

A Quantum Fourier Transform Approach for Image Alignment [†]

Alexander Geng ^{*,‡}  and Ali Moghiseh [‡] 

Fraunhofer Institute for Industrial Mathematics ITWM, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany; ali.moghiseh@itwm.fraunhofer.de

* Correspondence: alexander.geng@itwm.fraunhofer.de; Tel.: +49-631-31600-4183

[†] Presented at the 15th International Scientific Conference TechSys 2026—Engineering, Technologies and Systems, Plovdiv, Bulgaria, 14–16 May 2026.

[‡] These authors contributed equally to this work.

Abstract

This work explores the practical application of the Quantum Fourier Transform (QFT) for image alignment, focusing on the estimation of rotation angles in grayscale images containing structured line patterns and text. Motivated by the need to demonstrate quantum computing's potential in addressing real-world image processing tasks, we develop a hybrid approach that combines classical pre-processing with quantum computation. We demonstrate that QFT can be applied to a concrete image processing task, illustrating the practical utility of quantum computing beyond theoretical examples. A classical baseline using the Fast Fourier Transform (FFT) is implemented via the ToolIP framework, achieving fast and accurate angle estimation. In parallel, a quantum version replaces the FFT with a simulated QFT on IBM's Qiskit platform, using the Quantum Image Encoding Probability scheme to reduce qubit requirements. The classical method delivers results in milliseconds, while the quantum implementation, constrained by simulation and encoding overhead, demands significantly greater computational effort. Nonetheless, our findings show that QFT-based angle estimation is feasible and can yield results comparable to classical techniques. This study demonstrates the applicability of quantum algorithms to real-world image processing and underscores both the theoretical promise of QFT and the practical limitations faced in the current Noisy Intermediate-Scale Quantum (NISQ) era.

Keywords: quantum image processing; quantum Fourier transform; image alignment; hybrid quantum-classical algorithm; quantum probability image encoding; NISQ devices

1. Introduction

The Quantum Fourier Transform (QFT) [1,2] is the quantum analog of the Discrete Fourier Transform (DFT) [3] and represents a fundamental linear transformation on qubits. It plays a central role in a wide range of quantum algorithms, including Shor's algorithm for factoring and discrete logarithms [4,5], quantum phase estimation [6,7], hidden subgroup problems, and quantum linear system solvers like the HHL algorithm [8]. These algorithms demonstrate the transformative potential of quantum computing in addressing problems previously considered intractable for classical methods.

Despite this theoretical promise, the application of the QFT in practical domains such as image processing remains relatively underexplored [9,10]. In particular, image alignment tasks—such as estimating rotation angles in structured visual data—are traditionally addressed using classical spectral methods like the Fast Fourier Transform (FFT) [11,12]. However, as quantum hardware matures, it becomes increasingly important to investigate



Academic Editors: Nikola G. Shakev,
Valyo N. Nikolov, Nikolay R.
Kakanakov and Sevil A.
Ahmed-Shieva

Published: 1 August 2026

Copyright: © 2026 by the authors.
Licensee MDPI, Basel, Switzerland.
This article is an open access article
distributed under the terms and
conditions of the [Creative Commons
Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

whether QFT-based approaches can offer alternative pathways to processing and analyzing image features in the frequency domain [13].

In this paper, we present a novel approach to image rotation estimation that leverages the QFT in a hybrid quantum-classical framework. Our method integrates classical pre-processing with quantum computing to analyze gray value images containing structured line patterns and text. The image data is encoded using a quantum image representation scheme, and processed via a simulated QFT implementation on IBM's Qiskit platform. This work follows the formalism introduced by Nielsen et al. [6], and aims to assess the viability of QFT as a computational tool under the constraints of Noisy Intermediate-Scale Quantum (NISQ) devices.

2. Classical Fourier Transform

One of the most useful techniques to solve complex problems in mathematics is to transform the equation into another type of equation or into another space, where the solution is known. The Discrete Fourier Transform is one example of such a transform. It finds applications in various domains, like for example in the signal processing or image processing.

In image processing, the Fourier Transform can be used for several applications, such as image analysis, filtering, reconstruction, and compression [11]. For instance, we can use the Fourier Transform to remove periodic noise from images by identifying and filtering out unwanted frequency components in the frequency domain, resulting in a cleaner and clearer image.

For the definition of the Discrete Fourier Transform, we start with an input vector $x \in \mathbb{C}^N$. As output, we get again a vector $y \in \mathbb{C}^N$ with

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N}, \quad (1)$$

where $k \in \{0, \dots, N-1\}$.

3. Quantum Fourier Transform

The Quantum Fourier Transform (QFT) is the quantum analogue of the classical Discrete Fourier Transform (DFT) and plays a pivotal role in numerous quantum algorithms. By operating on the amplitudes of quantum states, the QFT transforms quantum information into its frequency-domain representation. It serves as a foundational component in key quantum computing applications, including integer factorization, phase estimation, and the solution of hidden subgroup problems. Unlike the classical DFT, the QFT can, under ideal conditions, be implemented with exponentially fewer computational steps, highlighting its potential for significant efficiency gains in quantum information processing.

We define the QFT on an orthonormal basis $|0\rangle, \dots, |N-1\rangle$ as a linear operator which acts on the basis states as:

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle. \quad (2)$$

We have for an arbitrary state $|\phi\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$

$$\begin{aligned} QFT |\phi\rangle &= \sum_{j=0}^{N-1} x_j QFT |j\rangle = \sum_{j=0}^{N-1} x_j \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle \\ &= \sum_{k=0}^{N-1} \left(\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N} \right) |k\rangle = \sum_{k=0}^{N-1} y_k |k\rangle, \end{aligned} \quad (3)$$

where y_k is the Discrete Fourier Transform of the amplitudes x_j .

