visualizations that circumvent scalability limits, visualizations for hybrid quantum computing, and further exploration in algorithm mechanisms.

In summary, this survey makes the following contributions:

- A comprehensive collection of 69 core PTs in VIS4QC (see Table IV.2), providing visualization researchers with an entry point into the field.
- A three-dimensional taxonomy that defines the research space of VIS4QC along *why* (usage and purpose), *what* (QC abstraction levels), and *how* (visualization techniques), enabling systematic navigation of this interdisciplinary space.
- A concise background on QC concepts and context relevant to visualization research, making QC accessible to visualization researchers.
- A diagnosis of the field's coverage by examining the problem space, identifying structural gaps, challenges at each abstraction level, and concrete future directions.

The remainder of this survey is organized as follows. Section II reviews and compares existing related surveys. Section III provides a concise QC tutorial relevant to visualization research. Section IV details our methodology for collecting references and building the taxonomy. Sections V, VI, and VII present our three-dimensional taxonomy: *why* (usage & purpose), *what* (QC abstraction levels), and *how* (visualization techniques), respectively. Section VIII discusses the current state of VIS4QC with the problem space. Section IX discusses challenges and opportunities at each abstraction level and considers implications for quantum education. Finally, Section X concludes the survey.

II. Related Surveys

Various research disciplines are relevant to VIS4QC, spanning human-computer interaction research, education, and games. Quantum Human Computer Interaction (QHCI) studies address the interaction design between users and quantum computers, spanning interaction modalities such as visual circuit authoring and system exploration dashboards. Although VIS4QC intersects with QHCI, the two differ in scope: QHCI treats visualization as one component of broader interface design, whereas VIS4QC focuses on visualization itself as the primary research contribution.

Quantum Human Computer Interaction: Ashktorab et al. [12] identified four main topics of QHCI based on their interviews with domain experts. They also surveyed existing tools, SDKs, and educational resources, including games and visualizations of quantum states. However, as QC has since progressed rapidly, this survey no longer captures the current landscape of QHCI and VIS4QC research. More recently, Kim et al. [13] surveyed tools and SDKs for QC, discussed challenges in QC interfaces, and proposed design principles with prototype examples. However, as it focused on QHCI, it did not discuss VIS4QC research in depth but treated visualization as an engineering feature of quantum tools. As its taxonomy focused on tool utility, it did not map the QC landscape from a visualization perspective or identify research directions at the intersection of the two fields.

VIS4QC: A survey by Bethel et al. [14] discussed VIS4QC alongside quantum rendering algorithms in computer graphics. Regarding VIS4QC, it summarized several representations of quantum states and visualizations for quantum circuits and algorithms by showing the transition of quantum states. However, its primary focus was on rendering rather than visualization research, and it did not cover advanced VIS4QC research or provide a taxonomy. A more recent survey by Bethel et al. [15] provided more background in QC and distilled a pipeline that is common in quantum visualization processing. They also discussed engineering challenges in quantum data processing and VIS4QC. Although this survey frequently mentioned VIS4QC studies, it focused more on data processing and the computer graphics pipeline than on providing a systematic overview of VIS4QC research. It did not provide a taxonomy either.

Quantum Education and Quantum Games: Educational research is beyond our scope, but as many educators in this field call for better visual tools to aid their teaching and have contributed to VIS4QC research, we include related education surveys here and discuss the educational aspects of VIS4QC in a dedicated section later. Doyle et al. [16] interviewed educators to provide insights into the challenges of teaching quantum concepts to interdisciplinary learners. They argue for a pedagogical shift away from traditional physics-heavy approaches toward methods that balance mathematical rigor and conceptual accessibility, where visualizations may play an important role. The article by Lane et al. [17] also provides an overview of the challenges, ongoing efforts, and future directions in quantum education, noting the potential role of VIS4QC. In addition, various games have been developed for quantum education. Two surveys [18], [19] and a book chapter by Skult et al. [20] summarize existing techniques of games and storytelling related to quantum physics and QC. Although games about quantum inevitably involve graphical presentations, these articles focus more on the gaming, entertainment, and educational aspects of quantum games rather than on visualization.

In contrast to these surveys, our work focuses on VIS4QC research and provides a comprehensive survey of existing studies, a concise background on relevant QC concepts, and a taxonomy that captures the current research landscape and informs future research directions.

III. Background: Quantum Computing

Visualization for QC requires familiarity with the notations, core concepts, and computational models of quantum computing. We provide the necessary background for VIS4QC research, covering qubits, quantum states, gates, circuits, algorithms, programming, and the current state of QC in this section. Because the physical implementations of quantum computers inherently introduce noise and errors, we also briefly introduce these aspects, as addressing them remains a major challenge in QC research.

Notation: We use the braket notation throughout the survey. A ket is a column vector, denoted as $|\psi\rangle$, where ψ is an arbitrary name for this vector. A bra is the conjugate transpose of a ket, denoted as $\langle\psi|$. This can also be used to conveniently represent matrix and vector operations. For example, the inner product of two vectors $|\psi\rangle$ and $|\phi\rangle$ is written as $\langle\psi|\phi\rangle$.

A. Qubits and Quantum States

Analogous to a classical bit, a quantum bit (*qubit*) represents the smallest unit of information in quantum computation. It abstracts away the details of the underlying quantum mechanics. However, unlike a classical bit, which can only be in one of two states (0 or 1) at a time, a qubit can be in a superposition of the two states simultaneously. For an N-qubit system, the state of these qubits can be in a superposition of 2^N basis states simultaneously. In contrast, for a system of N classical bits, its state can only be one of 2^N possible states.

The state of a single qubit is a unit vector in $\mathbb{C}^2$. This unit vector is called the *state vector*. State vectors can be represented in different bases. The standard basis, also known as the computational basis, is in the form of one-hot vectors, where only one entry is 1 and the others are 0. The standard basis B of a one-qubit quantum state is written as $B = \{|0\rangle, |1\rangle\}$, where the *basis states* are $|0\rangle := \begin{bmatrix}1,0\end{bmatrix}^T$ and $|1\rangle := \begin{bmatrix}0,1\end{bmatrix}^T$. In braket notation, the state vector can be written as $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where $\alpha_0, \alpha_1 \in \mathbb{C}$ are the *amplitudes* of the basis states and $\alpha_0^2 + \alpha_1^2 = 1$.

State vectors can comprehensively describe pure quantum states but not mixed states. A quantum state that can be exactly known is defined to be a *pure state*. Otherwise, it is a *mixed state*, which is a probabilistic mixture of pure states [21, p. 100]. For example, when preparing a qubit, a real quantum processor may produce $|0\rangle$ with probability 0.999 and $|1\rangle$ with probability 0.001. The resulting state is a mixed state.

The *density matrix* is a more general representation of quantum states that can describe both pure and mixed states, capturing the aforementioned probabilistic impurity. For a single-qubit pure state $|\psi\rangle$, the density matrix is defined as $\rho := |\psi\rangle\langle\psi|$. For a single-qubit mixed state, the density matrix is defined as $\rho := p|\psi_0\rangle\langle\psi_0| + (1-p)|\psi_1\rangle\langle\psi_1|$, where $p \in [0,1)$ is the probability of the first basis state $|\psi_0\rangle$ and $1-p$ is the probability of the second basis state $|\psi_1\rangle$.

For an N-qubit state $|\phi\rangle$, its state vector is a unit vector in $\mathbb{C}^{2^N}$. The standard basis contains 2^N basis vectors. When expressed in braket notation, the state vector has 2^N terms, written as

$$|\phi\rangle = \sum_{i=0}^{2^N-1} \alpha_i |i\rangle, \tag{1}$$

where amplitudes $\alpha_i \in \mathbb{C}$, $\sum_{i=0}^{2^N-1} |\alpha_i|^2 = 1$, and $|i\rangle$ is the i-th basis vector in which the i-th entry is 1 and all other entries are 0. The constraint $\sum |\alpha_i|^2 = 1$ is called the *normalization condition*, and we assume it throughout unless otherwise specified. The density matrix for $|\phi\rangle$ is defined as follows:

$$\rho = \sum_{i=0}^{2^N-1} p_i |i\rangle\langle i|, \tag{2}$$

where p_i is the probability of the i-th basis state $|i\rangle$ and $\sum_{i=0}^{2^N-1} p_i = 1$.

B. Quantum Gates, Measurements and Circuits

In QC, two essential quantum operations for manipulating quantum states are quantum gates and measurements. A sequence of quantum gates and measurements is composed as a quantum circuit. We can represent these quantum operations in different ways, such as matrices, formulae in braket notation, and circuit diagrams.

Quantum gates are mathematically unitary operators. A quantum gate can act on one or multiple qubits. We summarize common quantum gates and their representations in Table III.1.

Measurement is the primary method of extracting information from qubits. While quantum mechanics admits various types of measurements [21, Ch. 2.2], we focus here on the most common one: *projective measurement in the computational basis*, which is used by default in standard quantum circuits and all visualization studies surveyed. Upon such measurement, a qubit in superposition collapses to a computational basis state with a certain probability. The *measurement probability* of collapsing to a specific basis state $|i\rangle$, given a density matrix ρ of the quantum state, is calculated as

$$P(|i\rangle) = \langle i|\rho|i\rangle = \sum_j p_j \langle i|\psi_j\rangle\langle\psi_j|i\rangle = \sum_j p_j |\langle i|\psi_j\rangle|^2. \tag{3}$$

For the special case where the measured quantum state is a pure state, as in Eq.1, the probability of collapsing to $|i\rangle$ is simply $|\alpha_i|^2$, where α_i is the corresponding amplitude in the state vector.

We can present gates and measurements in circuit diagrams for clarity. By convention, the input of qubits is on the left and the output is on the right. For each qubit, there is a horizontal wire connecting the input and output. On a wire, a sequence of gates is placed, acting on a qubit. Additionally, all qubits are numbered from top to bottom and initialized as $|0\rangle$.

Fig. 2 shows a simple quantum circuit with two qubits. The first qubit is flipped by the X gate. Subsequently, a CNOT gate is applied with the first qubit as the control qubit and the second qubit as the target qubit. In the end, two measurements are performed on the two qubits. Therefore, the state transition is $|00\rangle \to |01\rangle \to |11\rangle$ and the measurement outcomes are $|11\rangle$. The ket notation follows the little-endian ordering, where the rightmost digit denotes the state of the first qubit and the leftmost digit denotes the state of the second qubit. This ordering follows the Qiskit convention; other references may use the opposite ordering.

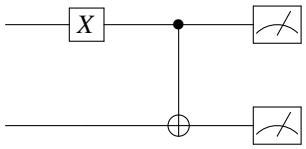


Fig. 2. A simple quantum circuit diagram that consists of an X gate, a CNOT gate, and two measurements.

In circuit diagrams, gates and measurements can form coarser-grained abstractions: blocks and layers. A block can contain multiple gates acting on different qubits. A layer can contain multiple blocks and gates that collectively act on all qubits. An example of a block and a layer is shown in Fig. 3.

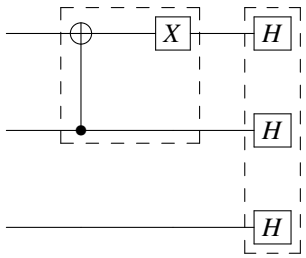


Fig. 3. A quantum circuit of three qubits containing one block of a CNOT and X gates and one layer of three Hadamard gates.

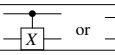
Such circuit diagrams provide a clear visual representation of a quantum computation process. Therefore, QC researchers and educators often use them to explain the principles of quantum computation, design and analyze quantum algorithms, and visualize circuit models.

The circuits described so far, which run a sequence of gates from left to right unconditionally, are called *static circuits*. *Dynamic circuits*, in contrast, can run or skip parts of a quantum circuit conditionally based on the measurement outcomes of preceding operations or other control signals, paralleling the dynamic control flow in classical computing.

Besides the static–dynamic distinction, there is also the distinction between a circuit model and its physical implementation. *Circuit models* describe the abstract operations of quantum gates and measurements, regardless of the constraints and optimizations in physical circuit implementations. For example, quantum processors of different physical architectures (e.g., superconducting [22], ion trap [23], and neutral atom [24]) may have different native gates that can be executed directly on device. For a non-native gate in a circuit model,

TABLE III.1
COMMON QUANTUM GATES AND THEIR REPRESENTATIONS. FOR EACH GATE, THE TABLE SHOWS ITS STANDARD ABBREVIATION, UNITARY MATRIX, AND CIRCUIT-DIAGRAM SYMBOL, LINKING THE MATHEMATICAL OPERATORS USED TO TRANSFORM QUANTUM STATES WITH THE VISUAL NOTATION USED TO DESCRIBE QUANTUM CIRCUITS.

Quantum Gate	Pauli-X (X)	Pauli-Y (Y)	Pauli-Z (Z)	Hadamard (H)	Controlled-Z (CZ)
Mathematical Form	$\mathrm{X}=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$	$\mathrm{Y}=\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$	$\mathrm{Z}=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$	$\mathrm{H}=\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	$\mathrm{CZ}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\end{bmatrix}$
Graphical Representation					

Quantum Gate	Rotation-X (RX(θ))	Rotation-Y (RY(θ))	Rotation-Z (RZ(θ))	SWAP	Controlled-NOT (CNOT, CX)
Mathematical Form	$\mathrm{RX}(\theta)=\begin{bmatrix}\cos\frac{\theta}{2} & -i\sin\frac{\theta}{2}\\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2}\end{bmatrix}$	$\mathrm{RY}(\theta)=\begin{bmatrix}\cos\frac{\theta}{2} & -\sin\frac{\theta}{2}\\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2}\end{bmatrix}$	$\mathrm{RZ}(\theta)=\begin{bmatrix}e^{-i\theta/2} & 0\\ 0 & e^{i\theta/2}\end{bmatrix}$	$\mathrm{SWAP}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$	$\mathrm{CNOT}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$
Graphical Representation					

it must be decomposed into a sequence of native gates before being executed on a real quantum processor. Another example of physical constraints is the connectivity of qubits. In some architectures (e.g., superconducting), qubits are connected to one another via wires, so they have only local interactions with neighboring qubits. Therefore, if two qubits in a circuit model need to interact with each other using a gate, intermediate qubit operations may be required to connect them.

These examples are only part of the problem. Physical constraints, implementation requirements, and optimization goals all shape the translation from a circuit model to a physical circuit implementation. This translation is called *transpilation*, and it is a non-trivial optimization problem.

C. Quantum Algorithms

Beyond the implementation details introduced in Sections III-A and III-B, quantum algorithms are another core component of QC. A quantum algorithm is a step-by-step procedure designed to solve a computational problem by using quantum-mechanical phenomena such as superposition and entanglement [25].

Like classical algorithms, quantum algorithms are abstract problem-solving strategies; as such, they should not be confused with any single implementation model. A common misconception among newcomers is that *quantum circuits are equivalent to quantum algorithms*, likely because algorithms are typically introduced through circuit diagrams. However, this equivalence does not hold. The classical parallel is that classical algorithms are distinct from their circuit implementations. Under the model based on quantum gates and circuits, the quantum operations in an algorithm are often expressed as one or more quantum circuits. However, a quantum algorithm is not necessarily identical to a single circuit. It may also include classical preprocessing, postprocessing, adaptive control, or repeated calls to quantum subroutines. A quantum algorithm may be implemented using other models as well, such as adiabatic model [26], measurement-based model [27] and continuous-variable model [28].

Due to the vast space and abstract nature of quantum algorithms, we focus here on two examples to illustrate the distinction between quantum algorithms and quantum circuits.

Shor's Algorithm: This algorithm is used to factor large integers, which is a difficult problem for classical computers, as there are no known polynomial-time classical algorithms for it. This quantum algorithm serves as a prime example of a hybrid algorithm, where the quantum circuit is merely a subroutine invoked in a loop. The algorithm uses a quantum computer solely to solve the order-finding problem (i.e., finding the period r of a function), while the actual factorization logic is handled classically. The high-level procedure is summarized in Algorithm 1. The *quantum circuit* (Lines 5–8), shown in Fig. 4, is responsible only for estimating the period. The Oracle in Shor's algorithm refers to the controlled modular exponentiation gates $U_{a^{2^k}}$, each computing $a^{2^k} \bmod N$ on the target register, controlled by one qubit of the input register. These gates collectively implement the function $f(x) = a^x \pmod N$. The critical logic for converting that period into factors (Lines 10–14) and the control flow (Lines 2, 11, 12, 14) are entirely classical. Visualizing only the circuit diagram would completely omit the probabilistic retry logic and the number-theoretic reasoning that define the algorithm's success. Visualization researchers need to look beyond the gate sequence to represent the full algorithmic workflow, including classical control, probabilistic branching, and the distinct roles of classical versus quantum modules.

Quantum Machine Learning (QML): QML further illustrates the hybrid nature of some quantum algorithms. QML algorithms typically use *parametric quantum circuits*, where the rotation angles of parametric gates (e.g., $\mathrm{RX}(\theta)$, $\mathrm{RY}(\theta)$, $\mathrm{RZ}(\theta)$ in Table III.1) serve as trainable parameters, analogous to the weights of a classical neural network. A classical optimizer iteratively adjusts these parameters based on measurement outcomes to minimize a cost function. This optimization loop runs entirely on classical hardware, whereas only the circuit evaluation is performed on a quantum computer. Thus, the "algorithm" encompasses not just the quantum circuit but the entire classical-quantum feedback loop, providing another

Algorithm 1 High-level workflow of Shor's Algorithm for factoring an integer N. The pseudocode separates the quantum order-finding subroutine from the surrounding classical preprocessing, postprocessing, and retry logic, illustrating that a quantum algorithm is not equivalent to a single circuit.

1: **Input:** A composite odd integer N to be factored.
2: Pick a random integer a where $1 < a < N$.
3: Compute $\gcd(a,N)$. If $\gcd(a,N) \neq 1$, return it as a factor.
4: **Quantum Order Finding:**
5: Construct a quantum circuit, an Oracle, for the function $f(x) = a^x \pmod N$.
6: Initialize qubits in superposition and apply the Oracle.
7: Apply Quantum Fourier Transform to create interference.
8: Measure to obtain a phase estimate s/r.
9: **Classical Post-processing:**
10: Use the continued fractions expansion algorithm on s/r to extract the period r.
11: If r is odd, go to Line 2.
12: If $a^{r/2} \equiv -1 \pmod N$, go to Line 2.
13: Compute $p = \gcd(a^{r/2}-1, N)$ and $q = \gcd(a^{r/2}+1, N)$.
14: If p or q is non-trivial (not 1 or N), **return** them. Otherwise, go to Line 2.

