electronics

MDPI

Article

A Real-Time Image Stitching and Fusion Algorithm Circuit Design Based on FPGA

Yu Jia, Ruibo Wang and Xianyang Jiang *

School of Physics and Technology, Wuhan University, Wuhan 430072, China; 2016301510037@whu.edu.cn (Y.J.); 2020202020095@whu.edu.cn (R.W.)
* Correspondence: jiang@whu.edu.cn

Abstract: In the widely used field of panoramic image stitching, the key technologies mainly cover two parts, i.e., image registration and image fusion. In order to achieve low-cost and real-time processing, researchers often design dedicated circuits for various image stitching algorithms. Many studies focus on image registration algorithms and ignore image fusion algorithms, let alone dedicated circuit design. In addition, to reduce the ghosting effect and deformation caused by seams in stitching, finding the best seam line based on the overlapping area of the stitching image is crucial, which directly affects the quality of image stitching and fusion. In order to solve the above problems and achieve the efficient fusion of registered images, an image stitching and fusion algorithm circuit based on a dynamic programming algorithm to search for seam lines was proposed. Comprehensive experimental results and a theoretical analysis based on Cyclone IV FPGA devices show that, with a clock frequency of 100 MHz, the proposed circuit takes about 7.04 ms to carry out the fusion processing of two 486 × 643 images, and the corresponding frame rate is approximately 142 FPS, achieving a perfect real-time stitching effect and meeting the demand for real-time image processing. After a theoretical derivation and comparison with other similar works, its processing speed is better than four state-of-the-art implementations.

Keywords: field-programmable gate arrays (FPGA); image stitch; digital integrated circuit; weighted fusion

Citation: Jia, Y.; Wang, R.; Jiang, X. A Real-Time Image Stitching and Fusion Algorithm Circuit Design Based on FPGA. *Electronics* **2024**, *13*, 271. https://doi.org/10.3390/electronics13020271

Academic Editor: Akash Kumar

Received: 28 November 2023
Revised: 4 January 2024
Accepted: 5 January 2024
Published: 7 January 2024

1. Introduction

In the rapidly evolving fields of image sensor technology and image processing, there is an increasing demand for large-field-of-view (FOV) and high-resolution images. This demand is driven by the need for more comprehensive and clearer scene descriptions [1–3]. Panoramic imaging technology, which has emerged in this context, primarily employs two methods for capturing real-time, high-resolution, large-FOV images. The first method uses wide-angle lenses, offering an extensive FOV but often at the cost of resolution and with the introduction of significant distortion [4,5]. The second, more popular, method is real-time image stitching technology. This approach splices multiple images with overlapping areas into a complete high-resolution image. Due to its rich information content, flexible shooting conditions, and cost-effectiveness, image stitching technology has become prevalent in applications like video surveillance, medical imaging, and remote sensing [6–9].

In the field of panoramic image stitching, key technologies primarily encompass two areas: image registration and image fusion [10]. Image registration aligns different images in the same scene, while image fusion combines these images, preserving critical information. Challenges like camera parameter differences and variations in resolution often lead to issues like ghosting, where objects appear duplicated, and deformation, resulting in unnatural image appearances. Seam lines, noticeable differences between stitched images, are addressed through image fusion.

In the pursuit of efficiency, cost-effectiveness, and real-time performance, the simplification and innovation of algorithms have consistently been a focus of effort and the design

of dedicated circuits for image stitching is also increasingly recognized as a popular choice. Yet, research has predominantly centered on circuits for image registration, somewhat neglecting image fusion circuits [11,12]. The effectiveness of image stitching relies on the seamless integration of all components, making the performance of image fusion circuits crucial. The research thus emphasizes selecting and optimizing appropriate image fusion algorithms for hardware implementation, aiming to balance resource use with processing speed without losing effectiveness.

1.1. Literature Review

1.1.1. Advancements in Feature Detection and Image Stitching

The development of feature detection and image stitching algorithms has been pivotal in enhancing the efficiency and accuracy of panoramic image generation. A notable contribution in this field is the Scale-Invariant Feature Transform (SIFT) algorithm, proposed by David Lowe et al. [13,14], which is renowned for its robustness against scale, rotation, and brightness changes, and its stability against variations in viewing angles, noise, and affine transformations.

Expanding upon this, Yang et al. [15] improved the image stitching process using an enhanced SURF method, focusing on efficiency and accuracy. Hussain et al. [16] applied the SIFT algorithm to autonomous driving systems, concentrating on collaborative perception. Further advancements were made in multi-channel image stitching, as demonstrated in another study [17], which enhanced the processing of multi-dimensional data. Similarly, the authors [18] explored the use of SIFT-based image stitching in the field of cooperative perception for autonomous vehicles. The synergy of SIFT with the Random Sampling Consensus (RANSAC) algorithm [19] has been critical in eliminating false matches and achieving accurate homography alignment, paving the way for multi-band fusion stitching algorithms. However, the computational intensity of SIFT and RANSAC poses challenges for real-time performance, leading to the exploration of deep learning methods by Xu Xiangyang et al. [20] and Bayrak et al. [21] for feature point registration and image fusion, albeit at a higher computational cost.