4. Circuit for Quantum Fourier Transform

Let n be the number of qubits used for the quantum circuit, $N = 2^n$, and $|0\rangle, \dots, |2^n - 1\rangle$ the computational basis for the quantum computer. For further development, we write a state $|k\rangle$ using its binary representation $|k\rangle = |k_0 k_1 \dots k_{n-1}\rangle$, where $k = k_0 2^{n-1} + k_1 2^{n-2} + \dots + k_{n-1} 2^0$. Then, we get for $|q\rangle = |q_0 \dots q_{n-1}\rangle := QFT |j\rangle$

$$|q\rangle = \frac{(|0\rangle + e^{2\pi i j_{n-1} 2^{-1}} |1\rangle)}{\sqrt{2}} \otimes \frac{(|0\rangle + e^{2\pi i (j_{n-1} 2^{-1} + j_{n-2} 2^{-2})} |1\rangle)}{\sqrt{2}} \otimes \dots \otimes \frac{(|0\rangle + e^{2\pi i j_1 2^{-n}} |1\rangle)}{\sqrt{2}} \quad (4)$$

$$= |q_0\rangle \otimes |q_1\rangle \otimes \dots \otimes |q_{n-1}\rangle.$$

This representation gives us a better intuition of how to construct an efficient quantum circuit for computing the Fourier transform and gives proof of why the Quantum Fourier Transform is unitary.

We iteratively construct the circuit while using the characteristics of the exponential function.

$$e^{2\pi i j_{n-1} 2^{-1}} = \begin{cases} 1, & j_{n-1} = 0 \\ -1, & j_{n-1} = 1. \end{cases} \quad (5)$$

Thus, we get

$$|q_0\rangle = \frac{(|0\rangle + e^{2\pi i j_{n-1} 2^{-1}} |1\rangle)}{\sqrt{2}} = \begin{cases} \frac{|0\rangle + |1\rangle}{\sqrt{2}}, & j_{n-1} = 0 \\ \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & j_{n-1} = 1. \end{cases} \quad (6)$$

To realize this on the quantum computer, we can directly use the Hadamard gate. For the second qubit, we will have the same procedure and use the multiplicative property of the exponential function

$$e^{2\pi i (j_{n-1} 2^{-1} + j_{n-2} 2^{-2})} = e^{\pi i j_{n-1}} e^{2\pi i j_{n-2} / 4}. \quad (7)$$

By that and by using the unitary transformation

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{pmatrix} \quad (8)$$

with $k = 2$, we have

$$|q_1\rangle = R_2 H |j_{n-2} j_{n-1}\rangle. \quad (9)$$

Analogously, we continue for all other qubits until we reach

$$|q_{n-1}\rangle = R_{n-1} R_{n-2} \dots R_2 H |j_2 \dots j_{n-2} j_{n-1}\rangle. \quad (10)$$

We can use, for example, a Phase gate

$$P(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}. \quad (11)$$

with $\theta = 2\pi i / 2^k$ to realize the unitary transformation R_k . Other combinations of quantum gates can also give us the unitary transformation R_k , but those are not discussed here. Figure 1 shows exemplary the circuit for a Quantum Fourier Transform with three qubits and the circuit of the IQFT.

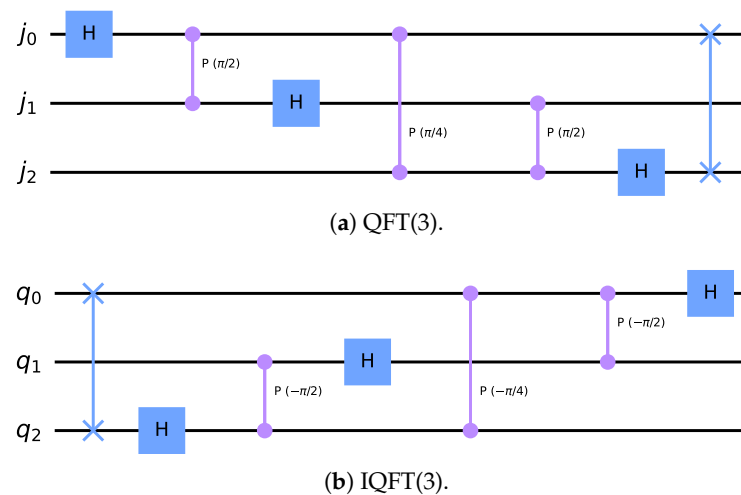


Figure 1. Circuits for the Quantum Fourier Transform (QFT) and its inverse (IQFT) for three qubits.

An efficient implementation of the Quantum Fourier Transform for n qubits is shown in Figure 2. Be aware, that the arrangement of the qubits is sometimes the other way round, for example in Qiskit [14]. It depends on whether the lowest qubit is seen as the most significant or as the least significant qubit [15].

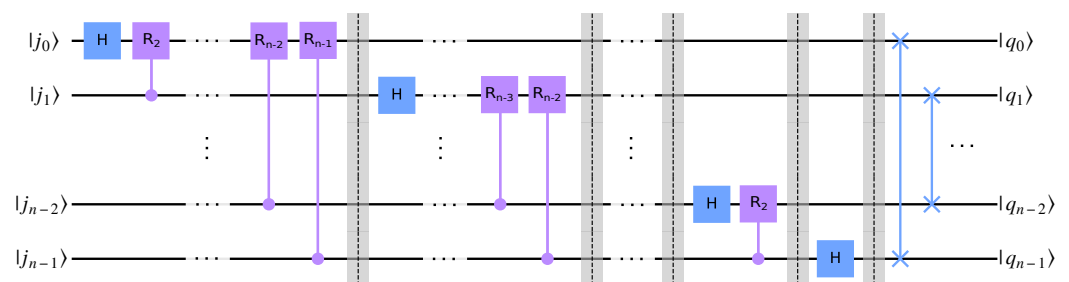


Figure 2. Circuit for the Quantum Fourier Transform. Note, that the order of the qubits at the end is reversed using SWAP-gates. The barriers (vertical dashed lines) are only for better visualization here.

The SWAP-gates at the end of the circuit are used to reverse the order, such that state $|j_0\rangle$ maps to $|q_0\rangle$, $|j_1\rangle$ to $|q_1\rangle$, and so on. Since each gate is unitary, the whole QFT is unitary [6].

When comparing the complexity of the Discrete Fourier Transform and the Quantum Fourier Transform, we can directly see the theoretical advantage of the quantum version [6]. It mainly comes from the fact, that we can encode the 2^n classical bits with using only n qubits. In total, we need for the first qubit one Hadamard gate and $n - 1$ conditional rotations, for the second qubit one Hadamard gate and $n - 2$ conditional rotations, and so on. So in total $n(n + 1)/2$ gates plus at most $n/2$ SWAP-gates. Consequently, we can implement the QFT with a computational complexity of $\Theta(n^2)$ operations.