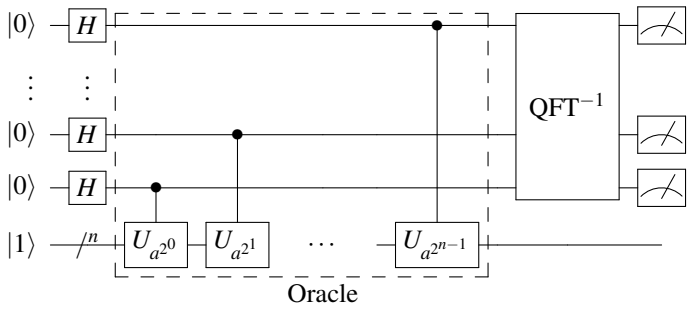


Fig. 4. Quantum circuit for Shor's order-finding subroutine. The top n qubits form the input register, initialized in superposition by Hadamard gates. The bottom n qubits store the target state. The controlled-$U_{a^{2^k}}$ gates (the Oracle) implement modular exponentiation. The inverse QFT extracts the phase, which is then measured.

clear example that quantum circuits alone do not constitute the full algorithm.

For a broader landscape of quantum algorithms and more rigorous discussions, we refer to the corresponding sections in various textbooks (e.g., [21], [29]–[32]) and surveys (e.g., [2], [3], [25], [33]). Two surveys are particularly relevant: Dalzell and McArdle et al. [33] provided a comprehensive and practical overview of the application domains and end-to-end complexities of quantum algorithms; Arnault et al. [2] provided a systematic classification of quantum algorithms based on the types of primitives, mathematical classes, and application domains. They also constructed a dependency graph of quantum algorithms to help understand the relationships between them.

D. Quantum Programming

To implement quantum algorithms and run them on quantum computers or simulators, quantum SDKs (e.g., Qiskit [7], Pennylane [9]) provide programmable interfaces. At its core, quantum programming involves constructing quantum circuits programmatically and executing them on various backends. Modern SDKs also support hybrid quantum-classical programming, where mid-circuit measurements and classical control flows are integrated with quantum operations. This enables dynamic circuits, as introduced in Section III-B, where the execution path depends on intermediate measurement outcomes.

Listing 1 provides a simple example of quantum programming with Qiskit [7]. It constructs the circuit shown in Fig. 2. The circuit is then executed on a *simulator*, which simulates ideal quantum computers and provides the resulting quantum state without any noise. The output state vector is $[0,0,0,1]^\top$, corresponding to $|11\rangle$ in the computational basis.

Listing 1. A simple Qiskit program that builds and simulates the circuit in Fig. 2.

```
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator

# Construct the circuit
qc = QuantumCircuit(2)
qc.x(0)       # X gate on qubit 0
qc.cx(0, 1)   # CNOT: qubit 0 controls qubit 1
qc.measure_all() # Measurements on all qubits

# Simulate and get the state vector
simulator = AerSimulator(method='statevector')
qc.save_statevector()
result = simulator.run(qc).result()
statevector = result.get_statevector()
# Output: [0, 0, 0, 1] corresponding to |11>
```

As shown in Listing 1, SDKs include built-in simulators that compute exact quantum states. These simulators, even though they cannot substitute for real quantum computers, are essential for VIS4QC research that relies on quantum states for several reasons:

1) **Noise-free Reference:** Current quantum computers, due to physical constraints and hardware limitations, introduce significant noise that obscures ideal behavior. Simulators provide clean and ideal results that serve as the ground truth for visualizations.
2) **Access to Intermediate States:** On real quantum hardware, measurement causes state collapse, making it impossible to observe intermediate quantum states during computation. Simulators provide a convenient way to obtain complete state vectors and density matrices at any point in the circuit, which is necessary for many surveyed visualizations that help understand the workings of quantum circuits and algorithms.
3) **Accessibility:** Real quantum computers have limited availability and long queue times. Simulators enable rapid prototyping and experimentation for such research.

Beyond simulation, most SDKs also provide built-in visualizations for circuits and quantum states, such as circuit diagrams, histograms of measurement outcomes, and Bloch spheres. These visualizations are discussed in detail in subsequent sections.

E. Noisy Quantum Computation

Current quantum computers (e.g., IBM's Heron[1] and Google's Willow[2]) have tens to hundreds of qubits, but cannot yet perform *fault-tolerant* computation. They share several key limitations [34]: high error rates in gates and measurements, short coherence times (i.e., the duration for which qubits maintain their quantum properties), and limited qubit connectivity. These imperfections result in noisy outputs that deviate from ideal quantum behavior. Despite the limitations, recent experiments have demonstrated quantum utility [6], showing that they can already produce results that are difficult to replicate classically. Nevertheless, noise remains the defining challenge in QC until fault-tolerant quantum computing becomes practical.

As a brief introduction here, we describe two basic types of noise and errors that commonly occur at the qubit level: Bit Flip and Phase Flip. For more types of noise and errors and more details, please refer to Chapter 8 of the book by Nielsen and Chuang [21].

Bit Flip: These errors occur when a qubit spontaneously changes from $|0\rangle$ to $|1\rangle$ or vice versa, analogous to classical bit flips. An example is $|0\rangle \rightarrow |1\rangle$ or $|1\rangle \rightarrow |0\rangle$. Another example is a three-qubit state $\phi = \alpha|000\rangle + \beta|111\rangle \rightarrow \phi' = \alpha|001\rangle + \beta|110\rangle$, due to a bit flip error on the third qubit. This directly changes the possible outcomes after measurement and their probabilities, leading to incorrect results.

Phase Flip: These errors alter the relative phase between the basis states of a qubit without changing the measurement probabilities in the computational basis. For example, $\phi = \alpha|000\rangle + \beta|111\rangle \rightarrow \phi' = \alpha|000\rangle - \beta|111\rangle$. Although the measurement probabilities of $|000\rangle$ and $|111\rangle$ remain unchanged (i.e., $|\alpha|^2$ and $|\beta|^2$ respectively), the phase difference between them is altered, which corrupts the quantum state. The corrupted quantum state may lead to incorrect outcomes when used in subsequent computations (e.g., algorithms that use the phase kickback effect[3]).

Solving such errors is one of the major challenges in QC. Two complementary approaches have emerged to address or mitigate this challenge: Quantum Error Correction (QEC) and Quantum Error Mitigation (QEM).

QEC aims to fully correct errors in noisy qubits and gates. QEC for qubits is accomplished by encoding logical qubits into many noisy qubits in various schemes (i.e., QEC codes). After encoding, a group of noisy qubits is entangled together to form a logical qubit. Measurements on a subset of these noisy qubits can detect errors without collapsing the logical qubit state. Based on the measurement results, additional quantum gates can be applied to correct the errors. QEC enables accurate single-shot computation but requires qubit counts and error rates beyond what current noisy quantum hardware can achieve. For more details, please refer to the introductory article by Roffe [35] and the online course by Google[4].

In contrast to QEC, QEM works within the constraints of current noisy quantum computers by using techniques such as zero-noise extrapolation [36] and probabilistic error cancellation [37]. These techniques modify quantum circuits so that errors have less impact on the expectation values estimated over many executions. While QEM cannot correct individual runs, it improves the accuracy of these statistical estimates, making it practical for applications on current hardware.

Understanding noise characteristics is essential for VIS4QC research. Since noise is pervasive in current quantum systems, several visualization systems focus on helping users understand, monitor, and mitigate its effects. For example, visualizations (e.g., [38], [39]) of qubit error rates, coherence times, and gate fidelities enable users to select less noisy hardware configurations. Noise-aware circuit visualizations (e.g., [40]) help developers understand how errors propagate through their circuits. As we survey in subsequent sections, several existing VIS4QC studies directly help researchers mitigate the challenges posed by current noisy quantum computers.

IV. Methodology

To map the research space of VIS4QC and diagnose its coverage, we analyzed 69 core PTs that are closely related to VIS4QC in the fields of QC, VIS, CG, HCI, and SE. Each of these PTs is coded based on the usage and purpose of visualizations and the abstraction level of QC, which are detailed in Sections V and VI. The third dimension of our taxonomy, *how* QC subjects are visualized (Section VII), is developed through an analytical review of the visualization techniques across the coded PTs. At different stages, we used Large Language Model (LLM) agents for data collection and summarization (detailed below), with manual verification at each step. All analysis and writing in this paper are human-authored.

A. Survey Scope

We began our analysis by collecting a database of potentially relevant PTs. To achieve extensive coverage, we used three methods in our collection pipeline, as shown in Fig. 5.

Systematic Search: We first selected the main journals and conferences in the fields of QC, VIS, CG, HCI, and SE based on our collaborators' suggestions and relevance to VIS4QC. The selected venues and their statistics are shown in Table IV.1. Subsequently, we collected all research publications from the selected venues spanning 2015 to 2025 using official search and download tools. We also collected the tools referred in these publications. The time range was chosen because: (1) this period covers the majority of relevant VIS4QC research; (2) as QC research is evolving rapidly, recent studies are likely to be more relevant to current research; and (3) we will cover relevant research prior to 2015 using the following two methods. Some venues were established after 2015, so we collected publications from these venues starting from their

[1] https://www.ibm.com/quantum/hardware

[2] https://blog.google/innovation-and-ai/technology/research/google-willow-quantum-chip/

[3] https://en.wikipedia.org/wiki/Phase_kickback

[4] https://www.coursera.org/learn/quantum-error-correction

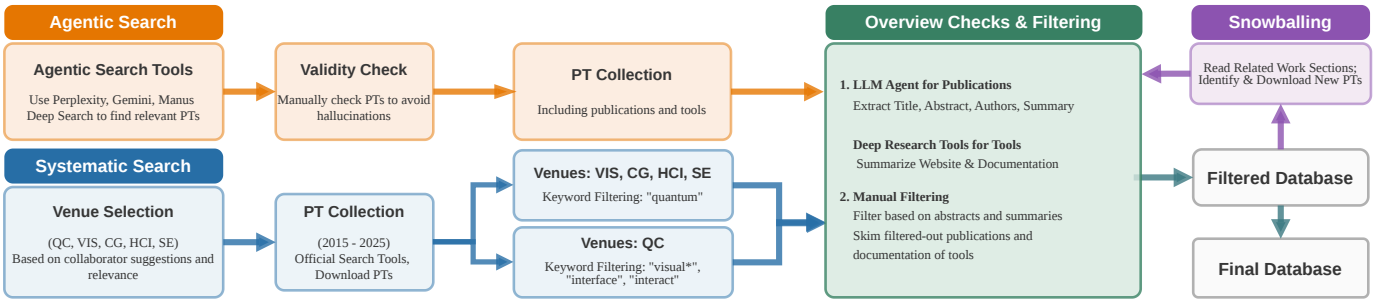


Fig. 5. Pipeline for collecting and filtering PTs. Candidate PTs were collected through systematic venue search, agentic search, and recursive snowballing. Systematic-search results were keyword-filtered by venue type, while agentic-search results were manually checked before collection. All candidates then passed through overview generation and manual filtering before entering the filtered database, which was further expanded through snowballing and finalized as the core collection.

TABLE IV.1
THE LIST OF VENUES INCLUDED IN OUR SYSTEMATIC SEARCH, SHOWING THE TIME SPAN AND THE TOTAL NUMBER OF PAPERS RETRIEVED FROM EACH VENUE. REFER TO TABLE A.1 IN THE APPENDIX FOR VENUE FULL NAMES.

Venue	Time Span	Total Papers
VIS & CG		
IEEE CG&A	2015–2025	1236
TVCG and IEEE VIS	2015–2025	4689
PacificVis	2015–2025	382
EuroVis	2015–2025	2806
HCI		
ACM CHI	2015–2025	8276
ACM UIST	2015–2025	1162
QC		
IEEE QCE	2020–2025	972
IEEE QSW	2022–2025	82
PRX Quantum	2020–2025	236
QIP	2015–2025	4038
Quantum	2017–2025	1898
ACM TQC	2020–2025	143
SE		
Nat. Comput. Sci.	2015–2025	373
ACM TOPLAS	2015–2025	202
ACM TOSEM	2015–2025	1145
IEEE TSE	2015–2025	1540
ESEC/FSE	2015–2025	2098
IEEE/ACM ASE	2015–2025	1991
ACM ICSE	2015–2025	8397
Total		**41666**

respective inception dates. The full collection from the selected venues contains 41666 publications. Due to the large number of publications in the full collection, we performed an initial screening to remove publications that are not related to QC or VIS. We searched for keywords in the title, abstract, and full text of each publication. For publications of VIS, HCI, and SE, our search keyword is "quantum", while for those of QC, our search keywords are "visual*", "interface", and "interact*", where "*" is a wildcard character. We collected the publications that matched our search strategy, which ensures broad coverage of relevant research despite possible false positives.

Agentic Search: We used existing agentic search tools to search for and collect relevant PTs on the Internet. Specifically, we used the deep research tools from Perplexity[5], Gemini[6], ChatGPT[7], and Manus[8] by prompting them to search for academic research related to VIS4QC. This complements the systematic search and broadens coverage. After receiving the list of search results, we manually checked their validity and downloaded the publications to avoid hallucination.

Snowballing: After our systematic and agentic search, we had an initial database of relevant PTs. Subsequently, for each publication, we manually read the section on related work, identified additional relevant PTs, and downloaded them. We also recursively searched for relevant references in the newly collected PTs until no more relevant PTs were found. We did the same snowballing for tools by reading their documentation and finding related tools from there.

Overview Checks and Criteria: As shown in Fig. 5, before collecting PTs into our database, we performed overview checks to further exclude irrelevant ones. For publications, we developed an LLM agent based on Gemini 2.5 Flash [41] to extract the title, abstract, authors, and summary of each in an overview document for further filtering. The agent is prompted to generate a summary of each paper that provides more details and key insights than the title and abstract. For the prompt of this agent, please refer to Listing 2 in the Appendix. For tools, we prompted the aforementioned deep research tools to summarize the websites and documentation into overview documents, focusing on visualization-related features. The prompt template we used is Listing 3. Based on the overview documents, we manually filtered out PTs irrelevant to VIS4QC by skimming through the publications and documentation of tools. For each, to determine its relevance, we validated three terms: **VIS**, **QC**, and **4 (for)**. To validate the term **VIS**, one of PTs should at least include visual presentations (i.e., graphical interfaces, visual interactions, and plotting), and the visual presentations are a significant part of the research. Here, we employed a broad definition of visualization since VIS4QC is an emerging field and its existing research is sparse. This criterion is permissive in terms of the forms of visualization, but it excludes studies that merely use existing

[5]https://www.perplexity.ai/
[6]https://gemini.google.com/
[7]https://chatgpt.com/
[8]https://manus.im/

visualizations as supplementary tools in their research. For example, *Quantum Game Club* [42] uses techniques of visual and interactive learning, incorporating a few visualizations in games. However, the main focus of this work is the proposed educational model and its validity, so it does not qualify as VIS4QC research. In contrast, *Intuit* [43] qualifies as VIS4QC research, because visualization is a crucial means for it to serve its main purpose, which is educational. To validate the term **QC**, it should help readers and users understand and/or solve the challenges and concepts that involve using quantum physics to perform *computation*. This criterion distinguishes VIS4QC studies from those that visualize various phenomena in quantum physics. For example, the work by Fickler et al. [44] perform real-time imaging on quantum entanglement, presenting graphical results from their experiments. Despite quantum entanglement being an important concept and phenomenon used in QC, this work is not a VIS4QC study, as its focus is not on quantum computation. Another example is the glyph design by Zhao et al. [45]. It is useful for visualizing the data of quantum spins, but it is not a relevant study, as it mainly focuses on helping users understand the data of quantum spins. To validate the term **4 (for)**, it should contribute a novel visualization to QC use cases. This excludes studies that use QC to process data for visual tasks (e.g., [46], and [47]).

Additionally, to avoid potential hallucinations in the generation of overview documents, we skimmed through the filtered-out publications with a focus on the three terms before removing them from our collection. We report that no publications were misclassified as irrelevant, and no issues of hallucination were found in our overview checks.

These complementary methods provide comprehensive coverage of relevant VIS4QC research. The final collection contains 69 PTs.

B. Coding of PTs

Our analysis proceeded in two stages. In the first stage, we read all overview documents of the core PTs, tagging them with applications, purposes, quantum concepts, and domains. We did not specifically focus on the types or patterns of visualizations, as existing related studies are sparse and the taxonomies of visualization are diverse. Our goal was to answer two key questions that define the problem space: *"Why are visualizations created, i.e., what are the target applications, use cases, and purposes?"* and *"What subjects of QC are the existing studies focusing on?"*

Two co-authors independently coded all PTs with their own codes. Additionally, to ensure consistency in quantum terminologies, they referred to the glossary in Nielsen and Chuang [21]. They then discussed their codes and categorization to reach an initial consensus. Subsequently, during weekly discussions, we further iterated on the codes and categorization with our expert collaborators over multiple rounds until a final consensus was reached.

In the second stage, the two authors coded the core PTs again to validate the final taxonomy. The coding categorizes PTs along the two problem-space dimensions: *Usage and Purpose of Visualizations* (*why*; Section V) and *Abstraction Levels of QC* (*what*; Section VI). The third dimension, which surveys the solution space, namely *how* existing work visualizes QC subjects (Section VII), is developed by analytically reviewing the visualization techniques employed across the coded PTs.

The codes of all core PTs are summarized in Table IV.2. Some PTs may be tagged with multiple codes, as some studies provide comprehensive solutions to multi-faceted problems. We further distinguish between PTs that provide only basic and conventional visualizations versus those that contribute novel or advanced visualization designs. In Table IV.2, triangles indicate PTs that provide only basic visualizations, including Bloch spheres and bar charts for qubits and states, as well as standard circuit diagrams for circuits. As QEC requires distinct QC knowledge and visualizations, we mark related PTs with a special round circle. Figure 13 presents the distribution of PTs by abstraction levels and purpose. Figure 14 illustrates the temporal distribution of surveyed PTs.