In contrast, in the classical counterpart, the Discrete Fourier Transform, we have to apply the operations to all 2^n elements. With the FFT algorithm, this results in $\Theta(2^n \log(2^n)) = \Theta(n2^n)$ operations for processing 2^n elements. Therefore, there is an exponential increase in the number of classical operations needed to calculate the DFT compared to the QFT. Unfortunately, the problem here is that we can not directly access the amplitudes in a quantum computer. Hence, there is no way to get the Fourier transformed amplitudes of the original state.

5. Angle Estimation of Regular Patterns in Images

As already mentioned at the beginning of this paper, the Quantum Fourier Transform and also its classical analogue can be used in several algorithms and applications [11,16]. We use it to estimate the angle of rotated objects in images. There are a lot of use-cases, where we need to know the angle. Figure 3 illustrates two use-cases. The first involves a production line where we need to detect whether the product is correctly aligned. The second one consists of printed text on paper. In cases where the text is rotated, as in Figure 3b, our objective is to measure the rotation angle and then counter-rotate the image to ensure that the text is readable.

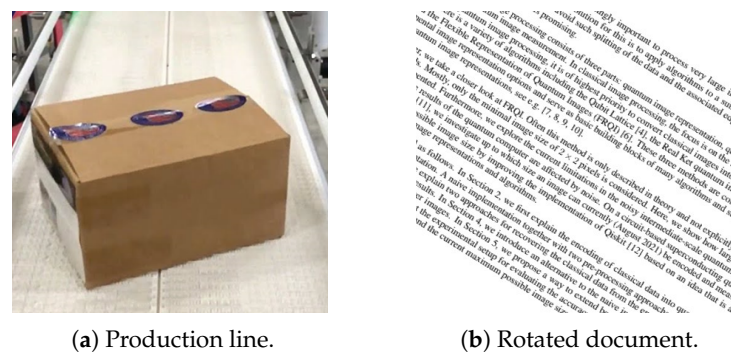


Figure 3. Use cases for image alignment: (a) A rotated package on a production line requiring angle estimation [17]. (b) A skewed text scan that needs to be realigned.

5.1. Classical Approach Using FFT

Figure 4 shows the graph we use for the angle estimation of the lines in the image. We implement it with the software ToolIP (Tool for Image Processing) [18], which is a software for interactively developing image processing algorithms developed by the image processing department of Fraunhofer ITWM. Algorithm 1 shows the pseudocode for the ToolIP graph.

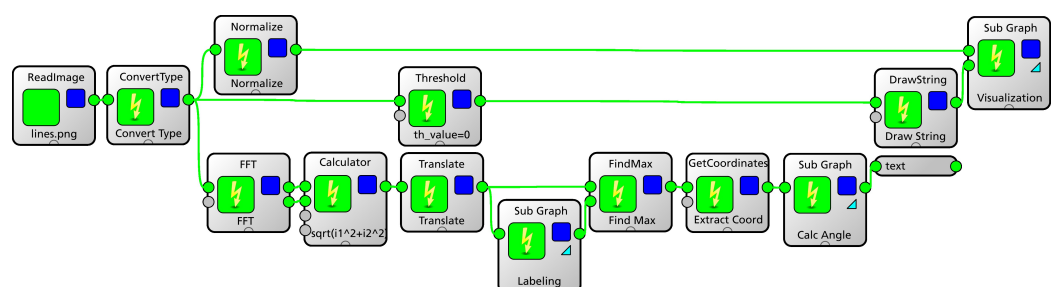


Figure 4. ToolIP [18] graph for estimating the angle of patterns in an image. Every box (plugin) contains pieces of code and is connected to other plugins.

The workflow starts with the ReadImage plugin, where the image with lines (see Figure 5a) is the output of this plugin. We insert this to the next plugin, where we convert the image type from integer values to float values, which we need for further calculations (see line 9 in Algorithm 1). Then, the information is forwarded into three further plugins, where the top two are only for visualization purposes and will not be discussed further here. The important parts are the plugins at the bottom of the ToolIP graph.

Algorithm 1 Angle estimation using FFT

```

1: procedure ANGLE-ESTIMATION-FFT( $X, M, N, t, m$ )
2:   # Input gray value image  $X$ ,
3:   # Height of the gray value image  $M$ ,
4:   # Width of the gray value image  $N$ .
5:   # Threshold value for labeling  $t$ 
6:   # Number of local maxima  $m$ 
7:   Output angle  $\alpha$ , # to be initialized later
8:   Temporary images  $E, F, G, H, I^*, I, J$ , # to be initialized later
9:    $E = \text{float}(X)$  # Convert pixel type from int to float
10:   $F = \text{FFT}(E)$  # Apply FFT to  $E$ 
11:  # Calculate absolute square-root of real and imaginary part of  $F$ 
12:   $G = \sqrt{(\text{Re } F)^2 + (\text{Im } F)^2}$ 
13:  # Translate image with given offsets  $M/2$  and  $N/2$ 
14:  for pixel  $p_{(0)} \in G$  do
15:     $H(p_{(0)}) = G([p_{(0)} + (M/2, N/2)]\%(M, N))$ 
16:  end for
17:  # Binarization
18:  for pixel  $p_{(1)} \in H$  do
19:    if  $\frac{H(p_{(1)}) - \min_{p \in H} H}{\max_{p \in H} H - \min_{p \in H} H} \cdot 255 > t$  then
20:       $I^*(p_{(1)}) = 1$ ;
21:    else
22:       $I^*(p_{(1)}) = 0$ ;
23:    end if
24:  end for
25:  # Labeling
26:  for pixel  $p_{(1)} \in H$  do
27:    if  $I^*(p_{(1)}) = 1$  then
28:      Label connected areas uniquely and consecutively;
29:      Start with  $I(p_{(1)}) = 1$ ;
30:    end if
31:  end for
32:  # Find local maxima per labeled region and select top  $m$ 
33:  Initialize empty list of local maxima (LOLM)
34:  for each unique label  $l$  in  $I$  do
35:    Find maximum pixel  $p_{(l)}$  in  $P_l$  with respect to intensity in  $H$ 
36:    Append  $p_{(l)}$  to LOLM
37:  end for
38:  Sort LOLM by intensity in descending order
39:  Select first  $m$  elements of LOLM and mark corresponding pixels in  $J$  with 255
40:  # Calculate angle if at least two maxima found
41:  if length of LOLM  $\geq 2$  then
42:    Take first two pixels  $(x_1, y_1), (x_2, y_2) \neq (M/2, N/2)$  from LOLM
43:     $\alpha = \text{round}(90 + \arctan((y_1 - y_2)/(x_1 - x_2)) \cdot 180/\pi)$ 
44:  else
45:    Break
46:  end if
47:  Return  $\alpha$ 
48: end procedure

```

First, we apply the FFT to the image. This FFT plugin corresponds to line 10 in Algorithm 1. We calculate the FFT for all pixel values.

The code for the following Calculator plugin is written in lines 11–12 in Algorithm 1. We calculate the absolute square root of the pixel values, which are imaginary values, with the equation

$$G(p_{(2)}) = \sqrt{\left(\operatorname{Re} F(p_{(2)})\right)^2 + \left(\operatorname{Im} F(p_{(2)})\right)^2}.$$

Afterwards, we shift pixels from the corners to the center and vice versa. We use half of the image's height and width as shift values for every pixel. These values are added to the current pixel position, and the modulo operator is applied to ensure that the resulting position remains within the image boundaries (see lines 13–16 in Algorithm 1). Typically, the pixels with the highest values are in the top-right and bottom-left corners. Through this translation, they are moved to the center of the image. This adjustment simplifies the process of angle estimation in the subsequent steps. The result of this translation is shown in Figure 5b.

With the following Labeling sub-graph, we give each pixel above a threshold t one label. This choice defines what we consider as foreground and what as background. Given that the majority of pixel values are close to 0 and the pixels of interest have values around 100, the specific threshold parameter chosen does not significantly impact the results. We selected the value of $t = 60$ based on an analysis of the histogram of pixel values. Lines 17–31 correspond to this sub-graph, where both the binarization and labeling are done. Figure 5c shows the result after applying that part of the code. We extracted a 36×36 patch out of the middle of the image, since the rest of the image is only black and the important parts are only the three non-zero pixels.

The result of the following FindMax plugin is an image that contains up to m global maxima, selected from the set of regional maxima. We use the labeled image I and the translated magnitude image H as inputs. For each connected region in I , we determine the pixel with the highest intensity in H and collect these into a list of local maxima (LOLM). From this list, we sort and select the top m brightest maxima. These m pixels are then marked in the output image J with the value 255. We set $m = 4$ in our case to ensure a sufficient number of reference points for robust angle estimation and later verification. This process is implemented in lines 32–40 of Algorithm 1.

After identifying the most relevant pixels, we extract their coordinates using the Get-Coordinates plugin. In our example, we obtain the coordinates $(0, 0)$, $(136, 121)$, $(128, 128)$, and $(120, 135)$.

For angle calculation in the Calc Angle sub-graph, we use the code from lines 41 to 47 in Algorithm 1, in particular the formula in line 44:

$$\alpha = \operatorname{round}\left(90 + \arctan\left(\frac{y_1 - y_2}{x_1 - x_2}\right) \cdot \frac{180}{\pi}\right). \quad (12)$$

The addition of 90° accounts for the image coordinate system, where the origin lies in the top-left corner and the y -axis increases downward. In our case, we use the non-zero pixels $(x_1, y_1) = (136, 121)$ and $(x_2, y_2) = (120, 135)$. The pixel $(128, 128)$ is excluded because it corresponds to the DC component (low-frequency average), which always appears as a bright central peak in the Fourier domain and does not carry directional information. Substituting the coordinates into Equation (12), we obtain the estimated angle $\alpha = 49^\circ$.

The remaining two plugins are only for visualization and plotting the estimated value on top of the input image like in Figure 5d. All operations of this classical approach can be executed quite rapidly, resulting in a total execution time of approximately 10 ms. For the computations, we used a computer with an Intel Xeon E5-2670 processor running at 2.60 GHz, a total RAM of 64 GB, and Red Hat Enterprise Linux 9.5 (Plow).

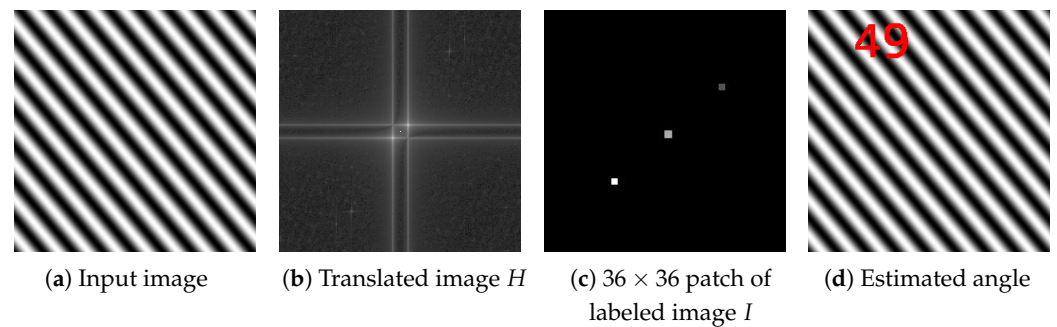


Figure 5. 256×256 pixel gray value image with lines rotated 50 degrees clockwise, intermediate results of the algorithm, and estimated angle written on the input image. The letters for the images refer to the letters used in Algorithm 1. We extracted a 36×36 patch in Figure 5c highlighting the important area of image I .

5.2. Hybrid Classical-Quantum Approach Using QFT

We can also replace the classical Fourier Transform with the Quantum Fourier Transform. We do this for the angle estimation and use the remaining part of the ToolIP graph in the same way as in the classical algorithm. Our goal is not to estimate the angle using real backends due to the current limitations discussed in [19]. Instead, we aim to demonstrate the feasibility of using the QFT algorithm for angle estimation with the `qasm_simulator`.