V. Usage and Purpose of Visualizations

Here, we discuss the first dimension of our taxonomy: *why* a visualization is created. Visualizations in QC serve distinct purposes, namely **learning & interpretation** and **productivity & utility**. Visualizations for **learning and interpretation** teach QC concepts or provide novel visual representations that offer fresh perspectives on established concepts. They generally require little prior QC knowledge and are common in education and research contexts. Visualizations for **productivity and utility** help users accomplish tasks such as debugging or machine selection more efficiently and can further improve task outcomes, such as lower training loss or higher model accuracy. They assume familiarity with the specific tasks involved. As shown in Figure 14, visualizations for learning and interpretation have accumulated a large body of research over decades since the Bloch sphere [10]. There are limited studies on visualizations for productivity and utility before 2023, and most of them are developed after 2023. Table V.1 summarizes two main categories along this dimension. We detail the characteristics of each category in the following sections.

Note that a work may serve both purposes simultaneously. This is especially true for books and SDKs, as they often offer visualizations to aid understanding while their advanced topics or features target experienced users.

A. Learning and Interpretation of QC Knowledge

Visualizations in this category serve two distinct goals. The first is to *teach* QC subjects to newcomers, helping students understand quantum concepts and algorithms through visual explanations. The second is to *interpret* QC subjects visually, providing researchers with new graphical representations that offer fresh perspectives on established concepts. Both goals target a broad audience and generally require little prior knowledge of QC, though interpretation-oriented work typically assumes basic familiarity with the target subject.

Teach QC Subjects: In quantum education, teachers use visualizations to teach new quantum concepts and algorithms to students. For example, *Intuit* [43] uses visual analogies

TABLE IV.2
CODINGS OF PTS. TRIANGLES ▲ INDICATE ONE WORK HAS OR USES BASIC AND CONVENTIONAL VISUALIZATIONS. ROUND CIRCLES ● INDICATE IT IS RELATED TO QEC. QS: QUBITS & STATES, GC: GATES & CIRCUITS, MS: MACHINES & SYSTEMS, AD: ALGORITHM DEVELOPMENT, LI: LEARNING & INTERPRETATION, PU: PRODUCTIVITY & UTILITY. INDUSTRY TOOLS ARE HIGHLIGHTED IN INDIGO AT THE BOTTOM. REFER TO TABLE A.1 FOR VENUE FULL NAMES.

PTs	Venue	Year	QS	GC	MS	AD	LI	PU
Bloch Sphere [10]	Phys. Rev.	1946	■				■	
Walck et al. [48]	Eur. J. Phys.	2001	■				■	
Decision Diagrams for QC [49]	ASP-DAC	2003	■	■		■	■	
Karafyllidis et al. [50]	QIP	2003				■	■	
ZX Calculus [51]	ICALP	2008	■	■		■	■	■
Altepeter et al. [52]	CLEO/IQEC	2009	■				■	
Majorana Representation [53]	Phys. Scr.	2010	■				■	
Fractal Representation [54], [55]	ICQNM & SysMea	2011 & 2012	■				■	
Tahan et al. [56]	APS March Meeting	2015	■	▲			■	
Chernega et al. [57]	J. Russ. Laser Res.	2017	■				■	
QuaSim [58]	IEEE EIT	2017				■	■	
ShorVis [59]	ICVRV	2017				■	■	
QuFlow [60]	SciVis	2018		■			■	
Rainbow Boxes [61]	GRAPP	2019	■				■	
GraphStateVis [62]	IEEE QCE	2021	■				■	
Arawjo et al. [63]	ACM UIST	2022		■	■		■	■
Jordon et al. [64]	IEEE QCE	2022				■	■	
Suau et al. [65]	IEEE QCE	2022	■				■	■
qprof [66]	ACM TQC	2023		■				■
QWalkVis [67]	IEEE QCE	2023				■	■	
Quantivine [68]	TVCG	2023		■				■
Senapati et al. [69]	QCCC	2023	■				■	
VACSEN [38]	TVCG	2023	■	■	■			■
VENUS [70]	CGF	2023	■				■	
Bley et al. [71]	Phys. Rev. Res.	2024	■				■	
Chou et al. [72]	IEEE QCE	2024	■			■	■	
CircInspect [73]	IEEE QCE	2024		■				■
HammingVis [40]	Graph. Models	2024	■				■	■
Kim et al. [74]	IEEE QCE	2024		■	■			■
QGrover [75]	IEEE QCE	2024				■	■	
QML Playground [76]	IEEE CG&A	2024	■			■	■	
QNotation [77]	IEEE QCE	2024	▲	▲			■	
QuantumEyes [78]	TVCG	2024	■	■			■	
QubiCSV [79]	Sci. Rep.	2024			■			■
QVis [39], [80]	IEEE QCE & ISVLSI	2023 & 2024	■		■			■
Seidel et al. [81]	arXiv	2024	■			■	■	
VIOLET [82]	TVCG	2024	■			■	■	
BEADS [83]	New J. Phys.	2025	■	■			■	
Intuit [43]	ACM CHI	2025	■	■			■	
Kim et al. [13]	ACM CHI	2025	■	■	■			■
MuqcsCraft [84]	arXiv	2025	■	■			■	
QCanvas [85]	IEEE QSW	2025	▲	▲			■	■
Patel et al. [86]	IJAMRS	2025	■			■	■	
Quantum Image Visualizer [87]	arXiv	2025				■		■
QuRAFT [88]	TVCG	2025	▲	▲		■	■	
state-o-gram [89]	arXiv	2025	■				■	
Sanderson [90]		2025	■			■	■	
Scruby [91]	arXiv	2025		●			■	
Senapati et al. [92]	IEEE QSW	2025	■					■
XQAI-Eyes [93]	arXiv	2025	■	■			■	■
QuTiP [94]–[96]	CPC & Phys. Rep.	2012[c]	■					■
pyQuil [97]	arXiv	2016		▲				■
Cirq [8]	Zenodo	2018[c]	▲	▲	■			■
Braket [98]		2020[c]	▲	▲				■
qecsim [99]	PhD Thesis	2020		●				■
TKet [100]	QST	2020		■				■
Qibo [101]	QST	2021		▲				■
Stim [102]	Quantum	2021		●				■
Pennylane [9]	arXiv	2022		■			■	■
qBraid [103]	Zenodo	2022[c]	▲	▲				■
Cuda Q [104]	DAC	2023	▲	▲				■
Q# [105]	EPTCS	2023		▲				■
Quantum Code Visualizer & PanQEC [106]	PRX Quantum	2023		●				■
Bloqade [107]	arXiv	2024	▲	▲				■
Qadence [108]	arXiv	2024		▲				■
Qiskit [7]	arXiv	2024	■	■	■		■	■
Qrisp [109][b]	arXiv	2024	■	▲		■		■
Qualtran [110]	arXiv	2024		■				■
Triple Alpha [111][a]		2025			■			■

[a] Due to limited access to it, we only checked the website of Triple Alpha
[b] Visualization limited to those in [81]
[c] Date of first announcement or release

to help students understand basic quantum concepts (e.g., superposition, measurement, and gates). *QNotation* familiarizes learners with different notations used in QC. Decision diagrams [112] often serve as a pedagogical tool, representing quantum states and operations as node-link trees that make recursive structures explicit. Arawjo et al. [63] develop a drawing-based interface for students to learn quantum programming. Various visualizations have also been developed to help learners understand textbook quantum algorithms [50], [59], [64], [72], [75], [86], [88], [90]. Quantum games [18]–[20], [56], [58], [113]–[116] serve the same teaching purpose but emphasize engagement and motivation, embedding visualizations within game mechanics to sustain learner interest. Because the goal is to build conceptual understanding, these visualizations typically focus on small-scale quantum systems where the underlying phenomena remain tractable enough to explain visually.

Interpret QC Subjects Visually: Beyond teaching, some visualizations offer researchers graphical representations that reveal structures not apparent from mathematical formalism alone. These studies often stem from physics and QC research. One foundational example is the Bloch Sphere [10] (Fig. 6), which gives a single-qubit state an intuitive geometric identity as a point on a unit sphere, making abstract operations such as rotations and phase shifts directly visible. Since then, numerous physicists and mathematicians have developed representations that extend this geometric intuition to richer quantum phenomena. For example, the fractal representations proposed by Galambos et al. [54], [55] reveal the internal structure of multi-qubit systems by exploiting self-similarity: each qubit appears as a building block of the fractal, making it possible to simultaneously see individual qubit properties and their entanglement, which the Bloch sphere alone cannot show. Chernega et al. [57] offer a complementary perspective on single-qubit states by mapping them onto the vertices of a triangle, establishing a one-to-one correspondence with points inside the Bloch sphere. This reframing makes uncertainty relations for measurement probabilities geometrically visible as constraints on the triangle's area. The Majorana representation [53] extends the Bloch sphere picture to multi-qubit systems by representing an N-qubit state as $2^N - 1$ points on the sphere. This representation preserves the geometric language while revealing how the points' configuration distinguishes entangled from separable states. Despite their potential applications in education, these visualizations are mainly used by physics and QC researchers to interpret properties of QC subjects. Compared to purely educational visualizations, they require a basic understanding of the target QC subject. For example, to use the fractal representations [54], [55], users need to understand the measurement probabilities.

B. Productivity and Utility

Visualizations in this category target experienced users who already possess domain knowledge and need to work more efficiently or achieve better task outcomes. Most existing PTs focus on productivity; that is, helping users accomplish a task

TABLE V.1
USAGE AND PURPOSE CATEGORIES IN VIS4QC. THE TABLE CONTRASTS VISUALIZATIONS FOR LEARNING AND INTERPRETATION WITH THOSE FOR PRODUCTIVITY AND UTILITY, SUMMARIZING THEIR GOALS, APPLICATION DOMAINS, ASSUMED PREREQUISITE KNOWLEDGE, AND REPRESENTATIVE EXAMPLES.

	Goals	Application Domains	Prerequisite Knowledge	Representative Examples
Learning and Interpretation	• To teach QC subjects • Interpret QC subjects visually with new graphical representations	• Education • Physics and QC Research	None or Basics	• *Intuit* [43] presents analogy-based visualizations to teach basic QC concepts. • The Bloch Sphere [10] for single-qubit states. • *Majorana Representation* [53] interprets multi-qubit states as points on the Bloch sphere.
Productivity and Utility	• To improve user productivity by presenting useful data visually, which can in turn lead to better task outcomes and performance	• Software Engineering • QC Research • System Management	Complete Understanding of Specific Tasks	• *VACSEN* [38] visualizes noise distributions, enabling users to quickly select less noisy quantum computers. • *XQAI-Eyes* [93] visualizes effects of quantum encoders and helps users select optimal encoders for QML models.

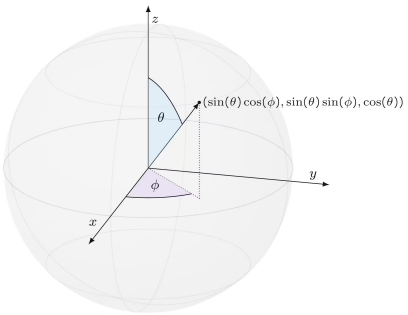


Fig. 6. Bloch Sphere. The illustration is from IBM Quantum Platform [117].

faster or with less effort. However, some also yield utility gains by improving the quality of the outcome itself.

Productivity: A number of QC SDKs (e.g., Qiskit [7], Cirq [8], Pennylane [9]) already help users program quantum circuits, debug, run classical simulations, and produce basic visualizations, reducing the manual effort required at each stage of the development workflow. The PTs surveyed here address productivity bottlenecks that SDKs leave unresolved. For example, selecting a quantum computer with a favorable noise profile is critical for computational fidelity. Yet, the noise landscape is complex and changes over time, forcing users into a time-consuming trial-and-error process. *VACSEN* [38] replaces this manual search with an interactive visual comparison of noise across machines and compiled circuits. *qprof* [66] helps users identify performance bottlenecks and optimize their implementations through visual profiling reports. *QubiCSV* [79] helps users rapidly analyze the vast amounts of data generated by qubit control systems. Beyond these specific bottlenecks, several recurring productivity needs span the QC development workflow. Verifying and debugging state transitions remain largely manual, as developers must trace how intermediate states change at each step. Understanding qubit connectivity topology is a prerequisite for efficient circuit compilation. Monitoring how hardware properties (e.g., coherence times and error rates) drift over time is necessary for selecting reliable execution windows. More broadly, Kim et al. [13] survey ten open-source QC tools, identify recurring usability barriers that slow developers down (e.g., the disconnect between high-level problem concepts and low-level circuit code, shallow machine-status dashboards), and propose interaction techniques (e.g., a conceptual program writer, a hardware dashboard) that streamline tasks from circuit authoring to result analysis.

Utility: Among these productivity-oriented PTs, some also improve *utility*; that is, they lead to task outcomes of higher quality rather than merely faster ones. In current work, however, utility gains arise as a consequence of productivity improvements rather than as an independent design objective. For example, *VACSEN* [38] not only speeds up the machine selection process but also enables users to discover less noisy configurations they might otherwise have overlooked, thereby improving computation fidelity. Similarly, *XQAI-Eyes* [93] helps users efficiently explore quantum encoder choices for QML tasks, and this broader exploration, in turn, yields encoders with lower training loss. Whether visualizations can be designed to optimize task outcomes directly, rather than as a byproduct of workflow efficiency, remains an open question (Section IX).

VI. ABSTRACTION LEVELS OF QC

This section addresses the second dimension of our taxonomy: *what* QC subjects are visualized. We first present our taxonomy based on the abstraction levels of QC, explain the rationales behind it, and then survey related studies in the following subsections.

Despite the large amount of research on QC, there is not yet a clear taxonomy of this broad field, except for divisions based on applications and hardware/software. To comprehensively map out the relevant QC landscape and inform future exploration in VIS4QC, we classify the VIS4QC research based on the abstraction levels of QC, dividing QC studies into two main categories: those concerning **Quantum Computer Implementation**, which focus on understanding, building, and optimizing quantum computers, and those concerning **Algorithm Development**, which focus on using quantum computation to solve problems in various use cases and domains. In

other words, the former concerns quantum computers, while the latter concerns applications of QC. Fig. 7 summarizes the abstraction levels of QC.

This taxonomy is based on our iterative discussions with our expert collaborators from both the QC and VIS communities. It balances coverage of existing VIS4QC studies, room for future exploration, and QC rigor. Instead of dividing QC studies into hardware and software, we focus on abstractions spanning concepts, software, and algorithms, which better suit visualization research. To avoid overlap between categories, we divide the abstractions into implementation details and algorithms, rather than classifying them simply by applications and use cases. We further divide implementation details and group them into secondary levels based on the building blocks of quantum computers. These building blocks (e.g., qubits, gates, and circuits) are commonly accepted abstractions in the QC community. The rationales behind the groups of building blocks are detailed in the following Sections VI-A1, VI-A2, and VI-A3. In contrast to the granularity of quantum computer implementations, we did not further subdivide algorithm development into secondary levels because (1) QC algorithms are emerging quickly, (2) the taxonomy of quantum algorithms (e.g., [2], [33]) is complex and evolving, and (3) existing VIS4QC studies related to quantum algorithms are sparse.

Note that although the abstraction levels in our taxonomy have clear separations, a VIS4QC study can span multiple levels, as multifaceted challenges are often addressed with a combination of different visualization techniques. The following subsections introduce each level and briefly survey relevant studies. Because Section VII (the third dimension: *how* to visualize) discusses many of the same studies from a visualization perspective, we focus here on defining the QC subjects and relevant examples and defer the details of the data and visualizations to that section.

A. Quantum Computer Implementation

A significant body of QC research focuses on implementing quantum computers. Most existing VIS4QC research also addresses this level, as shown in Figure 13. At this level, several key concepts further abstract the physical details of quantum computers at different granularities. We group these concepts and techniques into three secondary levels (Fig 7), which are discussed in turn below.

1) Qubits and States: As introduced in Section III-A, qubits abstract away the underlying quantum mechanical details that are irrelevant to quantum computation. We include key attributes of qubits, such as quantum states and statistics, in this secondary level.

Quantum States: Quantum states are the central data objects at this level. Existing quantum state visualization shows state vectors, density matrices and/or the attributes (e.g., measurement probabilities, phase, and amplitudes) of basis states. For example, the Bloch Sphere [10] visualizes the state vector of a single-qubit state, while the phase disk [118] (Fig. 10) visualizes the phase, the probability of basis state $|1\rangle$, and the purity of the state. For multi-qubit states, the exponential growth of the state space (2^N for N qubits) poses a a fundamental challenge of visualization scalability due to the limited screen space, so different studies target different aspects of a multi-qubit state. For example, *Rainbow Boxes* [61] visualizes measurement probabilities and relative phases of basis states. *VENUS* [70] focuses on measurement probabilities and their normalization condition (Fig. 11). *XQAI-Eyes* [93] targets the low-dimensional structure within the density matrix of a multi-qubit state.

Qubit Statistics: Beyond quantum states, the operational statistics of qubits, including purity, noise, and error rates, form another important class of data at this level. For example, Suau et al. [65] focuses on the quality of qubit implementations on a quantum computer. *HammingVis* [40] addresses error propagation across qubits. *QVis* [39], [80] targets the temporal evolution of qubit noise levels.

Following the criteria in Section IV-A, we exclude PTs that visualize qubit aspects not directly related to QC, such as low-level quantum mechanical properties.

2) Gates and Circuits: We establish this secondary level to represent the flow of quantum computation. Unlike the static representation of information in qubits, this level focuses on the dynamic manipulation of quantum information. It encompasses gates, measurements, and circuits, where gates and measurements serve as atomic operations, and circuits as composed programs. This level also includes state transitions that arise from circuit execution, analysis on specialized circuits (i.e., QML circuits), QEC techniques that operate at the circuit level (Section III-E), and operational metrics (e.g., gate fidelities and circuit depths). We discuss each below.

State Transitions: State transitions capture how quantum states change as gates are applied. The relevant data that has been visualized includes the intermediate basis-state parameters (e.g., probabilities and phases) at every step in a circuit's execution. For example, *QuFlow* [60] represents the complete parameter flow of all 2^N possible outputs through a circuit, showing how each gate transforms the probabilities and phases of every basis state. *CircInspect* [73] addresses both the circuit structure at multiple levels of abstraction (e.g., individual gates versus subroutine-level function blocks) and the program output at user-defined breakpoints, combining structural and state-transition data in a single representation.