Figure 6 shows the abstract circuit for the Quantum Fourier Transform for the image with lines. As encoding method, we use the amplitude encoding method Quantum Probability Image Encoding QPIE [20].

The main idea of the QPIE method is to use the probability amplitudes of a quantum state to store the image information. It uses two separate quantum registers, one for the M rows and one for the N columns of a two-dimensional image $F = (F_{k,l})_{M \times N}$. The image is flattened to a vector of size MN with

$$\hat{F} = (F_{0,0}, F_{0,1}, \dots, F_{0,N-1}, F_{1,0}, \dots, F_{M-1,N-1})^T.$$

Then, we can represent the image by

$$|F\rangle = \sum_{j=0}^{2^n-1} c_j |j\rangle, \quad c_j = \frac{\hat{F}_j}{\sqrt{\sum_j \hat{F}_j^2}} \quad (13)$$

where $n = \lceil \log_2(MN) \rceil$. For simplicity, the basis states $|j\rangle$ are named by the decimal number j corresponding to the vector of zeros and ones read as binary numbers. The images themselves can have any arbitrary rectangular shape $M \times N$. We round the number of required qubits n to the next integer, since $\log_2(MN)$ is not necessarily an integer value. We do not need this rounding if we have a $2^k \times 2^l$ gray value image. Then we need exactly $k + l$ qubits to represent the image with QPIE.

We transform the input image into the form according to Equation (13) and insert the normalized input vector into the `initialize` command in Qiskit. It transforms the quantum states of the qubits directly from the $|0\rangle$ states to the quantum states, according to the input values.

For the 256×256 input image, we need 16 Hadamard gates, 56 controlled Phase gates, 8 SWAP-gates and 16 measurements to realize this circuit excluding the encoding part with QPIE. Since all of them are not basis gates, they would need to be transpiled for a real backend. Most of the quantum gates are needed to encode the image. The quantum gates needed for the QFT itself are only a small amount. The current error rates per two-qubit

gates (ECR) shown in Table 1 exclude the usage of the real hardware for this large number of gates.

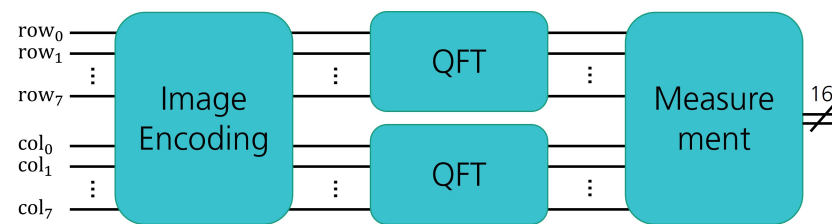


Figure 6. Circuit for calculating the QFT of the image with lines from Figure 5a using QPIE as quantum image encoding method.

Table 1. Calibration data of openly accessible IBM backends taken on 8 May 2025. Values are median values. Error rates of 2-qubit gates (ECR error), 1-qubit gates (SX error), and readout error are given in %.

Backend	2-Qubit (%)	1-Qubit (%)	Readout (%)	T1 (μs)	T2 (μs)
ibm_sherbrooke	0.7213	0.0238	1.465	278.65	179.72
ibm_brisbane	0.8573	0.0239	1.733	234.70	131.15

Also, missing connections between qubits would result in an even higher number of two-qubit gates using SWAP-gates to bridge the missing connections.

Figure 7 shows the results for using QFT. As we see, the translated image in Figure 7b is a bit noisier than the image in Figure 5b, where we used FFT. The reason for this is the sampling noise. We can only get a certain precision with the used 100,000 shots. To deal with this noise, we use the morphological operator erosion [11] with a 3×3 window-size for classical post-processing. Thereby, we search for the minimum value of the image within a region around every pixel. This way, we reduce the number of pixels with high values (white pixels in Figure 7b). It will help us in estimating the angle because only the central points of the regions still have high values. Mathematically, the erosion of an image F with the structuring element B at the position (x, y) is given by

$$[F \ominus B](x, y) = \min_{(s, t) \in B} F(x + s, y + t). \quad (14)$$

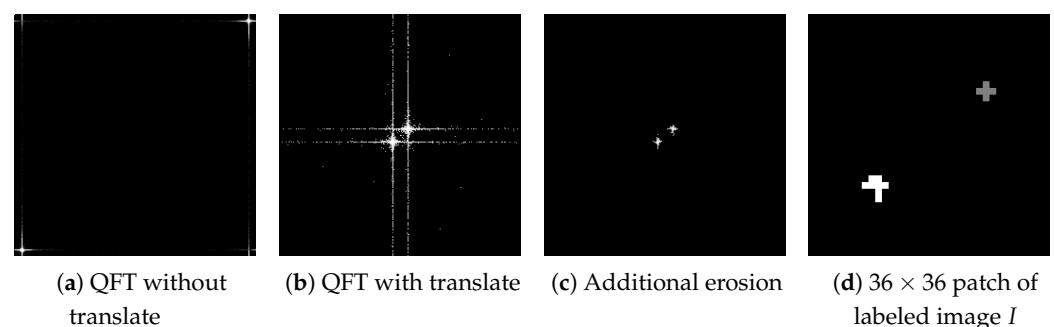


Figure 7. 256×256 result from QFT circuit: result without translation, result after translation (with exponent 0.1 applied for better visualization), additionally applied erosion, and 36×36 patch showing the important area of image I . Since the input image and estimated angle are the same as in Figure 5a and Figure 5d, respectively, they are not shown again.

The result of the erosion is visualized in Figure 7c. We use this image to continue with the same algorithm as in the classical case. The changed ToolIP graph is shown in Figure 8. Lines 13–49 in Algorithm 1 are exactly the same as in the classical case.

In the end, we get the same result as in the classical case, with 49 degree rotation detected.

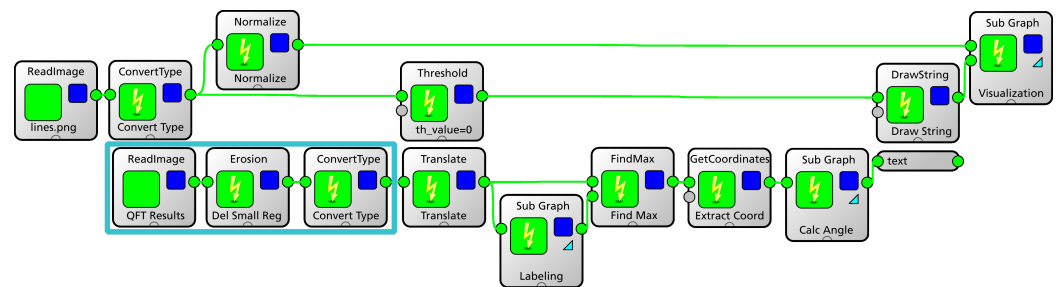


Figure 8. ToolIP graph for estimating the angle of patterns in an image starting with the image after applying QFT. Every box (plugin) contains pieces of code and is connected to other plugins. The framed part is the part, which is differing from the classical algorithm in Figure 4.

For the calculation, we used a computer with an Intel Xeon E5-2670 processor running at 2.60 GHz, a total RAM of 64 GB, and Red Hat Enterprise Linux 9.5 (Plow). In total, the algorithm needs around 5.4 s until it is executed, which is a factor of around 540 more than the classical counterpart. We could decrease the time by decreasing the number of shots. However, this would increase the sampling noise and affect the results.

With only one angle and on the toy sample with lines, it is not proven that our algorithm works to estimate the angle of lines in an image. Because of that, we continued in the same manner and estimated the angles of text snippets with varying rotation angles in the following section.

6. Application of FFT and QFT for Angle Estimation on Rotated Text Images

Determining the rotation of an object or text is a critical preprocessing step in many image processing applications, as it directly affects the accuracy of subsequent analysis tasks such as segmentation and recognition. For instance, in the context of image of text digitization, identifying the correct orientation of the text ensures accurate optical character recognition (OCR) and layout analysis. Similarly, in industrial settings such as a production line, the detection of the rotation of objects allows precise robotic manipulation, quality control, and defect detection. Failure can lead to significant errors and reduced system reliability.

We demonstrated how either the FFT or the QFT can be used to estimate the rotation angle of lines in an image. For comparison of QFT and FFT, we incrementally vary the rotation of a lorem ipsum text in steps of ten degrees, ranging from 0° to 170°, as shown in Figure 9. The rotation angles from 180° to 360° are essentially the same as those from 0° to 170°, so we omit them here.

Figure 10 shows the absolute differences

$$\text{diff}_{abs} = |\alpha_{out,j} - \alpha_{in,j}|$$

between the expected angles $\alpha_{in,j}$ and the angles $\alpha_{out,j}$ from either FFT or QFT, for $j \in \{0, 1, \dots, 17\}$. We used the `qasm_simulator` with 100,000 shots for the QFT.

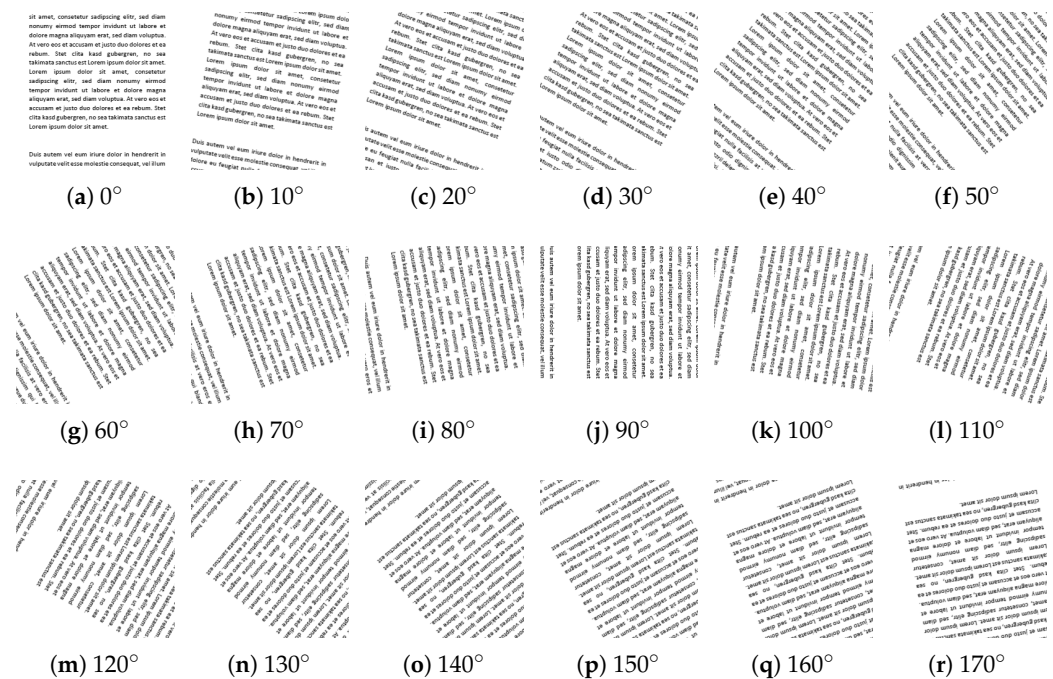


Figure 9. 256×256 images containing rotated text ranging from 0° to 170° .

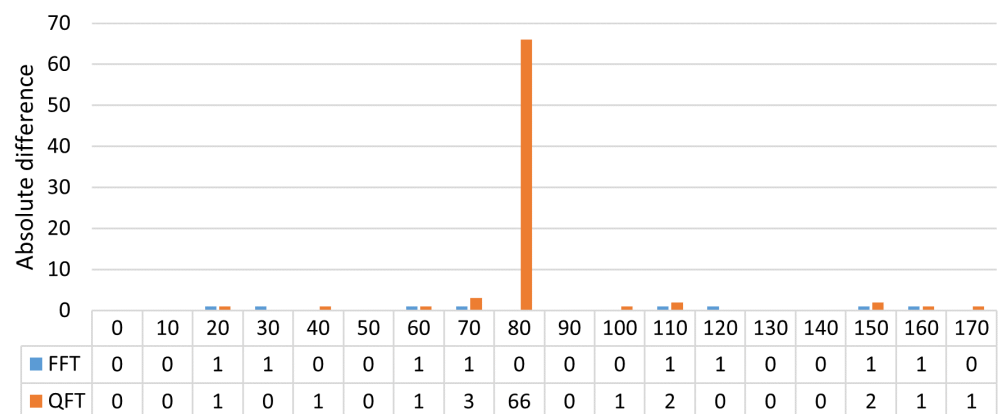


Figure 10. Absolute differences between the expected rotations shown on the x -axis and the angles from either FFT or QFT.