Circuit Models and Physical Implementations: A circuit model represents a sequence of quantum operations (e.g., gates and measurements) applied to qubits. Because quantum gates are unitary, circuit structures have a distinctive property: qubit lines run strictly in parallel without fan-in or fan-out, unlike classical circuits, where wires can merge and branch. Circuit diagrams are the standard visual notation for these models. However, at scale, this notation becomes impractical: current quantum computers have tens to hundreds of qubits, and after transpilation, the physical circuit may contain hundreds of gates. These scalability challenges have motivated alternative representations. *Quantivine* [68] (Fig. 12) addresses the hierarchical component structure of large-scale circuits (e.g., nested sub-circuits grouped by function), repetitive gate patterns (e.g., vertical, horizontal, and diagonal repetitions), and contextual

		Included Items	Representative Examples
Quantum Computer Implementation	Qubits & States	• Quantum States: state vector, density matrix, basis states • Qubit Statistics: error rates, T1 and T2 times, etc.	• Vector field visualization by Suau et al. [107] • *VENUS* [88] • *HammingVis* [15]
	Gates & Circuits	• State Transitions • Circuit Models and Physical Implementations • Spectral Analysis of Circuits • Statistics of Gates and Circuits • QEC Techniques	• *QuFlow* [64] and *CircInspect* [53] • *Quantivine* [120] • Spectral analysis in PennyLane [6] • *VACSEN* [87] • PanQEC [41]
	Machines & Systems	• Machine Properties and System Data • Interaction with Quantum Computers	• *QVis* [14, 105] • *QubiCSV* [12] • QHCI research by Kim et al. [54]
Algorithm Development		• Quantum Algorithms • Graphical Tools and Systems • Graphical Languages for Algorithm Analysis	• *ShorVis* [109] and *QGrover* [76] • *QML Playground* [24] • *ZX-Calculus* [20–22, 58]

Fig. 7. Taxonomy of VIS4QC research by QC abstraction level. We separate work on quantum computer implementation from work on algorithm development. Implementation-focused work is further organized into qubits and states, gates and circuits, and machines and systems; each level lists the included subjects and representative examples.

information including qubit provenance, gate placement and parallelism, as well as qubit connectivity and entanglement. Additionally, Qiskit [7] and Qualtran [110] represent circuits as directed acyclic graphs (DAGs), exposing computational dependencies between gates that are implicit in standard circuit diagrams.

Spectral Analysis of Circuits: The spectral structure of QML circuits, i.e., the Fourier frequencies and coefficients that characterize a circuit's representational capacity, forms another subject at this level. Pennylane [9] implements the Fourier analysis proposed by Schuld et al. [119] and visualizes these spectral components.[9]

QEC Techniques: QEC techniques, such as stabilizer codes and surface codes (Section III-E), operate at the circuit level and introduce their own data structures (e.g., code lattices, stabilizer groups, and syndrome patterns). Dedicated visualization approaches remain sparse. Existing PTs include Stim [102] and its interactive editor Crumble for stabilizer circuits, PanQEC [106] for topological codes, qecsim [99] for Pauli operators on code lattices, and Scruby [91] for quantum product codes.

Statistics of Gates and Circuits: Operational statistics of gates and circuits, such as gate fidelities, circuit depths, and gate counts, form another class of data at this level. For example, *VACSEN* [38] focuses on circuit-level noise profiles across different quantum computers.

3) Machines and Systems: This level represents the highest layer of implementation, encompassing the physical machine, its supporting infrastructure, and the human interface to the system. We also include QHCI studies (e.g., [13], [74]) in this level, as they concern the interaction between humans and quantum systems as a whole.

[9]Pennylane documentation on Fourier Analysis

Machine Properties and System Data: Key machine-level data include qubit connectivity topology, overall system performance metrics, and their temporal evolution. For example, *QVis* [39], [80] covers the qubit connectivity topology of a quantum computer alongside characterization metrics (e.g., relaxation time T_1, decoherence time T_2, readout error rate) and their temporal trends, as well as inter-qubit similarity in terms of these metrics. *VACSEN* [38] covers similar noise attributes across multiple quantum computers, together with gate-level error rates and the connectivity between quantum gates. At the physical level, *QubiCSV* [79] is the only study that addresses calibration and control system data, including qubit calibration parameters (e.g., drive frequency, readout frequency), gate configurations (e.g., phase, amplitude, envelope), and characterization metrics (e.g., readout fidelity, randomized benchmarking infidelity, coherence times). Beyond *QubiCSV*, no other study addresses this type of physical-level system data.

Interaction with Quantum Computers: We include QHCI studies that address the interaction design between users and quantum computers. Arawjo et al. [63] address quantum circuit structure authoring, including gates, sub-circuit definitions, and qubit bundles. This is done through a pen-based notation that supports abstraction features such as custom gate definitions and recursive circuit composition within Jupyter notebooks. Kim et al. [13] survey existing QHCI studies and propose interface techniques that span the full QC development workflow. These techniques cover circuit composition (linking high-level problem concepts to gate-level circuits), machine selection (gate-qubit error rates and time-serial machine data), circuit optimization (logical versus physical circuits, fidelity information), and result analysis (measurement outcomes, problem-specific outputs, and uncertainty).

TABLE VI.1
SEQUENTIAL VISUALIZATIONS FOR TEXTBOOK QUANTUM ALGORITHMS. THE TABLE LISTS THE TARGET ALGORITHMS AND THE INTERMEDIATE QUANTITIES VISUALIZED DURING EXECUTION: PROBABILITIES (P), STATE VECTORS (SV), AND AMPLITUDES (A).

PTs	Algorithms	P	SV	A
Chou et al. [72]	Grover's Algorithm [120]	■	■	
Karafyllidis et al. [50]	Quantum Fourier Transform	■		
Patel et al. [86]	Grover's Algorithm [120]	■		
QGrover [75]	Grover's Algorithm [120]			■
QuRAFT [88]	Few-qubit Quantum Algorithms, including Grover's Algorithm [120]	■	■	
QWalkVis [64], [67]	Quantum Walk	■		
Sanderson [90]	Grover's Algorithm [120]		■	
ShorVis [59]	Shor's Algorithm [1]		■	

B. Algorithm Development

Unlike the quantum computer implementation that focuses on qubits, gates, and circuits, algorithm development emphasizes the conceptual logic and mathematical structure of quantum algorithms. At this level, we include studies that develop visualizations for specific quantum algorithms, as well as those that create graphical languages for algorithm analysis.

Quantum Algorithms: Most existing visualizations at this level address textbook quantum algorithms, such as Grover's Algorithm [120], Shor's Algorithm [1], and Quantum Fourier Transform. The primary data are intermediate values produced during algorithm execution, including probabilities, state vectors, and amplitudes, which are typically presented in the sequential order of algorithmic steps. A notable example is the video by Sanderson [90] on Grover's Algorithm [120]: it operates entirely at the level of state vectors and their geometric relationships, without descending to gates or circuits. These studies are summarized in Table VI.1. Beyond textbook algorithms, some studies develop interactive systems that address more complex algorithmic subjects, such as quantum backtracking [109], quantum machine learning [76], [82], and quantum cryptography [58].

Graphical Languages: Graphical languages provide alternative representations of quantum computation processes. They are not designed for one specific quantum algorithm. Instead, they support reasoning about how quantum states, gates, and circuits compose and transform. These representations can expose algebraic and compositional structures that are difficult to see in standard circuit diagrams. The ZX-calculus [51], [121]–[123] is a rigorous graphical language that represents quantum processes as tensor networks, generalizing Z and X operations to reveal such structures. Decision diagrams [49], [112], [124] offer compact graph-based representations of states and gates that capture their algebraic structure.

VII. VISUALIZATIONS FOR QC

We detail the third dimension of our taxonomy: *how* QC subjects are visualized. Whereas Sections V and VI define the problem space (*why* and *what*), this dimension surveys the solution space. Throughout the section, we discuss what the data types are, what the challenges in visualizing the data are, and how different visualizations address the challenges.

We group existing visualization techniques for QC according to whether the data to be visualized is intrinsic to quantum computing or not. Specifically, the QC data that has been visualized in existing VIS4QC research can be categorized into two types: **computational data** that arises from or is intrinsic to the quantum computation process, and **operational data** that describes the characteristics and performance of the quantum hardware and software environment.

We also cross-reference the abstractions from Section VI to subdivide data at each level into the two categories, as shown in Fig. 9. At the Qubits and States level, **quantum states** are intrinsic to the computation, carrying complex-valued amplitudes that encode superposition and entanglement. Qubit statistics such as error rates and coherence times (T_1, T_2), by contrast, characterize the physical medium independent of what it computes. At the Gates and Circuits level, **circuit structures** define the program's logic and how gates compose, and **state transitions** specify how quantum states transform, whereas data such as gate fidelities and circuit depths quantify execution quality and complexity. The Machines and Systems level concerns the physical substrate that hosts computation rather than performs it, so the data here, such as qubit connectivity and calibration parameters, is exclusively operational. At the Algorithm Development level, operational data such as accuracy and fidelities measure the quality of algorithmic outcomes, while the computational aspect reduces to sequences of transformations on the data from lower levels and thus is not discussed separately.

We detail the visualization techniques for the computational data across the abstraction levels in Section VII-A and summarize the techniques for operational data in Section VII-B. Fig. 8 overviews the visualization techniques for the computational and operational data.

A. Visualizations for Computational Data

This subsection discusses visualization techniques for the computational data, organized by the three subjects identified above: quantum states, state transitions, and circuit structures. Within each subject, we examine how different visualization techniques address the unique challenges and requirements of that data type.

1) Quantum States: For quantum states, a single-qubit state requires two complex numbers, while an N-qubit state requires 2^N complex numbers. This exponential growth poses the central scalability challenge for visualization design, which also distinguishes visualizations for multi-qubit states from those for single-qubit states. Therefore, we first introduce single-qubit visualizations (emphasizing visual encoding choices) and then discuss visualizations for multi-qubit states (emphasizing visualization techniques for scaling to more qubits).

Single-Qubit States: Visualization techniques for single-qubit states primarily differ in geometric mapping and visual encoding.

Spheres: These visualizations originate from the Bloch Sphere [10]. Bloch reparameterized a single-qubit state as

$$|\psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle \tag{4}$$

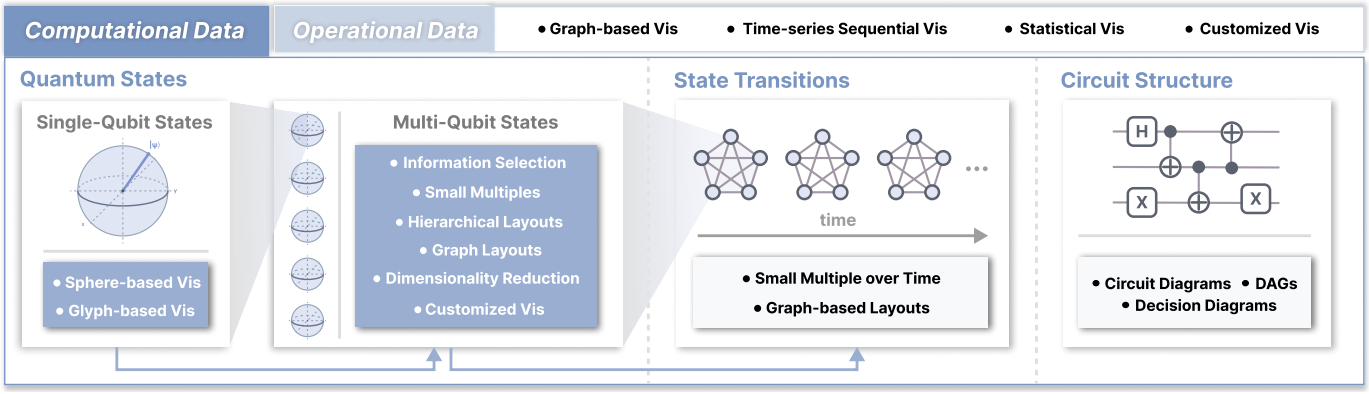


Fig. 8. Overview of visualization techniques for QC data. Computational data requires techniques tailored to quantum-specific structures: quantum states are visualized through single-qubit encodings and multi-qubit scaling techniques, state transitions use small multiples and graph-based layouts to show how states change during circuit execution, and circuit structures use circuit diagrams, DAGs, and decision diagrams. Operational data, such as hardware statistics and circuit metrics, is commonly visualized with general-purpose graph-based, time-series, statistical, and customized views.

	Computational Data	Operational Data
Qubits & States	Quantum States	Qubit Statistics (e.g., error rates, T1 and T2 times, etc.)
Gates & Circuits	State Transitions Circuit Structures	Statistics of Gates and Circuits (e.g., gate counts, circuit depth, etc.)
Machines & Systems	—	Hardware Properties and Statistics (e.g., connectivity, calibration data, etc.)
Algorithm Development	Sequence of Transformations on Comp. Data	Metrics of Algorithms (e.g., accuracy, fidelities, etc.)

Fig. 9. Data by abstraction levels and data types. We discuss visualizations for quantum states, state transitions, and circuit structures in detail in Section VII-A and summarize those for operational data in Section VII-B.

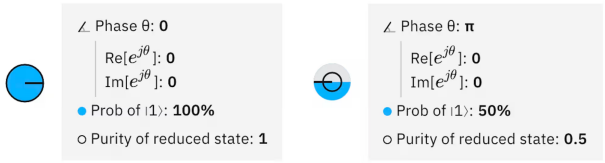


Fig. 10. Phase-disk representations from IBM Quantum Composer [118]. A phase disk uses a compact 2D glyph to encode a single-qubit state, with filled area representing the probability of measuring $|1\rangle$, the radial arm representing phase, and the ring radius representing purity.

where $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. This parameterization naturally maps the two complex amplitudes of a single-qubit pure state to a point on a unit sphere with two angles (θ, ϕ), as shown in Fig. 6. Mixed states appear as points inside the sphere, with radial distance encoding purity. The Bloch Sphere can also visualize the effects of quantum errors such as bit flips, phase flips, and decoherence [125], where different noise channels correspond to distinct geometric transformations of the state vector. *BEADS* [83] extends this technique using colored spherical glyphs, where surface color encodes expectation values of Pauli operators, and brightness encodes the entanglement level of that qubit within a larger system.

2D Glyphs: Despite the natural correspondence between single-qubit states and spheres, sphere-based representations have drawbacks common in 3D visualizations [126], [127], as

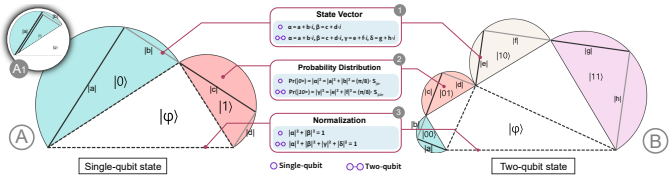


Fig. 11. Single-qubit and two-qubit state visualizations from *VENUS* [70, p. 5]. Coordinated right triangles encode complex amplitudes through line segments and measurement probabilities through semicircle areas, extending the same visual grammar from one qubit to two qubits. Courtesy of the authors.

well as difficulties in visualizing probabilities and amplitudes. 2D glyphs address these drawbacks by projecting single-qubit information (e.g., amplitudes, probabilities, and phases) onto 2D representations. For example, the *phase disk* [118] (Figure 10) combines multiple encodings: filled area for probability, a radial arm for phase, and ring radius for purity. This compact design enables the display of multiple qubit states simultaneously alongside circuit diagrams. The vector field visualization by Suau et al. [65] maps the Bloch Sphere onto a 2D plane using Robinson projection, overlaying heatmaps (for purity or fidelity) with vector arrows (showing deviations from ideal states) to reveal state-dependent errors.

Multi-Qubit States: No single approach can fully represent large multi-qubit states. Therefore, researchers have developed different techniques that view the data of multi-qubit states from different perspectives, which entail distinct trade-offs between completeness, scalability, and interpretability. The reasons why individual approaches fail are as follows. While single-qubit visualizations provide building blocks, multi-qubit states exhibit properties that no single-qubit view can capture. For example, entanglement exists only in the joint state and cannot be observed from individual qubits alone. Probability and phase are distributed across an exponentially large basis-state space rather than factoring neatly into per-qubit components. Patterns, symmetries, and anomalies often emerge only when viewing the full joint state or when comparing states across different conditions, such as ideal versus noisy or before versus after a gate operation. The exponential growth of the

state space (2^N for N qubits) makes visualizing these properties significantly more challenging than visualizing single-qubit states. Below, we detail these techniques and discuss how they address the scaling challenge. Some techniques process selected or projected state values with statistical charts, some stack single-qubit views, and others leverage visual structure to reveal patterns.

Small Multiples: Instead of inventing new visualizations, these approaches circumvent the challenge of exponential growth by stacking single-qubit visualizations side by side. This technique scales linearly with N but sacrifices the ability to show inter-qubit relationships, such as entanglement. Phase disks or Bloch spheres shown alongside circuit diagrams in IBM Quantum Composer [118] exemplify such techniques. Multiple Q-Beads in *BEADS* [83] can be overlaid on a circuit diagram to show the single-qubit states of multiple qubits.