With both transformations, we can estimate the angles as accurately as possible, with a maximum absolute difference of 3° , except for the angle 80° . This empirically demonstrates the usability of our algorithm for estimating angles in this use case.

6.1. Post-Processing in the QFT Case

As shown in Figures 5c and 7d, the results obtained using the Quantum Fourier Transform (QFT) exhibit higher noise levels compared to those from the classical Fast Fourier Transform (FFT). To mitigate this, we applied an additional morphological erosion step in the QFT-based pipeline.

Nevertheless, in certain cases, such as the 80° orientation, we observe a substantial deviation in the estimated angle—up to 66° . Figure 11 shows the difficulties to extract the corresponding points of that image for calculating the rotation.

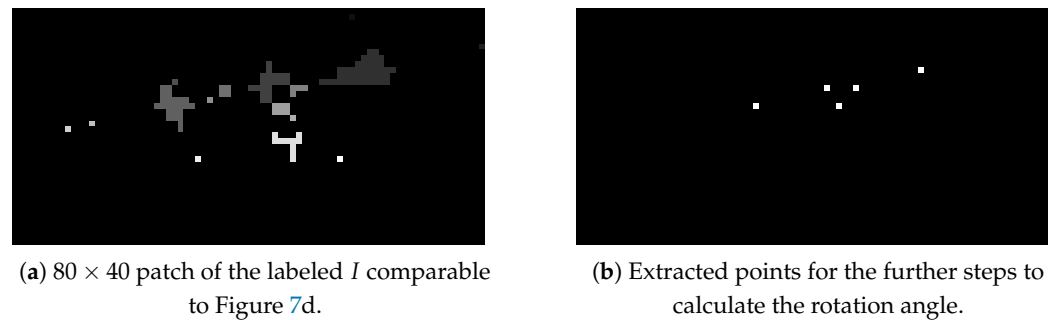


Figure 11. Labeled image and extracted points for 80° rotation of the text in the image. The images are displayed in a normalized version.

This discrepancy arises primarily from noise-induced artifacts in the labeled image patch. In such cases, noise from the quantum circuit can result in bigger regions of interest, which have to be reduced to two points for calculating the angle. Then it is a classical problem to decrease the amount of points (see Figure 11a) to two of the most relevant points. In Figure 9, we plot five most relevant points using a threshold-based labeling and finding maximum method in ToolIP. Then two of them are used to calculate the rotation angle using Equation (12).

Although additional post-processing or denoising techniques could potentially improve robustness against such artifacts of choosing the wrong points, these strategies were not investigated further within the scope of this work and are left for future exploration.

6.2. Required Quantum Gates and Circuit Depth

To implement the encoding procedure using the Quantum Phase Interpolation Encoding (QPIE) method combined with the Quantum Fourier Transform (QFT), the quantum circuit for the 256×256 gray value images requires a single initialization operation, 16 Hadamard gates (one applied to each qubit), and 56 controlled phase gates. After these operations, 16 measurements—one per qubit—are performed, yielding results that can be postprocessed to estimate the rotation angle of the text within an image. While the high-level gate count appears modest, the actual resource requirements significantly increase when the circuit is compiled to the native basis gates of IBM's quantum hardware, which include the identity (id), square-root of X (SX), rotation around Z (RZ), echoed cross-resonance (ECR), and Pauli-X (X) gates. The decomposition of each high-level gate into these native operations introduces considerable overhead, making the physical implementation more resource-intensive. Table 2 summarizes the estimated basis gate count. In total, we need 454,149 single-qubit gates and 65,631 two-qubit gates.

Table 2. Estimated basis gate decomposition of the circuit for the 256×256 image (see Figure 9) on IBM quantum hardware. Note that running it on IBM's real hardware may result in slightly varying numbers due to the optimization process in Qiskit's transpile function, mainly depending on the chosen `optimization_level`—we used the default `optimization_level = 2`.

High-Level Operation	RZ	SX	X	ECR
Initialization	258,775	129,274	65,552	65,519
QFT	324	128	96	112
Measurement	—	—	—	—
Total	259,099	129,402	65,648	65,631

As observed, most quantum gates originate from the initialization of the quantum state, which introduces a substantial number of operations, far exceeding the capabilities of current quantum hardware. Even when using the `qasm_simulator` with basis gates as the gate set, the resulting circuits are extremely large and require long execution times.

7. Conclusions and Discussion

In this work, we presented a method for estimating the angles of objects, specifically lines and text, in the images using both classical Fast Fourier Transform (FFT) and Quantum Fourier Transform (QFT). We developed an algorithm capable of utilizing the output of either transform as basis for angle detection. Importantly, we demonstrate that QFT can be applied to a concrete image processing task, illustrating the practical utility of quantum computing beyond theoretical examples. This represents a meaningful step toward solving real-world problems with quantum computers, providing an early proof of concept for quantum-enhanced image analysis in the NISQ era.

We have the theoretical benefit of using the Quantum Fourier Transform instead of the Discrete Fourier Transform with logarithmic complexity compared to the classical analogue. However, there is currently no efficient way to implement it for a reasonable image size. The 112 two-qubit gates for the QFT and over 65,000 for the image encoding for a 256×256 gray value image are challenging in the current NISQ era (see also [19,21]), where each of the entangling gates has around 1% error. That is the reason why we only used the `qasm_simulator` for the QFT. In the future, as hardware continues to improve and errors decrease, we can expect that more will become possible in the future.

Quantum random access memory holds immense potential to enhance the utility of the Quantum Fourier Transform in quantum computing. By facilitating rapid data retrieval and manipulation, QRAM can significantly reduce the computational overhead we experienced in using QPIE for image encoding. This is the biggest part of the algorithm.