Graphs and Node-Link Diagrams: These approaches circumvent the challenge of exponential growth by formulating multi-qubit states into graph data, emphasizing relationships between nodes that represent basis states or qubits. Edges then encode a chosen relation between those nodes, such as one-bit adjacency between basis states, membership in a qubit-correlation subset, or a graph-state entangling relation. In *Dimensional Circle Notation* by Bley et al. [71], the graph is the N-dimensional hypercube of the computational basis. Each node represents one basis state $|i_N \dots i_1\rangle$ and carries a circle glyph for its complex amplitude α_i. The filled inner-circle radius of the circle glyph encodes the amplitude magnitude. The area encodes measurement probability, and the radial line encodes phase. Each edge connects two basis states that differ by exactly one bit, so an edge represents changing one qubit value while holding the other qubit values fixed. This construction extends systematically with the number of qubits. Adding a qubit adds one hypercube dimension, doubles the number of nodes from 2^N to 2^{N+1}, and introduces a new family of edges parallel to the new qubit axis. For four- and five-qubit examples, Bley et al. use projected or modular hypercube layouts, assigning selected qubits or groups of qubits to axes so that the graph can emphasize particular entanglement relations or unitary operations. *BEADS* [83] uses a different graph abstraction. Its nodes are qubits, drawn as Q-Beads that encode each qubit's reduced state. Links represent correlation subsets among those qubits. For a pairwise relation $\{k,l\}$, a correlation bead is placed on the line between Q_k and Q_l, indicating that the encoded correlation belongs to that two-qubit subsystem. The bead type distinguishes connected entanglement-based correlations (E-Beads), compound correlations (C-Beads), and total correlations (T-Beads). For higher-order relations, the graph becomes a hypergraph: a bead labeled by a subset such as $\{k,l,m\}$ represents a three-qubit correlation function, and auxiliary lines indicate which qubits participate in that relation. *GraphStateVis* [62] visualizes graph states, a special family of multi-qubit stabilizer states used in QEC, as undirected qubit graphs. Each node represents a qubit, and each edge represents the controlled-Z relation used to prepare the graph state. Changing the graph topology changes the represented stabilizer state. Together, these examples show that graph visualizations for multi-qubit states can define nodes over basis states or physical qubits. The corresponding links then represent different state relations, including one-bit basis-state adjacency, qubit-correlation subsets, and graph-state entangling relations.

Hierarchical Data and Visualizations: These approaches view multi-qubit states as hierarchical data, building on the hierarchical and recursive nature of the states and rooted in the binary decomposition of basis states. They manage the exponential growth through visual patterns with recursive or fractal structures, imposing an order on qubits and using recursive decomposition to represent the state. *Rainbow boxes* [61] visualizes a multi-qubit state with a treemap. Qubits are columns, and basis terms of separable factors are shown as stacked boxes spanning the qubits involved in each factor. *Fractal representations* [54], [55] build a self-similar hierarchy from a single-qubit bar. Each added qubit recursively copies a scaled bar underneath the 0/1 segments of the previous level, so the layout follows the binary decomposition of basis states. Segment width encodes probability, bar color encodes bit value, and horizontal markers encode phase. *VENUS* [70], shown in Figure 11, uses a shallow geometric hierarchy rather than a full fractal layout. Its one-qubit visual unit combines right triangles for the real and imaginary parts of each amplitude with semicircles whose areas encode the corresponding probabilities; the two-qubit view repeats this unit for the four basis states and uses auxiliary triangles to link the units through the shared normalization constraint. The backtracking-tree view in Qrisp [81] visualizes a search-specific hierarchy over encoded basis states. The root is the starting search state, moving down the tree adds one branch choice at a time, and each node shows the resulting intermediate candidate. Branches that fail the rejection test are cut off, while node color indicates the sign of the nonzero amplitude as the quantum walk spreads through the tree.

Spheres: These approaches extend the idea of the Bloch Sphere [10] with clever mathematical formulations, but they do not address the exponential growth challenge. *Q-sphere* [7] arranges basis states on a sphere by Hamming weight, encoding probability as point size and phase as color. The *Majorana representation* [53] maps an N-qubit state to $2^N - 1$ points on the Bloch sphere. While the number of points is still exponential, this re-parameterization translates algebraic properties into geometric ones: the spatial configuration of points reflects entanglement structure and separability.

Basic Charts: These approaches share a common pipeline: they first select *partial* information from a multi-qubit state or project high-dimensional state data into a lower-dimensional space, and then visualize the processed values with basic statistical charts. Therefore, they trade completeness for compact summaries, distributions, or projections rather than directly representing the full quantum state. When the selected quantities are probabilities, amplitudes, or density-matrix entries, simple bar charts can directly display values for all 2^N basis states, which is practical only for small N, typically no more than eight qubits. The cityscape plot [7] is a 3D bar chart that visualizes matrix elements in the density matrix of a quantum

state. The *state-o-gram* [89] is a specialized scatterplot that maps phase angle to horizontal position and probability to vertical extent, unifying amplitude and phase information in a single chart. For larger state spaces or distribution-level analysis, dimensionality reduction methods further transform the data before visualization. *XQAI-Eyes* [93] applies Principal Component Analysis (PCA) to density matrices and visualizes the projected quantum state distributions in QML applications with a 2D scatter plot.

Customized Glyphs: To address challenges in specific scenarios, these approaches design glyphs tailored to specific analytical needs, with scalability depending on the design. The satellite chart in *VIOLET* [82] positions "satellite" glyphs at hypercube vertices representing basis states, with line segments connecting to qubit axes; line color encodes probability, and stacked bars at corners show single-qubit marginals. The dandelion chart in *QuantumEyes* [78] uses a scatter plot where position encodes complex amplitude (real vs. imaginary) and circle area encodes probability.

Despite these advances, no existing visualization fully represents an N-qubit state for large N without information loss. All approaches necessarily involve selective display (showing only significant terms), aggregation (combining related terms), or dimensionality reduction.

2) Quantum State Transitions: Beyond static quantum states, state transitions (Section VI-A2) represent a higher level of abstraction. Therefore, they require visualizations that combine multiple state representations and illustrate their relationships over time. This adds a temporal scaling challenge to the exponential state-space scaling discussed above, requiring additional considerations.

Small Multiples: These approaches view state transitions as a sequence of quantum states and visualize them as a series of state visualizations. *Rainbow boxes* [61] arrange state snapshots sequentially, making it possible to observe how gates transform probability, phase, and entanglement; for instance, seeing a CNOT gate convert single-column (separable) rectangles into multi-column (entangled) ones. *BEADS* [83] overlays multiple BEADS representations on circuit diagrams, showing how BEADS evolve over time.

Graph Visualizations: Beyond simple sequences, these approaches regard state transitions as links between nodes in a graph, revealing deeper transition structures, such as interference and noise propagation. *QuFlow* [60] uses a regular graph layout where nodes are arranged in rows (basis states) and columns (gates); cell color encodes probability, and connecting lines show how amplitudes flow between states, with different line patterns for different gate types. However, since each basis state occupies a separate row, the layout scales as 2^N and is limited to small qubit counts. *QuantumEyes* [78] coordinates multiple views: (1) a Sankey-like diagram where flowing bands connect basis states across gate operations, with band width encoding probability transitions; (2) a node-link graph showing state transformations, including splitting and merging from interference; and (3) a table explaining individual gate operations. *HammingVis* [40] visualizes erroneous measurement outcomes in Hamming space using a force-directed layout, where nodes are output bitstrings, directed edges reveal likely error propagation among neighboring outcomes, and node proximity reflects Hamming distance.

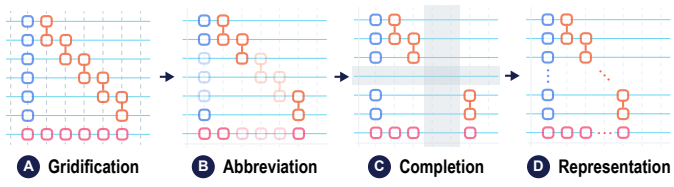


Fig. 12. Visual abstraction process in *Quantivine* [68, p. 6]. The method converts a large quantum circuit into a compact representation by identifying functional modules, grouping repeated gate patterns, and preserving contextual information such as qubit provenance and gate placement. Courtesy of the authors.

While existing approaches show state snapshots or transitions, none of them explicitly visualize the *difference* between two quantum states. Such a visualization of quantum state differences would highlight what has changed across a gate operation or between ideal and noisy conditions.

3) Circuit Structure: Node-link diagrams are the dominant visual representation for quantum circuit structures because the data is inherently graph-like. However, they have two different formats, quantum circuit diagrams and general graphs, for different needs.

Quantum Circuit Diagrams: Standard circuit diagrams represent gates as boxes or symbols on horizontal qubit lines. Unlike classical circuits, quantum circuits have no fan-in or fan-out: qubit lines run strictly in parallel without merging, which constrains layouts to a grid of parallel wires with gate symbols placed on them. While familiar to practitioners and students, this notation becomes impractical to scale for large circuits with hundreds of qubits and thousands of gates. Therefore, a few approaches address this by abstracting repetitive patterns. *Quantivine* [68] (Figure 12) groups gates into semantically meaningful modules, using program analysis to identify module boundaries and abstracts repetitive patterns (vertical, horizontal, diagonal) into compact representations. Its context views show qubit provenance, parallelism, and a connectivity matrix for entanglement relationships. *CircInspect* [73] abstracts gates into collapsible blocks, supporting drill-down from high-level overviews into specific components.

General Graphs: These approaches represent general node-link diagrams. DAGs offer an alternative representation that makes computational dependencies explicit, which are convenient for circuit analysis and optimization. They represent gates as nodes and qubit dependencies as directed edges, removing the fixed qubit-line layout of circuit diagrams to expose the partial ordering among operations. Qiskit [7] and Qualtran [110] provide DAG visualizations that display parallelism through graph width and highlight critical paths. Decision diagrams [112] offer another visual interpretation of quantum computations. They represent quantum states and operations as node-link trees, where nodes represent recursive decompositions, and edges encode amplitude contributions using color for phase and thickness for magnitude.

B. Visualizations for Operational Data

Operational data in QC, including hardware properties, calibration parameters, error rates, and circuit metrics, can largely be visualized using general-purpose techniques. We categorize these visualizations into graph-based, time-series, statistical, and customized approaches.

1) Node-Link Diagrams: For operational data, node-link diagrams primarily represent qubit topology and connectivity. Quantum processors have fixed coupling maps that determine which qubits can directly interact, and this graph structure maps directly to node-link diagrams.

Both Qiskit [7] and Cirq [8] provide functions to visualize device topology as node-link diagrams, where nodes represent qubits and edges represent allowed two-qubit gate connections. Qiskit's `plot_gate_map()` renders the qubit layout with customizable node colors, sizes, and labels, while Cirq's `draw_gridlike()` displays Google's grid-based qubit architectures. These topology views can be augmented with additional data: Qiskit's `plot_error_map()` overlays gate error rates using color encoding, combining topological structure with statistical information. *QVis* [39], [80] extends topology visualization with interactive capabilities: qubit selection, cross-view filtering, and color-coding by cluster membership based on temporal performance patterns.

In contrast to the node-link diagrams for computational data (Section VII-A), which represent exponential state spaces and entanglement relationships, the node-link diagrams for operational data represent fixed physical structures with a small number of nodes (i.e., qubits) without the challenge of exponential growth.

2) Basic Charts: Standard basic charts (e.g., bar charts, histograms, line charts, and heatmaps) are widely used for displaying distributions and comparisons of QC metrics.

Bar charts and histograms appear frequently. *QVis* [39], [80] combines several basic charts for operational metrics. It uses binned heatmaps to aggregate dense time-value observations and line charts to inspect selected qubits over time. It also uses histograms for metric distributions and distance-matrix heatmaps for qubit similarity. Its clustering view shows groups of qubits as small multiples of time-series traces, with a barycenter line summarizing each cluster. For circuit optimization, QVis uses bar charts and stacked bars to compare circuit depth and single-qubit versus multi-qubit gate counts across optimization levels. Other systems use basic charts as targeted summaries rather than as the primary interface structure. Kim et al. [13] use basic charts and tables after translating raw bitstring counts into problem-specific variables, such as natural numbers or Boolean assignments. They also visualize layer-wise and cumulative estimated success probability for circuit fidelity. *QubiCSV* [79] plots calibration properties within and across commits.

Basic charts for computational data must additionally consider complex amplitudes and phases. In contrast, basic charts for operational data deal with real-valued metrics and can directly apply standard chart types.

3) Customized Glyphs: Some visualizations introduce novel glyph designs for QC-specific data and tasks that standard categories do not cover. For example, *VACSEN* [38] introduces several domain-specific designs: a Computer Evolution View that adopts a circuit-like visual metaphor to display the temporal evolution of qubit and gate properties, with custom glyph encodings where color saturation represents T_1/T_2 values and bar length represents gate usage counts. These customized designs demonstrate that while general-purpose techniques cover many needs, certain QC workflows benefit from specialized visual encodings.

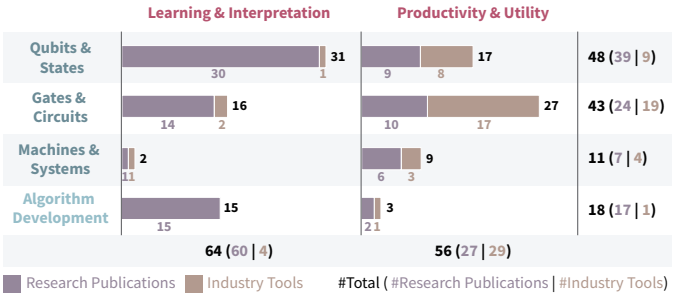


Fig. 13. Coverage of surveyed PTs across the problem space. Rows show QC abstraction levels, columns show usage and purpose categories, and each stacked bar separates research publications from industry tools. The distribution highlights concentrations in Qubits & States for Learning & Interpretation and in Gates & Circuits for Productivity & Utility, with thinner coverage at the Machines & Systems and Algorithm Development levels.

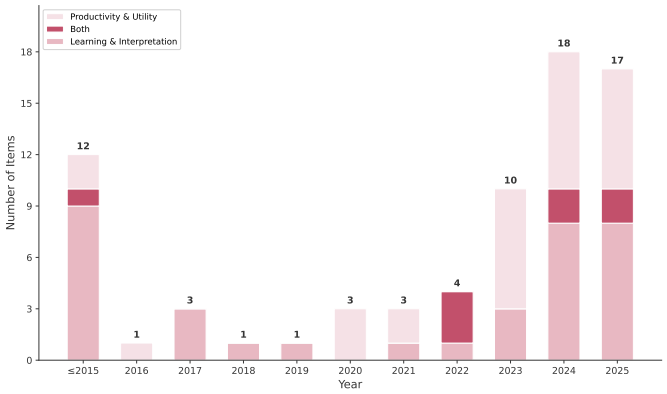


Fig. 14. Temporal distribution of surveyed PTs by usage and purpose. The first bin aggregates PTs published before and including 2015; later bins show yearly counts. Colors distinguish Learning & Interpretation, Productivity & Utility, and PTs coded with both purposes, showing a sharp increase in VIS4QC research from 2023 to 2025.

Noteworthy is that these categories are not mutually exclusive. For example, the function `plot_error_map()` in Qiskit [7] combines graph-based topology with statistical color encoding. Temporal heatmaps blend time-series structure with statistical aggregation.

VIII. Current State of VIS4QC

The current VIS4QC landscape is still small but growing quickly. Our survey identifies 69 core PTs, and Fig. 14 shows that much of the recent growth occurs after 2023. Earlier work mainly supports Learning & Interpretation, often by explaining quantum states, gates, and textbook algorithms. Since 2023, more Productivity & Utility work has appeared, reflecting the maturation of quantum hardware, SDKs, and development workflows. Together, these trends suggest a field

that is expanding beyond explanation-oriented visualizations, but has not yet developed evenly across application needs.

A. Coverage Patterns in the Problem Space

Figure 13 shows the distribution of surveyed PTs across our problem-space grid and separates them by **source** (research publications vs tools). This reveals structural patterns that aggregated counts obscure. We read the grid at three scales: by usage category, by abstraction level, and by individual cells.

Academia–Industry Divide: Comparing the two usage categories, Learning & Interpretation (64 PTs) and Productivity & Utility (56 PTs) appear roughly balanced. However, overlaying the source distinction reveals that these two categories are populated by largely separate communities (academia vs industry). Learning & Interpretation is overwhelmingly driven by research publications (60 out of 64), with minimal industry presence. In contrast, tools constitute a much larger share of Productivity & Utility (29 out of 56), comparable to research publications (27 out of 56). In other words, academic researchers predominantly develop visualizations for learning and interpretation, while industry practitioners ship tools for productivity and utility. Neither community adequately serves the other's needs. This structural mismatch has concrete consequences: as discussed in Section IX, academia has not built the productivity tools that practitioners need, while industry tools have not adopted the richer visualizations that academia has developed.

Thin Coverage at the Machine and Algorithm Levels: Moving from usage categories to abstraction levels, the coverage is heavily uneven. Machines & Systems (11 total) and Algorithm Development (18 total) are substantially underrepresented compared to Qubits & States (48) and Gates & Circuits (43). The field's coverage drops sharply as one moves toward higher abstraction levels, leaving the levels closest to practical applications with the least visualization support.

Concentration in States × Learning and Circuits × Productivity: Zooming in on individual cells reveals where the coverage concentrates. Learning × Qubits & States (31), i.e., PTs that visualize qubits and states for learning, and Productivity × Gates & Circuits (27) together account for roughly half of the field. This concentration reflects both historical momentum and present-day incentives. Quantum state visualization has roots in physics well before quantum computing became a field (the Bloch sphere dates to 1946 [10]), giving the Learning × Qubits & States cell a decades-long head start. Productivity × Gates & Circuits, by contrast, emerged alongside modern SDKs and has grown rapidly since 2023 (Figure 14). The current problem space is thus shaped by where the field has been, not only by where it needs to go.

Algorithms Explained but Not Tooled: The imbalance is especially stark within Algorithm Development. Productivity × Algorithm Development contains only 3 PTs, making it the least populated cell in the grid. Almost no one is building productivity tools that help experienced users develop, debug, or optimize quantum algorithms. Meanwhile, Learning × Algorithm Development (15) consists entirely of research publications that explain algorithms visually but do not provide tools to *work with* them.

Productivity without Utility: A subtler gap runs through the Productivity & Utility category: all existing tools target *productivity* (faster workflows), but none targets *utility* (better task outcomes) as a primary goal. Utility gains, when they occur, arise as a byproduct of improved workflows rather than as an independent design objective (Section V).

B. Application Contexts in the Current Landscape

The coverage patterns above describe the field in aggregate. Application contexts show how these patterns surface in concrete settings, where visualization needs are shaped by users, tasks, and domain constraints. We treat these contexts as cross-cutting views because one application context may span multiple abstraction levels and usage purposes.

1) Quantum Education: Quantum education is the clearest application context for the Learning & Interpretation side of the problem space. It spans multiple abstraction levels because students must move from basic quantum mechanics (QM) concepts to qubits, gates, circuits, algorithms, and hardware. This is a demanding setting because students must coordinate multiple representations [128], [129], often without direct laboratory experience [130]. Although many quantum education studies are not visualization papers, several use visual representations to teach quantum concepts [56], [58], [113]–[116]. This context makes the coverage patterns above concrete: learners need connected visual support across topics, but existing VIS4QC work is uneven, fragmented, and rarely packaged for classroom use.

Uneven Coverage & Problem Space Gaps: The uneven coverage observed in Figure 13 directly impacts education. Learning & Interpretation × Qubits & States is well-populated (31 PTs), providing educators with many options for teaching quantum states. However, Learning & Interpretation × Machines & Systems contains only 2 PTs, meaning students receive almost no visual support for understanding quantum hardware. Current VIS4QC research has developed opportunistically rather than systematically, resulting in a lack of continuity between existing visualizations. Some topics are well-explored, such as single-qubit state visualization, while others remain neglected, such as the transition from noiseless to noisy qubits and the transition from simple algorithms to advanced ones. Even where visualizations do exist for individual topics, they rarely connect to one another, leaving gaps in the conceptual progression that learners must navigate.

Lack of Visual Coherence: There is no coherent visual progression that guides learners from foundational concepts to advanced topics. Visualizations do not naturally evolve as students move from single-qubit to multi-qubit systems, from individual gates to complete algorithms, or from conceptual models to real quantum systems. These pedagogical transitions lack established visual pathways, forcing educators to improvise or abandon visualization altogether at key conceptual junctures. This problem is compounded by the diversity of visualization research, where novelty is a primary contribution.

Different research groups develop incompatible representations for similar quantum concepts, making it difficult for educators to assemble a coherent visual curriculum even when individual components exist. Such diversity advances the research frontier but creates a barrier in education, where learners need visualizations that connect across topics.

Lack of Integration: Existing visualizations are predominantly standalone applications, each developed for a specific purpose. No comprehensive educational toolkit integrates them. This fragmentation, rooted in the research–industry divide (Section VIII), forces teachers to navigate multiple platforms with different interaction paradigms and visual styles. As a result, many quantum computing courses rely on mathematical formalism alone, despite evidence that visualization improves conceptual understanding [128]. For example, researchers have proposed rich representations for multi-qubit states (e.g., [53], [55], [70]), yet instructors often fall back to abstract vectors and matrices because these research prototypes are not packaged for classroom use.

Lack of Engagement: The abstract and counterintuitive nature of QM makes engaging learning experiences especially important for sustaining student motivation. Quantum games [18]–[20], [56], [58], [113]–[116] show promise by offering immediate feedback and intrinsic motivation that traditional lectures cannot provide. However, few existing games use the visualization techniques developed in the research community, instead relying on basic representations. The scarcity of interdisciplinary talent bridging game design, visualization, and quantum physics further constrains progress.

Taken together, quantum education shows how aggregate coverage gaps become practical barriers in an application context. The field contains many useful visualizations for individual concepts, but they remain difficult to assemble into a coherent learning pathway from single-qubit states through circuits, algorithms, and real quantum hardware.

2) Quantum Software Engineering: For the Productivity & Utility side of the problem space, quantum software engineering is the clearest application context. It spans the development workflow from circuit authoring and simulation to transpilation, backend selection, execution, debugging, optimization, and result analysis. Unlike quantum education, this context assumes experienced users who are trying to make quantum programs work on noisy hardware and in hybrid quantum-classical pipelines.

Fragmented Workflow Support: Existing VIS4QC support for quantum software engineering is fragmented across workflow stages. *Quantivine* [68] and *CircInspect* [73] support circuit inspection and abstraction; *qprof* [66] supports profiling; *VACSEN* [38] and *QVis* [39], [80] support noise-aware machine and circuit analysis; *HammingVis* [40] supports analysis of erroneous measurement outcomes; and *QubiCSV* [79] supports calibration and control data analysis.

Weak Hardware-Algorithm Connection: Quantum software engineering depends on connecting high-level algorithm intent to low-level hardware constraints. However, existing visualizations usually expose only one part of this chain at a time. Developers can inspect circuits, compiled implementations, backend metrics, or measurement outcomes, but receive limited support for reasoning about how these artifacts jointly shape program behavior and execution quality. This gap is also reflected in workflow-level analyses of existing QC tools [13].

Algorithms Explained but Not Operationalized: Existing algorithm-level visualizations [50], [59], [67], [72] are mainly designed to explain algorithms rather than support developers who need to implement, debug, or optimize them. As a result, algorithm understanding remains separated from day-to-day software engineering workflows in SDKs.

Productivity without Outcome Guidance: Current visualizations for quantum software engineering mainly improve productivity by making workflow artifacts easier to inspect, as seen in work on circuit abstraction, profiling, and noise-aware analysis [38], [66], [68], [73]. They rarely make utility an explicit target, such as guiding developers toward higher-fidelity circuits, better backend choices, lower-depth implementations, or more reliable task outcomes.

IX. Challenges and Opportunities

Building on the coverage patterns and application-context analysis in Section VIII, we discuss challenges and opportunities at each abstraction level, drawing on the three dimensions *Usage & Purpose*, *Abstraction Levels* and *VIS4QC*.

A. Qubits and States

Qubits & States is the most populated abstraction level (48 PTs), with a strong concentration in the Learning & Interpretation column (31 PTs). Despite this relative maturity, two issues stand out: the scalability of state visualization and the secondary treatment of phase.

Scalability of Quantum State Visualization: Visualizing the 2^N-dimensional complex state space of an N-qubit system quickly becomes intractable even for small N (e.g., $N = 5$). As discussed in Section VII-A1, each visualization technique for multi-qubit state visualization entails distinct limitations. Basic statistical charts quickly become unusable as the number of basis states grows exponentially with N. Small multiples scale linearly but cannot capture inter-qubit relationships such as entanglement. Hierarchical layouts can theoretically extend to arbitrary N but are limited by physical constraints (e.g., screen size) in practice. Graph layouts capture inter-qubit relationships, but the number of edges can grow quadratically with N. Dimensionality reduction enables distribution-level analysis for large N, but it can obscure the underlying quantum structure due to lossy information compression. This fundamental tension between completeness, scalability, and interpretability remains unresolved.

Phase Visualization: Many designs treat phase as secondary: while probability is readily conveyed through area or position, the complex phase of quantum amplitudes is often relegated to color mapping. This issue can obscure subtle phase relationships that are critical to quantum interference. Since phase is central to essentially all quantum phenomena (e.g., interference and entanglement), improving visualizations for phases at the state level would directly benefit the understanding of gates, circuits, and algorithms that depend on it.

B. Gates and Circuits

Gates & Circuits is well-covered overall (43 PTs), but the two usage categories differ sharply in depth. The Productivity column is dominated by tools that provide basic circuit diagrams (17 out of 27), while the Learning & Interpretation column consists almost entirely of research publications with richer visualizations. Neither side adopts techniques from the other. Beyond this source mismatch, several visualization challenges and opportunities emerge at this level.

Dynamic Quantum Circuits: Circuits that feature mid-circuit measurements, classical control flow, and conditional operations pose new challenges that existing circuit diagram conventions do not address, as the circuit structure itself becomes dependent on runtime measurement outcomes. As quantum hardware increasingly supports these features, visualization techniques must evolve accordingly.

Visualizations of State Differences: As mentioned in Section VII-A2, the explicit visualization of state differences remains unexplored. State differences embed more mathematical structures than step-by-step state snapshots. For example, when a Hadamard gate is applied to qubit k of an N-qubit system, all 2^N amplitudes may change. Yet the transformation is local: the gate acts as $I^{\otimes(k-1)} \otimes H \otimes I^{\otimes(N-k)}$, affecting only a 2×2 subspace. Visualizing the structure within state differences, such as identifying the local operator responsible for the change, would expose this underlying simplicity, turning an apparently global change into an interpretable local operation. Techniques that reveal such structures in state differences could make quantum evolution understandable at scale.

Visualizations for Transpilation Understanding: As circuit models are transpiled to target-specific gate sets and qubit topologies with various optimizations, developers often struggle to understand how their high-level algorithms and circuit models map to physical implementations. This is one concrete form of the weak hardware-algorithm connection observed in quantum software engineering (Section VIII-B2). Such a fine-grained understanding is crucial for optimizing real-world applications, especially when current quantum hardware is far from noise-tolerant. However, transpilation is particularly challenging because it presents an optimization problem under hardware constraints with various trade-offs, heuristics, and stochasticity. For example, optimizing for the minimum count of physical gates while considering available native gates and qubit connectivity is non-trivial and often requires heuristic and stochastic search algorithms. This means a circuit model can be translated into different physical circuit implementations with varying fidelities and depths across runs of the same transpiler. Conversations with our QC collaborators reveal a strong interest in visual tools that explain transpilation decisions, highlight optimization opportunities, and help diagnose performance bottlenecks introduced during transpilation. Such tools would directly address a practical pain point in quantum software development.

Visual Debugging Tools: Quantum debugging differs fundamentally from classical debugging in several ways. First, the no-cloning theorem prevents inspection without disturbance: unlike classical variables that can be freely examined, measuring a quantum state collapses it, making traditional "print statement" debugging impossible. Second, quantum errors can be subtle and non-local: a circuit may produce correct probability distributions yet harbor phase errors that only manifest through interference in subsequent operations. Third, entanglement creates correlations that have no classical analog, meaning bugs can propagate across qubits in ways that are difficult to trace. Fourth, probabilistic outcomes require statistical aggregation over many runs to distinguish true errors from expected quantum randomness. These distinctions suggest opportunities for visualizations that help developers identify where quantum states deviate from expectations and track the propagation of errors through entangled subsystems. Such visualizations need to work within the constraints of quantum measurement. For example, to obtain accurate information about quantum states, small-scale validation of quantum circuits can be done through classical simulation, requiring integration with classical simulators. However, debuggers may need to incorporate advanced techniques such as quantum state tomography to analyze quantum states on quantum computers. These approaches can provide simulated or approximate state information without fully collapsing the quantum states during computation.

C. Machines and Systems

As the most underrepresented abstraction level (11 PTs), Machines & Systems has many open problems, particularly in the Learning & Interpretation column where almost no existing work teaches how quantum machines work visually.

Visualizations for Quantum Hardware Statistics: Quantum hardware generates rich operational data, including calibration parameters, noise profiles, and cross-talk patterns. Visualizing these could help hardware developers and algorithm designers reduce the impact of noise on their computations. Currently, only 9 productivity-oriented PTs address this level, most providing basic representations.

Visualizations for Different Hardware Paradigms: The emergence of new quantum computing hardware paradigms, including photonic, neutral atom, and topological approaches, will require visualization techniques adapted to their unique characteristics. Each modality brings different native gate sets, connectivity constraints, and error models, suggesting opportunities for modality-specific visual representations. For example, quantum computers based on superconducting qubits pose qubit connectivity constraints, while those based on neutral atoms are more flexible in connectivity, allowing for even all-to-all connectivity. These characteristics, rooted in hardware design, pose challenges for visualization techniques and require tailored approaches.

D. Algorithm Development

Algorithm Development is the most underexplored level in terms of productivity support: Productivity × Algorithm Development contains only 3 PTs, making it the sparsest cell in the grid. Current VIS4QC research predominantly focuses on algorithm primitives (e.g., individual quantum gates, basic

circuits, and simple state transformations) rather than on complete quantum algorithms. The visualized algorithms are typically textbook examples, as shown in Table VI.1. Complex quantum algorithms that combine multiple primitives, involve classical-quantum hybrid workflows, or require sophisticated error mitigation strategies remain largely unexplored from the perspective of visualization research. This limitation constrains the use of visualizations in real-world quantum algorithm development and debugging. Stronger visualization support for Algorithm Development would benefit both major application contexts discussed in Section VIII: it would help learners connect circuit primitives to complete algorithms in quantum education (Section VIII-B1), and help developers operationalize algorithm understanding in quantum software engineering (Section VIII-B2).

Mechanism-Based Algorithm Visualization: Current approaches predominantly visualize quantum *states*; that is, the computational data at various steps during algorithm execution. An alternative approach focuses on visualizing quantum *mechanisms*, i.e., the underlying mathematical structures that explain *why* algorithms work, rather than *what* states they produce. This enables fundamentally more scalable visualizations. Consider Grover's Algorithm [120] as a canonical example. Conventional approaches visualize amplitude distributions across 2^N basis states at each iteration, requiring exponential visual complexity. In contrast, the mechanism-based approach [90] recognizes that Grover's Algorithm operates through rotations in a two-dimensional subspace spanned by the target state and the uniform superposition. Visualizing this 2D rotation by showing how successive Grover iterations rotate the state vector toward the target achieves $O(1)$ visual complexity, regardless of the number of qubits. This geometric interpretation not only scales better but also provides a deeper insight into the algorithm. This approach may extend to other quantum algorithms. For example, the Quantum Fourier Transform might be visualized through its action on periodic structures rather than individual amplitudes. Quantum phase estimation could be represented as a frequency decomposition process. These approaches remain largely unexplored. Animation is a natural fit for this approach, as continuous motion can directly convey the geometric transformations that underlie these algorithms. Such views are particularly useful in quantum education: learners could focus on why an algorithm works and gain better intuitions rather than following every intermediate state or circuit operation.

Visualizations for Hybrid Computing: As quantum-classical hybrid computing becomes the prevailing approach, visualizations must evolve to represent the interplay between classical optimization loops and quantum circuit execution. Current visualizations treat quantum circuits in isolation. However, future research should visualize the complete hybrid workflow, including classical preprocessing, quantum execution, measurement processing, and parameter updates. For example, Variational Quantum Eigensolver (VQE) methods [131] are a canonical form of hybrid quantum-classical computing that enables novel applications in quantum chemistry and materials science. However, visualizations for VQE methods are largely unexplored in VIS4QC research. Such visualizations would belong to the Productivity × Algorithm Development cell, which is the sparsest region of our problem space.

Utility-First Design: As noted in the quantum software engineering discussion (Section VIII-B2), existing productivity tools improve workflow efficiency but do not target task outcomes directly. Algorithm Development is where this gap matters most. A tool that helps developers not just build circuits faster but also arrive at higher-fidelity algorithms would be far more valuable. However, visualizations that are designed to optimize outcomes directly (e.g., guiding parameter choices or error mitigation strategies) remain underexplored.

X. Conclusion

This survey presents a comprehensive overview of Visualization for Quantum Computing (VIS4QC), an interdisciplinary field at the intersection of visualization and quantum computing research. We systematically collected 69 core PTs and proposed a three-dimensional taxonomy that defines the research space of VIS4QC: *why* (Usage & Purpose) and *what* (QC Abstraction Levels) define the problem space, while *how* (VIS4QC) surveys the solution space.

Examining *why* and *what* together with the source of PTs reveals that academia and industry focus on largely different areas: academia concentrates on visualizing quantum states for learning, while industry provides basic circuit diagrams for productivity. Neither side serves the other's needs. Algorithm Development and Machines & Systems receive the least coverage overall, and productivity tools for algorithm development are nearly absent, despite being where QC delivers practical value. These structural patterns, along with the distinct visualization challenges at each abstraction level, point to concrete future directions. Examples include visual explanations of algorithmic mechanisms, tools for hybrid quantum-classical workflows, visualizations for quantum hardware characteristics, and practical tools for transpilation understanding and visual debugging. We also examined implications for quantum education, where the main challenge is not the absence of individual visualizations but the lack of visual coherence and integration across abstraction levels.

With the centennial of quantum mechanics behind us and QC rapidly advancing, effective visualizations will be necessary to support researchers, developers, and educators. We hope this survey provides visualization researchers with a structured entry point into this emerging field and inspires future contributions that bridge the gap between visualization innovation and practical quantum computing needs.

Acknowledgments

Besides the LLM-assisted methods mentioned in Section IV, we also used LLMs to assist us with proofreading and LaTeX programming. However, we wrote and edited the text content of this paper manually. This project is supported by the Ministry of Education, Singapore, under Academic Research Fund Tier 2 (Proposal ID: T2EP202220049). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997. [Online]. Available: https://doi.org/10.1137/S0097539795293172

[2] P. Arnault, P. Arrighi, S. Herbert, E. Kasnetsi, and T. Li, "A typology of quantum algorithms," 2024, _eprint: 2407.05178. [Online]. Available: https://arxiv.org/abs/2407.05178

[3] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, no. 1, p. 15023, Jan. 2016. [Online]. Available: https://doi.org/10.1038/npjqi.2015.23

[4] A. Ciceri, A. Cottrell, J. Freeland, D. Fry, H. Hirai, P. Intallura, H. Kang, C.-K. Lee, A. Mitra, K. Ohno, D. Pemmaraju, M. Proissl, B. Quanz, D. Rajan, N. Shimada, and K. Yograj, "Enhanced fill probability estimates in institutional algorithmic bond trading using statistical learning algorithms with quantum computers," 2025. [Online]. Available: https://arxiv.org/abs/2509.17715

[5] R. Santagati, A. Aspuru-Guzik, R. Babbush, M. Degroote, L. González, E. Kyoseva, N. Moll, M. Oppel, R. M. Parrish, N. C. Rubin, M. Streif, C. S. Tautermann, H. Weiss, N. Wiebe, and C. Utschig-Utschig, "Drug design on quantum computers," *Nature Physics*, vol. 20, no. 4, pp. 549–557, Apr. 2024. [Online]. Available: https://doi.org/10.1038/s41567-024-02411-5

[6] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala, "Evidence for the utility of quantum computing before fault tolerance," *Nature*, vol. 618, no. 7965, pp. 500–505, Jun. 2023. [Online]. Available: https://doi.org/10.1038/s41586-023-06096-3

[7] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with qiskit," 2024. [Online]. Available: https://arxiv.org/abs/2405.08810

[8] C. Developers, *Cirq*. Zenodo, Aug. 2025. [Online]. Available: https://zenodo.org/doi/10.5281/zenodo.4062499

[9] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. D. Matteo, A. Dusko, T. Garg, D. Guala, A. Hayes, R. Hill, A. Ijaz, T. Isacsson, D. Ittah, S. Jahangiri, P. Jain, E. Jiang, A. Khandelwal, K. Kottmann, R. A. Lang, C. Lee, T. Loke, A. Lowe, K. McKiernan, J. J. Meyer, J. A. Montañez-Barrera, R. Moyard, Z. Niu, L. J. O'Riordan, S. Oud, A. Panigrahi, C.-Y. Park, D. Polatajko, N. Quesada, C. Roberts, N. Sá, I. Schoch, B. Shi, S. Shu, S. Sim, A. Singh, I. Strandberg, J. Soni, A. Száva, S. Thabet, R. A. Vargas-Hernández, T. Vincent, N. Vitucci, M. Weber, D. Wierichs, R. Wiersema, M. Willmann, V. Wong, S. Zhang, and N. Killoran, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," 2022. [Online]. Available: https://arxiv.org/abs/1811.04968

[10] F. Bloch, "Nuclear Induction," *Physical Review*, vol. 70, no. 7-8, pp. 460–474, Oct. 1946. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.70.460

[11] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, Jun. 1982. [Online]. Available: https://doi.org/10.1007/BF02650179

[12] Z. Ashktorab, J. D. Weisz, and M. Ashoori, "Thinking Too Classically: Research Topics in Human-Quantum Computer Interaction," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Glasgow Scotland Uk: ACM, May 2019, pp. 1–12. [Online]. Available: https://dl.acm.org/doi/10.1145/3290605.3300486

[13] H. Kim, M. J. Jeng, and K. N. Smith, "Toward Human-Quantum Computer Interaction: Interface Techniques for Usable Quantum Computing," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, Apr. 2025, pp. 1–18, arXiv:2502.00202 [cs]. [Online]. Available: http://arxiv.org/abs/2502.00202

[14] E. W. Bethel, M. G. Amankwah, J. Balewski, R. Van Beeumen, D. Camps, D. Huang, and T. Perciano, "Quantum Computing and Visualization: A Disruptive Technological Change Ahead," *IEEE Computer Graphics and Applications*, vol. 43, no. 6, pp. 101–111, Nov. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10309230/

[15] E. W. Bethel, R. V. Beeumen, and T. Perciano, "Quantum Computing and Visualization Research Challenges and Opportunities," Jan. 2026, arXiv:2601.07872 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2601.07872

[16] L. Doyle, F. Seifollahi, and C. Singh, "Building bridges in quantum information science education: expert insights to guide framework development for interdisciplinary teaching and evolution of common language," *EPJ Quantum Technology*, vol. 13, no. 1, p. 2, Dec. 2025. [Online]. Available: https://doi.org/10.1140/epjqt/s40507-025-00454-y

[17] W. B. Lane, M. Michelini, and C. Singh, "Editorial: Focused collection: Investigating and improving quantum education through research," *Phys. Rev. Phys. Educ. Res.*, vol. 21, p. 010002, Apr 2025. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevPhysEducRes.21.010002

[18] N. Skult, L. Piispanen, M. Atas, K. Jankiewicz, E. Surer, J. Smed, and Z. C. Seskir, "A Chronicle of Quantum Technologies in Game and Software Development," *IEEE Computer Graphics and Applications*, vol. 44, no. 5, pp. 14–26, 2024.