Performance-wise, the QFT remains slower than the FFT even in simulation, due to the overhead of quantum gate translation. On actual quantum hardware, additional delays arise from circuit compilation, qubit connectivity limitations, and queuing time. As a result, using QFT for angle estimation on real backends currently yields impractical results.

Approaches such as implementing the Quantum Fourier Transform (QFT) using dynamic circuits [22] or employing approximate versions of the QFT [23] were not considered in this work. Such methods could help bridge the gap between theoretical quantum advantages and practical feasibility, especially when tailored to the constraints and capabilities of near-term quantum hardware.

Author Contributions: Conceptualization, A.G. and A.M.; methodology, A.G. and A.M.; software, A.G. and A.M.; validation, A.G. and A.M.; formal analysis, A.G. and A.M.; investigation, A.G. and A.M.; writing—original draft preparation, A.G.; writing—review and editing, A.G. and A.M.; visualization, A.G. and A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the Federal Ministry for Economic Affairs and Climate Action through the EniQmA project (funding number 01MQ22007A).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data and simulation materials used in this study are available from the corresponding author upon reasonable request, subject to any applicable restrictions.

Acknowledgments: This work was supported by the Quantum Initiative Rhineland-Palatinate (QUIP) and by the European Union, Interreg Upper Rhine Valley program within the UpQuantVal project. We also acknowledge the Fraunhofer-wide Capacity Building Campaign Quantum Now, which enabled us to perform experiments on both U.S. and EU-based IBM Quantum systems.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

QFT	Quantum Fourier Transform
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
IBM	International Business Machines Corporation
NISQ	Noisy Intermediate-Scale Quantum
ToolIP	Tool for Image Processing
QPIE	Quantum Probability Image Encoding
IQFT	Inverse Quantum Fourier Transform
QRAM	Quantum Random Access Memory
RAM	Random Access Memory
RZ	Z-axis rotation gate
SX	Square-root X gate
ECR	Echoed Cross-Resonance

References

1. Coppersmith, D. An approximate Fourier transform useful in quantum factoring. *arXiv* **2002**, arXiv:quant-ph/0201067. [\[CrossRef\]](#)
2. Hales, L.; Hallgren, S. An improved quantum Fourier transform algorithm and applications. In Proceedings of the 41st Annual Symposium on Foundations of Computer Science, Redondo Beach, CA, USA, 12–14 November 2000; pp. 515–525. [\[CrossRef\]](#)
3. Heideman, M.; Johnson, D.; Burrus, C. Gauss and the history of the fast fourier transform. *IEEE ASSP Mag.* **1984**, *1*, 14–21. [\[CrossRef\]](#)
4. Shor, P. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134. [\[CrossRef\]](#)
5. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [\[CrossRef\]](#)
6. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information*, 10th ed.; Cambridge University Press: Cambridge UK; New York, NY, USA, 2010.
7. Kitaev, A.Y. Quantum measurements and the Abelian Stabilizer Problem. *arXiv* **1995**, arXiv:quant-ph/9511026. [\[CrossRef\]](#)
8. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.* **2009**, *103*, 150502. [\[CrossRef\]](#) [\[PubMed\]](#)
9. Yan, F.; Ilyasu, A.M.; Venegas-Andraca, S.E. A survey of quantum image representations. *Quantum Inf. Process.* **2016**, *15*, 1–35. [\[CrossRef\]](#)
10. Li, H.; Quan, D. Quantum Image Processing: A Review and Future Research Directions. *Entropy* **2021**, *23*, 448. [\[PubMed\]](#)
11. Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*, 4th ed.; Pearson: New York, NY, USA, 2018.
12. Schindler, K.; Suter, D. Fourier-based rotation-invariant object recognition using contours and appearance models. In *Proceedings of the Asian Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1–12.
13. Schuld, M.; Sinayskiy, I.; Petruccione, F. An introduction to quantum machine learning. *Contemp. Phys.* **2015**, *56*, 172–185. [\[CrossRef\]](#)
14. Qiskit Contributors. Qiskit: An Open-source Framework for Quantum Computing. *Zenodo* **2019**. [\[CrossRef\]](#)
15. Qiskit Contributors. Learn Quantum Computation Using Qiskit. Available online: <http://community.qiskit.org/textbook> (accessed on 10 June 2023).
16. Osgood, B.G. *Lectures on the Fourier Transform and Its Applications*; IAS/Park City Mathematics Series/AMS; Electrical Engineering Department, Stanford University: Stanford, CA, USA, 2019.

17. Multi-Conveyor LLC. Product Turning Conveyors. 2023. Available online: <https://www.multi-conveyor.com/packaging-solutions/product-turners> (accessed on 8 May 2025).
18. Fraunhofer ITWM, Department of Image Processing. ToolIP—Tool for Image Processing. 2024. Available online: www.itwm.fraunhofer.de/toolip (accessed on 8 May 2025).
19. Geng, A. Application of Hybrid Quantum Machine Learning for Image Processing in the NISQ Era. Ph.D. Thesis, Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau, Kaiserslautern, Germany, 2024. [[CrossRef](#)] [[PubMed](#)]
20. Yao, X.W.; Wang, H.; Liao, Z.; Chen, M.C.; Pan, J.; Li, J.; Zhang, K.; Lin, X.; Wang, Z.; Luo, Z.; et al. Quantum Image Processing and Its Application to Edge Detection: Theory and Experiment. *Phys. Rev. X* **2017**, *7*, 031041. [[CrossRef](#)]
21. Geng, A.; Moghiseh, A.; Redenbach, C.; Schladitz, K. Improved FRQI on superconducting processors and its restrictions in the NISQ era. *Quantum Inf. Process.* **2023**, *22*, 104. [[CrossRef](#)]
22. Bäumer, E.; Tripathi, V.; Seif, A.; Lidar, D.; Wang, D.S. Quantum Fourier Transform using Dynamic Circuits. *Phys. Rev. Lett.* **2024**, *133*, 150602. [[CrossRef](#)]
23. Gao, X.; Jin, Y.; Guo, M.; Chen, H.; Zhang, E.Z. Linear Depth QFT over IBM Heavy-hex Architecture. *arXiv* **2024**, arXiv:2402.09705. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.