[19] L. Piispanen, M. Pfaffhauser, J. Wootton, J. Togelius, and A. Kultima, "Defining quantum games," *EPJ Quantum Technology*, vol. 12, no. 1, p. 7, Jan. 2025. [Online]. Available: https://doi.org/10.1140/epjqt/s40507-025-00308-7

[20] N. Skult and J. Smed, "The Marriage of Quantum Computing and Interactive Storytelling," in *Games and Narrative: Theory and Practice*, B. Bostan, Ed. Cham: Springer International Publishing, 2022, pp. 191–206. [Online]. Available: https://doi.org/10.1007/978-3-030-81538-7_13

[21] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. USA: Cambridge University Press, 2011.

[22] H. Y. Wong, "Review of superconducting qubit devices and their large-scale integration," 2026. [Online]. Available: https://arxiv.org/abs/2602.04831

[23] A. Steane, "The ion trap quantum information processor," *Applied Physics B: Lasers and Optics*, vol. 64, no. 6, pp. 623–643, Jun. 1997. [Online]. Available: http://dx.doi.org/10.1007/s003400050225

[24] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak, "Quantum computing with neutral atoms," *Quantum*, vol. 4, p. 327, 2020. [Online]. Available: http://dx.doi.org/10.22331/q-2020-09-21-327

[25] M. Mosca, "Quantum algorithms," 2008. [Online]. Available: https://arxiv.org/abs/0808.0369

[26] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, Jan. 2018. [Online]. Available: http://dx.doi.org/10.1103/RevModPhys.90.015002

[27] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. Van den Nest, "Measurement-based quantum computation," *Nature Physics*, vol. 5, no. 1, pp. 19–26, Jan. 2009. [Online]. Available: https://doi.org/10.1038/nphys1157

[28] C. Weedbrook, S. Pirandola, R. García-Patrón, N. J. Cerf, T. C. Ralph, J. H. Shapiro, and S. Lloyd, "Gaussian quantum information," *Reviews of Modern Physics*, vol. 84, no. 2, pp. 621–669, May 2012. [Online]. Available: http://dx.doi.org/10.1103/RevModPhys.84.621

[29] P. Kaye, R. Laflamme, and M. Mosca, *An introduction to quantum computing*, ser. Oxford scholarship online. Oxford: Oxford University Press, 2020.

[30] M. Lanzagorta and J. Uhlmann, *Quantum Computer Science*, ser. Synthesis Lectures on Quantum Computing. Cham: Springer International Publishing, 2008. [Online]. Available: https://link.springer.com/10.1007/978-3-031-02512-9

[31] N. D. Mermin, "Quantum Computer Science: An Introduction."

[32] H. Y. Wong, *Introduction to Quantum Computing: From a Layperson to a Programmer in 30 Steps*. Cham: Springer International Publishing, 2024. [Online]. Available: https://link.springer.com/10.1007/978-3-031-36985-8

[33] A. M. Dalzell, S. McArdle, M. Berta, P. Bienias, C.-F. Chen, A. Gilyén, C. T. Hann, M. J. Kastoryano, E. T. Khabiboulline, A. Kubica, and et al., *Quantum Algorithms: A Survey of Applications and End-to-end Complexities*. Cambridge University Press, 2025.

[34] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018. [Online]. Available: https://doi.org/10.22331/q-2018-08-06-79

[35] J. Roffe, "Quantum error correction: an introductory guide," *Contemporary Physics*, vol. 60, no. 3, pp. 226–245, 2019. [Online]. Available: https://doi.org/10.1080/00107514.2019.1667078

[36] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, "Digital zero noise extrapolation for quantum error mitigation," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2020, pp. 306–316.

[37] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Phys. Rev. Lett.*, vol. 119, p.

180509, Nov 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.119.180509

[38] S. Ruan, Y. Wang, W. Jiang, Y. Mao, and Q. Guan, "*VACSEN* : A V isualization A pproa c h for Noi s e Awaren e ss in Qua n tum Computing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 462–472, Jan. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9904430/

[39] C. A. Steed, J. Chae, S. Dasgupta, and T. S. Humble, "QVis: A Visual Analytics Tool for Exploring Noise and Errors in Quantum Computing Systems," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Bellevue, WA, USA: IEEE, Sep. 2023, pp. 211–214. [Online]. Available: https://ieeexplore.ieee.org/document/10313853/

[40] J. Chen, Z. Wen, L. Zheng, J. Lu, H. Lu, Y. Ren, and W. Chen, "HammingVis: A visual analytics approach for understanding erroneous outcomes of quantum computing in hamming space," *Graphical Models*, vol. 136, p. 101237, Dec. 2024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1524070324000250

[41] G. DeepMind, "Gemini 2.5 flash and gemini 2.5 flash image model card," https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-5-Flash-Model-Card.pdf, 2025, accessed: 2025-11-24.

[42] D. Li, Z. Zhang, X. Xu, and M. Hosseini, "Quantum Game Club: Engaging with Quantum Computing Through Interactive Learning," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Montreal, QC, Canada: IEEE, Sep. 2024, pp. 143–154. [Online]. Available: https://ieeexplore.ieee.org/document/10821155/

[43] M. Karunathilaka, S. Ruan, L.-P. Yuan, J. Li, Z. Liang, K. Athapaththu, Q. Guan, and Y. Wang, "Intuit: Explain Quantum Computing Concepts via AR-based Analogy," in *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, Apr. 2025, pp. 1–8, arXiv:2503.00835 [cs]. [Online]. Available: http://arxiv.org/abs/2503.00835

[44] R. Fickler, M. Krenn, R. Lapkiewicz, S. Ramelow, and A. Zeilinger, "Real-Time Imaging of Quantum Entanglement," *Scientific Reports*, vol. 3, no. 1, p. 1914, May 2013. [Online]. Available: https://www.nature.com/articles/srep01914

[45] H. Zhao, G. W. Bryant, W. Griffin, J. E. Terrill, and J. Chen, "Evaluating Glyph Design for Showing Large-Magnitude-Range Quantum Spins," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 4, pp. 1868–1884, Apr. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10002316/

[46] T. Tomono, K. Tsujimura, and T. Godo, "Quantum Kernels for Difficult Visual Discrimination," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Bellevue, WA, USA: IEEE, Sep. 2023, pp. 262–263. [Online]. Available: https://ieeexplore.ieee.org/document/10313789/

[47] X.-B. Nguyen, H.-Q. Nguyen, S. Y.-C. Chen, S. U. Khan, H. Churchill, and K. Luu, "QClusformer: A Quantum Transformer-based Framework for Unsupervised Visual Clustering," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Montreal, QC, Canada: IEEE, Sep. 2024, pp. 347–352. [Online]. Available: https://ieeexplore.ieee.org/document/10821154/

[48] S. N. Walck and N. C. Hansell, "Characterization and visualization of the state and entanglement of two spins," *European Journal of Physics*, vol. 22, no. 4, p. 343, Jul. 2001. [Online]. Available: https://doi.org/10.1088/0143-0807/22/4/309

[49] G. F. Viamontes, M. Rajagopalan, I. L. Markov, and J. P. Hayes, "Gate-level simulation of quantum circuits," in *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 295–301. [Online]. Available: https://doi.org/10.1145/1119772.1119829

[50] I. G. Karafyllidis, "Visualization of the Quantum Fourier Transform Using a Quantum Computer Simulator," *Quantum Information Processing*, vol. 2, no. 4, pp. 271–288, Aug. 2003. [Online]. Available: https://link.springer.com/10.1023/B:QINP.0000020076.36114.13

[51] B. Coecke and R. Duncan, "Interacting quantum observables," in *Automata, Languages and Programming*, L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 298–310.

[52] J. B. Altepeter, E. R. Jeffrey, M. Medic, and P. Kumar, "Multiple-Qubit Quantum State Visualization," in *Conference on Lasers and Electro-Optics/International Quantum Electronics Conference*. Baltimore, Maryland: OSA, 2009, p. IWC1. [Online]. Available: https://opg.optica.org/abstract.cfm?uri=IQEC-2009-IWC1

[53] H. Mäkelä and A. Messina, "*N* -qubit states as points on the Bloch sphere," *Physica Scripta*, vol. T140, p. 014054, Sep. 2010. [Online]. Available: https://iopscience.iop.org/article/10.1088/0031-8949/2010/T140/014054

[54] M. Galambos and S. Imre, "New Method for Representation of Multi-qbit Systems Using Fractals," in *Proceedings of the International Conference on Quantum Nano and Micro Technologies (ICQNM)*. Nice/Saint Laurent du Var, France: IARIA, Aug. 2011, pp. 52–56, iSSN: 2308-3530. [Online]. Available: http://personales.upv.es/thinkmind/dl/conferences/icqnm/icqnm_2011/icqnm_2011_3_30_80175.pdf

[55] ——, "Visualizing the Effects of Measurements and Logic Gates On Multi-Qubit Systems Using Fractal Representation," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 1-2, pp. 1–10, 2012. [Online]. Available: https://www.thinkmind.org/library/SysMea/SysMea_v5_n12_2012/sysmea_v5_n12_2012_1.html

[56] C. Tahan, "Quantum mechanics is intuitive," APS March Meeting, San Antonio, Texas, March 2015. [Online]. Available: https://tahan.com/charlie/talks/2015MMquantumintuitiontalk.pdf

[57] V. N. Chernega, O. V. Man'ko, and V. I. Man'ko, "Triangle Geometry of the Qubit State in the Probability Representation Expressed in Terms of the Triada of Malevich's Squares," *Journal of Russian Laser Research*, vol. 38, no. 2, pp. 141–149, Mar. 2017. [Online]. Available: http://link.springer.com/10.1007/s10946-017-9628-6

[58] A. Parakh, M. Subramaniam, and E. Ostler, "QuaSim: A virtual quantum cryptography educator," in *2017 IEEE International Conference on Electro Information Technology (EIT)*. Lincoln, NE, USA: IEEE, May 2017, pp. 600–605. [Online]. Available: http://ieeexplore.ieee.org/document/8053434/

[59] Z. Tao, Y. Pan, A. Chen, and L. Wang, "ShorVis: A Comprehensive Case Study of Quantum Computing Visualization," in *2017 International Conference on Virtual Reality and Visualization (ICVRV)*. Zhengzhou, China: IEEE, Oct. 2017, pp. 360–365. [Online]. Available: https://ieeexplore.ieee.org/document/8719175/

[60] S. Lin, J. Hao, and L. Sun, "QuFlow: Visualizing Parameter Flow in Quantum Circuits for Understanding Quantum Computation," in *2018 IEEE Scientific Visualization Conference (SciVis)*. Berlin, Germany: IEEE, Oct. 2018, pp. 37–41. [Online]. Available: https://ieeexplore.ieee.org/document/8823602/

[61] J.-B. Lamy, "Dynamic Software Visualization of Quantum Algorithms with Rainbow Boxes:," in *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, 2019, pp. 155–163. [Online]. Available: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007247801550163

[62] M. Miller and D. Miller, "GraphStateVis: Interactive Visual Analysis of Qubit Graph States and their Stabilizer Groups," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Broomfield, CO, USA: IEEE, Oct. 2021, pp. 378–384. [Online]. Available: https://ieeexplore.ieee.org/document/9605348/

[63] I. Arawjo, A. DeArmas, M. Roberts, S. Basu, and T. Parikh, "Notational Programming for Notebook Environments: A Case Study with Quantum Circuits," in *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. Bend OR USA: ACM, Oct. 2022, pp. 1–20. [Online]. Available: https://dl.acm.org/doi/10.1145/3526113.3545619

[64] A. Jordon, A. Hawkins-Seagram, and U. Stege, "Implementation and Viusalization of Quantum Walks," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Broomfield, CO, USA: IEEE, Sep. 2022, pp. 796–798. [Online]. Available: https://ieeexplore.ieee.org/document/9951201/

[65] A. Suau, M. Vuffray, A. Y. Lokhov, L. Cincio, and C. Coffrin, "Vector Field Visualization of Single-Qubit State Tomography," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Broomfield, CO, USA: IEEE, Sep. 2022, pp. 528–534. [Online]. Available: https://ieeexplore.ieee.org/document/9951204/

[66] A. Suau, G. Staffelbach, and A. Todri-Sanial, "qprof: A gprof-Inspired Quantum Profiler," *ACM Transactions on Quantum Computing*, vol. 4, no. 1, pp. 1–28, Mar. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3529398

[67] A. Jordon, A. Hawkins-Seagram, S. Norrie, J. Ossorio, and U. Stege, "QWalkVis: Quantum Walks Visualization Application," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Bellevue, WA, USA: IEEE, Sep. 2023, pp. 87–93. [Online]. Available: https://ieeexplore.ieee.org/document/10313735/

[68] Z. Wen, Y. Liu, S. Tan, J. Chen, M. Zhu, D. Han, J. Yin, M. Xu, and W. Chen, "Quantivine: A Visualization Approach for Large-scale Quantum Circuit Representation and Analysis," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–11, 2023, arXiv:2307.08969 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2307.08969

[69] P. Senapati, T. M. Athawale, D. Pugmire, and Q. Guan, "Advancing Comprehension of Quantum Application Outputs: A Visualization Technique," in *Proceedings of the 2023 International Workshop on Quantum Classical Cooperative*, ser. QCCC '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 25–28, event-place: Orlando, FL, USA. [Online]. Available: https://doi.org/10.1145/3588983.3596689

[70] S. Ruan, R. Yuan, Q. Guan, Y. Lin, Y. Mao, W. Jiang, Z. Wang, W. Xu, and Y. Wang, "*VENUS* : A Geometrical Representation for Quantum State Visualization," *Computer Graphics Forum*, vol. 42, no. 3, pp. 247–258, Jun. 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/cgf.14827

[71] J. Bley, E. Rexigel, A. Arias, N. Longen, L. Krupp, M. Kiefer-Emmanouilidis, P. Lukowicz, A. Donhauser, S. Küchemann, J. Kuhn, and A. Widera, "Visualizing entanglement in multiqubit systems," *Phys. Rev. Res.*, vol. 6, no. 2, p. 023077, Apr. 2024, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.6.023077

[72] Y.-H. Chou, Y.-C. Jiang, S.-Y. Kuo, and S.-Y. Kung, "Bridging the Quantum Education Gap: Hands-on Visualization Projects for Quantum Search Algorithm," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Montreal, QC, Canada: IEEE, Sep. 2024, pp. 112–121. [Online]. Available: https://ieeexplore.ieee.org/document/10821097/

[73] M. Khan, P. J. Nair, and O. Di Matteo, "CircInspect: Integrating Visual Circuit Analysis, Abstraction, and Real-Time Development in Quantum Debugging," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Montreal, QC, Canada: IEEE, Sep. 2024, pp. 1000–1006. [Online]. Available: https://ieeexplore.ieee.org/document/10821435/

[74] H. Kim and K. N. Smith, "Interaction Techniques for User-Friendly Interfaces for Gate-Based Quantum Computing," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Montreal, QC, Canada: IEEE, Sep. 2024, pp. 482–483. [Online]. Available: https://ieeexplore.ieee.org/document/10821108/

[75] S. Norrie, A. Estey, H. Müller, and U. Stege, "QGrover: Teaching Grover's Algorithm Through Visual Exploration," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Montreal, QC, Canada: IEEE, Sep. 2024, pp. 17–24. [Online]. Available: https://ieeexplore.ieee.org/document/10821077/

[76] P. Debus, S. Issel, and K. Tscharke, "Quantum Machine Learning Playground," *IEEE Computer Graphics and Applications*, vol. 44, no. 5, pp. 40–53, Sep. 2024, arXiv:2507.17931 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2507.17931

[77] S. Norrie, A. Estey, H. Müller, and U. Stege, "QNotation: A Visual Browser-Based Notation Translator for Learning Quantum Computing," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Montreal, QC, Canada: IEEE, Sep. 2024, pp. 25–33. [Online]. Available: https://ieeexplore.ieee.org/document/10821137/

[78] S. Ruan, Q. Guan, P. Griffin, Y. Mao, and Y. Wang, "QuantumEyes: Towards Better Interpretability of Quantum Circuits," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 9, pp. 6321–6333, Sep. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10319321/

[79] D. Brahmbhatt, Y. Xu, N. Vora, L. Chen, N. Fruitwala, G. Huang, Q. Ji, and P. Nguyen, "An open-source data storage and visualization platform for collaborative qubit control," *Scientific Reports*, vol. 14, no. 1, p. 22703, Sep. 2024. [Online]. Available: https://www.nature.com/articles/s41598-024-72584-9

[80] J. Chae, C. A. Steed, and T. S. Humble, "Visual Analytics of Performance of Quantum Computing Systems and Circuit Optimization," in *2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. Knoxville, TN, USA: IEEE, Jul. 2024, pp. 613–618. [Online]. Available: https://ieeexplore.ieee.org/document/10682768/

[81] R. Seidel, R. Zander, M. Petrič, N. Steinmann, D. Q. Liu, N. Tcholtchev, and M. Hauswirth, "Quantum backtracking in qrisp applied to sudoku problems," 2024. [Online]. Available: https://arxiv.org/abs/2402.10060

[82] S. Ruan, Z. Liang, Q. Guan, P. Griffin, X. Wen, Y. Lin, and Y. Wang, "*VIOLET* : V isual Analyt i cs f o r Exp l ainable Quantum N e ural Ne t works," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 6, pp. 2862–2874, Jun. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10507082/

[83] D. Huber and S. J. Glaser, "BEADS: a canonical visualization of quantum states for applications in quantum information processing," *New Journal of Physics*, vol. 27, no. 9, p. 094509, Sep. 2025. [Online]. Available: https://iopscience.iop.org/article/10.1088/1367-2630/ae0514

[84] M. J. McGuffin and J.-M. Robert, "Visualizing Quantum Circuits: State Vector Difference Highlighting and the Half-Matrix," Oct. 2025, arXiv:2510.00895 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2510.00895

[85] S. J'oczik, B. Kecsk'es, A. Kov'acs, O. K'alm'an, and Z. Zimbor'as, "Qcanvas: An Interactive Browser-Based Quantum Circuit Design and Simulation Platform," in *2025 IEEE International Conference on Quantum Software (QSW)*. Helsinki, Finland: IEEE, Jul. 2025, pp. 137–140. [Online]. Available: https://ieeexplore.ieee.org/document/11134324/

[86] H. Patel, "Simulation and Visualization of Quantum Algorithms in Wolfram Mathematica: An Interactive Toolkit for Quantum Computing Education," *International Journal of Advanced Multidisciplinary Research and Studies*, vol. 5, no. 1, pp. 1204–1208, 2025. [Online]. Available: https://www.multiresearchjournal.com/arclist/list-2025.5.1/id-3785

[87] A. Heim, T. Lang, A. Gall, E. Gröller, and C. Heinzl, "Quantum Image Visualizer: Visual Debugging of Quantum Image Processing Circuits," Apr. 2025, arXiv:2504.09902 [cs]. [Online]. Available: http://arxiv.org/abs/2504.09902

[88] Z. Wen, J. Chen, Y. Lu, S. Tan, J. Yin, M. Zhu, and W. Chen, " QuRAFT: Enhancing Quantum Algorithm Design by Visual Linking between Mathematical Concepts and Quantum Circuits ," *IEEE Transactions on Visualization & Computer Graphics*, no. 01, pp. 1–15, Dec. 2025. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TVCG.2025.3642559

[89] F. Schinkel, "state-o-gram –A Novel 2D Visualization for Quantum States," Aug. 2025, arXiv:2508.18390 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2508.18390

[90] G. Sanderson, "But what is quantum computing? (Grover's Algorithm)," https://www.3blue1brown.com/lessons/grover, Apr. 2025, 3Blue1Brown. Accessed: 2026-01-23.

[91] T. Scruby, "Visualising quantum product codes," 2025. [Online]. Available: https://arxiv.org/abs/2507.11577

[92] P. Senapati, Q. Guan, D. Pugmire, C. C. Lu, and T. M. Athawale, "Visualization of Noisy and Less Noisy Computational Basis States in Quantum Computing," in *2025 IEEE International Conference on Quantum Software (QSW)*. Helsinki, Finland: IEEE, Jul. 2025, pp. 12–21. [Online]. Available: https://ieeexplore.ieee.org/document/11134352/

[93] S. Ruan, F. Liang, R. Ramakrishna, C. Ren, R. Yan, Q. Guan, J. Li, and Y. Wang, "Towards explainable quantum ai: Informing the encoder selection of quantum neural networks via visualization," 2025. [Online]. Available: https://arxiv.org/abs/2512.14181

[94] J. Johansson, P. Nation, and F. Nori, "Qutip: An open-source python framework for the dynamics of open quantum systems," *Computer Physics Communications*, vol. 183, no. 8, pp. 1760–1772, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465512000835

[95] ——, "Qutip 2: A python framework for the dynamics of open quantum systems," *Computer Physics Communications*, vol. 184, no. 4, pp. 1234–1240, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465512003955

[96] N. Lambert, E. Gigu'ere, P. Menczel, B. Li, P. Hopf, G. Su'arez, M. Gali, J. Lishman, R. Gadhvi, R. Agarwal, A. Galicia, N. Shammah, P. Nation, J. R. Johansson, S. Ahmed, S. Cross, A. Pitchford, and F. Nori, "Qutip 5: The quantum toolbox in Python," *Physics Reports*, vol. 1153, pp. 1–62, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0370157325002704

[97] R. S. Smith, M. J. Curtis, and W. J. Zeng, "A practical quantum instruction set architecture," 2016.

[98] Amazon Web Services, "Amazon braket," https://aws.amazon.com/braket/, 2020, accessed: 2026-01-22.

[99] D. K. Tuckett, "Tailoring surface codes: Improvements in quantum error correction with biased noise," Ph.D. dissertation, University of Sydney, 2020, (qecsim: https://github.com/qecsim/qecsim).

[100] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "TKET: A Retargetable Compiler for NISQ devices," *Quantum Science and Technology*, vol. 6, Nov. 2020.

[101] S. Efthymiou, S. Ramos-Calderer, C. Bravo-Prieto, A. Pérez-Salinas, D. García-Martín, A. Garcia-Saez, J. I. Latorre, and S. Carrazza, "Qibo: a framework for quantum simulation with hardware acceleration," *Quantum Science and Technology*, vol. 7, no. 1, p. 015018, dec 2021. [Online]. Available: https://doi.org/10.1088/2058-9565/ac39f5

[102] C. Gidney, "Stim: a fast stabilizer circuit simulator," *Quantum*, vol. 5, p. 497, Jul. 2021. [Online]. Available: https://doi.org/10.22331/q-2021-07-06-497

[103] R. J. Hill, H. Gupta, R. Young, and K. Setia, "qBraid-SDK: Platform-agnostic quantum runtime framework." Jan. 2025. [Online]. Available: https://github.com/qBraid/qBraid

[104] J.-S. Kim, A. McCaskey, B. Heim, M. Modani, S. Stanwyck, and T. Costa, "Cuda quantum: The platform for integrated quantum-classical computing," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–4.

[105] K. Singhal, K. Hietala, S. Marshall, and R. Rand, "Q# as a quantum algorithmic language," *Electronic Proceedings in Theoretical Computer Science*, vol. 394, pp. 170–191, Nov. 2023. [Online]. Available: http://dx.doi.org/10.4204/EPTCS.394.10

[106] E. Huang, A. Pesah, C. T. Chubb, M. Vasmer, and A. Dua, "Tailoring three-dimensional topological codes for biased noise," *PRX Quantum*, vol. 4, p. 030338, Sep 2023. [Online]. Available: https://link.aps.org/doi/10.1103/PRXQuantum.4.030338

[107] P. Weinberg, K.-H. Wu, J. Long, and X.-z. R. Luo, "Queracomputing/bloqade-python: v0.15.11," May 2024. [Online]. Available: https://doi.org/10.5281/zenodo.11114110

[108] D. Seitz, N. Heim, J. P. Moutinho, R. Guichard, V. Abramavicius, A. Wennersteen, G.-J. Both, A. Quelle, C. de Groot, G. V. Velikova, V. E. Elfving, and M. Dagrada, "Qadence: a differentiable interface for digital-analog programs." 2024. [Online]. Available: https://github.com/pasqal-io/qadence

[109] R. Seidel, S. Bock, R. Zander, M. Petrič, N. Steinmann, N. Tcholtchev, and M. Hauswirth, "Qrisp: A framework for compilable high-level programming of gate-based quantum computers," 2024. [Online]. Available: https://arxiv.org/abs/2406.14792

[110] M. P. Harrigan, T. Khattar, C. Yuan, A. Peduri, N. Yosri, F. D. Malone, R. Babbush, and N. C. Rubin, "Expressing and Analyzing Quantum Algorithms with Qualtran," 2024. [Online]. Available: https://arxiv.org/abs/2409.04643

[111] Horizon Quantum Computing, "Triple alpha: Integrated development environment for quantum computing," https://www.horizonquantum.com/triple-alpha, 2025, accessed: 2026-01-22.

[112] R. Wille, L. Burgholzer, and M. Artner, "Visualizing Decision Diagrams for Quantum Computing (Special Session Summary)," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Grenoble, France: IEEE, Feb. 2021, pp. 768–773. [Online]. Available: https://ieeexplore.ieee.org/document/9474236/

[113] P. Migdał, K. Jankiewicz, P. Grabarz, C. Decaroli, and P. Cochin, "Visualizing quantum mechanics in an interactive simulation – Virtual Lab by Quantum Flytrap," *Optical Engineering*, vol. 61, no. 08, Jun. 2022, arXiv:2203.13300 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2203.13300

[114] F. Chiarello and M. G. Castellano, "Board Games and Board Game Design as Learning Tools for Complex Scientific Concepts: Some Experiences," *International Journal of Game-Based Learning*, vol. 6, no. 2, pp. 1–14, Apr. 2016. [Online]. Available: https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/IJGBL.2016040101

[115] D. Carberry, J. Shou Neergaard-Nielson, E. Van Nieuwenburg, and M. Andersson, "Board Games to teach Quantum Technologies to Engineers: A Peer Instruction Approach," *SSRN Electronic Journal*, 2022. [Online]. Available: https://www.ssrn.com/abstract=4258302

[116] A. Goff, "Quantum tic-tac-toe: A teaching metaphor for superposition in quantum mechanics," *American Journal of Physics*, vol. 74, no. 11, pp. 962–973, Nov. 2006. [Online]. Available: https://pubs.aip.org/ajp/article/74/11/962/864424/Quantum-tic-tac-toe-A-teaching-metaphor-for

[117] J. Watrous. (2025) Bloch sphere. IBM Quantum Learning. Part of the course "General formulation of quantum information". [Online]. Available: https://quantum.cloud.ibm.com/learning/en/courses/general-formulation-of-quantum-information/density-matrices/bloch-sphere

[118] IBM Quantum, "IBM Quantum Composer," 2025, accessed: 2026-01-18. [Online]. Available: https://quantum.cloud.ibm.com/composer

[119] M. Schuld, R. Sweke, and J. J. Meyer, "Effect of data encoding on the expressive power of variational quantum-machine-learning models," *Phys. Rev. A*, vol. 103, p. 032430, Mar 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.103.032430

[120] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219. [Online]. Available: https://doi.org/10.1145/237814.237866

[121] B. Coecke and A. Kissinger, *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

[122] A. Kissinger and J. van de Wetering, *Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation*. Preprint, 2024.

[123] B. Coecke and S. Gogioso, *Quantum in Pictures: A New Way to Understand the Quantum World*, 1st ed. Cambridge Quantum, February 2023.

[124] R. Wille, S. Hillmich, and L. Burgholzer, "Decision Diagrams for Quantum Computing," in *Design Automation of Quantum Computers*, R. O. Topaloglu, Ed. Cham: Springer International Publishing, 2023, pp. 1–23. [Online]. Available: https://link.springer.com/10.1007/978-3-031-15699-1_1

[125] QuTube. Decoherence on a Bloch sphere. QuTech. [Online]. Available: https://www.qutube.nl/quantum-computer-12/decoherence-on-a-bloch-sphere

[126] E. R. Tufte, *The visual display of quantitative information*. USA: Graphics Press, 1986.

[127] T. Munzner, *Visualization Analysis and Design*, ser. AK Peters Visualization Series. Boca Raton, FL: CRC Press, 2014.

[128] G. Passante and A. Kohnle, "Interactive homework to support student learning of measurement uncertainty in quantum mechanics," *Phys. Rev. Phys. Educ. Res.*, vol. 20, p. 020131, 2024.

[129] V. Borish and H. J. Lewandowski, "Affordances and challenges of incorporating a remote, cloud-accessible quantum experiment into undergraduate courses," *Phys. Rev. Phys. Educ. Res.*, vol. 21, p. 010133, 2025.

[130] E. B. Moore, J. M. Chamberlain, R. Parson, and K. K. Perkins, "Phet interactive simulations: Transformative tools for teaching chemistry," *Journal of Chemical Education*, vol. 91, no. 8, pp. 1191–1197, Aug 2014. [Online]. Available: https://doi.org/10.1021/ed4005084

[131] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, and J. Tennyson, "The variational quantum eigensolver: A review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022, the Variational Quantum Eigensolver: a review of methods and best practices. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0370157322003118

APPENDIX
PROMPTS AND VENUE ABBREVIATIONS

A. Prompts

B. List of Venues

TABLE A.1
ABBREVIATIONS AND FULL NAMES OF VENUES.

Abbr.	Full Name
ACM CHI	ACM Conference on Human Factors in Computing Systems
ACM ICSE	ACM International Conference on Software Engineering
ACM TOPLAS	ACM Transactions on Programming Languages and Systems
ACM TOSEM	ACM Transactions on Software Engineering and Methodology
ACM TQC	ACM Transactions on Quantum Computing
ACM UIST	ACM Symposium on User Interface Software and Technology
arXiv	arXiv preprint
ASP-DAC	Asia and South Pacific Design Automation Conference
CGF	Computer Graphics Forum
CHI PLAY	ACM Annual Symposium on Computer-Human Interaction in Play
CLEO/IQEC	Conference on Lasers and Electro-Optics/International Quantum Electronics Conference
CPC	Computer Physics Communications
DAC	IEEE/ACM Design Automation Conference
DATE	Design, Automation and Test in Europe Conference
Entropy	Entropy (MDPI)
EPTCS	Electronic Proceedings in Theoretical Computer Science
ESEC/FSE	Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering
Eur. J. Phys.	European Journal of Physics
EuroVis	Eurographics Conference on Visualization
Graph. Models	Graphical Models
GRAPP	International Conference on Computer Graphics Theory and Applications
ICALP	International Colloquium on Automata, Languages and Programming
ICQNM	International Conference on Quantum, Nano and Micro Technologies
ICVRV	International Conference on Virtual Reality and Visualization
IEEE CG&A	IEEE Computer Graphics and Applications
IEEE EIT	IEEE International Conference on Electro/Information Technology
IEEE QCE	IEEE International Conference on Quantum Computing and Engineering
IEEE QSW	IEEE International Conference on Quantum Software
IEEE TSE	IEEE Transactions on Software Engineering
IEEE VIS	IEEE Visualization Conference (including InfoVis, VAST, SciVis)
IEEE/ACM ASE	IEEE/ACM International Conference on Automated Software Engineering
IJGBL	International Journal of Game-Based Learning
ISVLSI	IEEE Computer Society Annual Symposium on VLSI
J. Russ. Laser Res.	Journal of Russian Laser Research
Nat. Comput. Sci.	Nature Computational Science
New J. Phys.	New Journal of Physics
Opt. Eng.	Optical Engineering
PacificVis	IEEE Pacific Visualization Symposium
Phys. Rep.	Physics Reports
Phys. Rev.	Physical Review
Phys. Rev. Res.	Physical Review Research
Phys. Scr.	Physica Scripta
PRX Quantum	PRX Quantum
QCCC	ACM International Workshop on Quantum Classical Cooperative
QIP	Quantum Information Processing
QST	Quantum Science and Technology
Quantum	Quantum
Sci. Rep.	Scientific Reports
SciVis	IEEE Scientific Visualization Conference
SSRN	Social Science Research Network
SysMea	International Journal On Advances in Systems and Measurements
TVCG	IEEE Transactions on Visualization and Computer Graphics
Zenodo	Zenodo

Listing 2. Prompt Template for our summarization agent.

```
<paper pdf file>

Help me summarize the paper please.

## Requirements
* You should not copy the abstract of the paper.
* Give a detailed summary of the papers that
    includes:
  * General background: the research background and
     related work
  * Motivation: the motivation or the problem to be
     solved
  * Is it a new problem or old problem?
    * If it is a new problem -> justify why it is
       important and interesting
    * If it is an old problem -> what are the current
       solutions and why they are not good enough?
  * any new techniques introduced
  * major contributions
  * future work and limitations
  * anything else you think is important
  * You should write fluent sentences and paragraphs
      to make it more readable, not bullet points.
* The format of the summary should use Markdown with
    a frontmatter section that includes the title,
    authors and abstract. Any math expressions
    should be rendered in LaTeX.

## Example

Below is an example of summary that showcases the
    required format. Do not include the backticks
    ``` when you write the summary.

```
---
title: Some paper title
author: First author name
author: Second author name
author: Third author name
author: more authors...
abstract: the abstract from the paper, just copy it
    from the paper, do not summarize the paper
    yourself

This paper is about ... It has some latex symbols
    like a, b, c and equations like
$$
a + b = c
$$

```
```

Listing 3. Prompt Template for our SDK search agent.

```
For the quantum SDK <SDK-name>, help me find out all
     functionalities of it that are related to
    visualization, graphical interfaces and plotting
    . Give me a list of functionalities and links to
     corresponding website/documentation if
    available.
```
```