1.1.2. Image Fusion Methods and Seam-Line-Based Fusion

The realm of image fusion includes various methods, like wavelet transform-based fusion [19], multi-resolution spline fusion [22,23], and notably, seam-line-based fusion [24]. Seam-line-based fusion is distinguished for its ability to retain original image content, ensure image consistency, and reduce computational load, making it suitable for FPGA hardware implementation.

The seam-line-based fusion focuses on identifying the least noticeable seam line in overlapping areas between different images, effectively reducing ghosting and deformation. Currently, there are many seam search algorithms in image stitching, such as the Dijkstra algorithm [25], simulated annealing algorithm [26], and Graph Cuts algorithm [27,28]. However, these algorithms face challenges in real-time applications due to their high computational demands and memory requirements for hardware implementation. Addressing this, Kwang-Wook Lee et al. [29] proposed a seam search algorithm based on the Greedy algorithm, offering computational efficiency but with the risk of local optimality. A more globally focused approach is seen in the dynamic programming algorithm [30,31], which, while efficient in obtaining optimal seam lines, has limitations in terms of its computational complexity. Hua Gu et al. [32] proposed improvements to this algorithm, enhancing its ability to find a global optimal seam line. Additionally, Yin et al. [33] introduced an enhanced two-stage dynamic programming algorithm for moving targets and a real-time local adjustment method for seam lines.

1.2. Highlights

In this paper, we present a novel design of a dedicated hardware circuit for image stitching, optimized for real-time efficiency in relatively static scenes. Focused on seam-

line-based fusion, this design addresses the less-explored realm of image fusion algorithms. Our contribution lies in the implementation of a dynamic programming algorithm for seam line search, significantly enhancing image fusion speed. Experiments on Cyclone IV FPGA devices demonstrate the circuit's ability to process two 486 × 643 images in about 7.04 ms (142 FPS), and, in fixed content scenes, it achieves an even faster rate of 266 FPS. This performance surpasses four current state-of-the-art implementations, marking a significant advancement in real-time image stitching technology.

The paper is structured as follows: in Section 2, we introduce and compare dynamic programming with the Greedy algorithm, focusing on their time and space complexities. In Section 3, we explore hardware algorithm design, emphasizing dynamic programming's application in image stitching and fusion techniques. In Section 4, we focus on the hardware circuit design, detailing the roles of various components in our image fusion system. In Section 5, we present experimental results, showcasing the system's efficiency and effectiveness through various metrics and comparisons. The conclusion is noted in Section 6. Finally, Section 7 is dedicated to future work.

2. Dynamic Programming Algorithm Analysis

Dynamic programming is an algorithmic idea commonly used for optimization problems and is widely used in image processing, computer vision, and other fields. In image processing, dynamic programming is used to find the optimal seam line that minimally disrupts the visual content of the images being stitched together. The strength of dynamic programming lies in its ability to break down a complex problem into simpler subproblems, solve each subproblem just once, and store their solutions in a table (usually a two-dimensional array), avoiding the redundant work of solving the same subproblem multiple times. In this way, the table can be directly looked up to obtain the solutions to the subproblems when needed, thus significantly improving the efficiency of the algorithm. Image fusion, a technology that combines images acquired from different sensors or viewing angles to obtain more comprehensive information, often requires consideration of how to obtain high-quality fusion results while maintaining the consistency of image information. Therefore, the application of dynamic programming can be understood as the process of optimizing image pixel values to maximize the visual consistency and quality of the stitched image.

2.1. Dynamic Programming Search Algorithm for Image Seam Lines

The process of dynamic programming in image fusion can be summarized as follows:

Cost function (energy function) construction: First, a cost function needs to be defined, which measures the difference in overlapping areas between images acquired from different viewing angles. The design of the energy value function takes into account the pixel value, color, and other characteristics of the image, as well as the smoothness and consistency requirements of the fusion.

State space definition: Treat the pixels of the image as part of the state space to construct a state space, where each state represents a possible value of a pixel. The dimensions of this state space correspond to the number of pixels in the overlapping region image.

Subproblem definition: Split the original image into a series of subproblems, with each subproblem corresponding to a pixel or a group of pixels. Each subproblem needs to determine the optimal pixel value based on the cost function.

Dynamic programming solution: Start from the subproblems and calculate the optimal seam line of each subproblem from the bottom up. In this process, the results of the solved subproblems are used to gradually construct the optimal seam line to the entire problem.

Fusion result generation: Once the dynamic programming solution is completed, the optimal seam line can be obtained through backtracking so that when the left and right images are fused here, the fused image can be generated with a smoother transition effect.

2.2. Comparison between Dynamic Programming Algorithm and Greedy Algorithm

The time complexity and space complexity of dynamic programming are as follows: Suppose the dimensions of the two images are $m \times n$, respectively, where m represents the width of the image and n represents the height of the image. Use a dynamic programming algorithm to calculate the seam line because you need to traverse each pixel of the image and consider information such as the value and weight of adjacent pixels to calculate the optimal seam line. The time complexity is usually $O\ (m * n)$, that is, $O(n^2)$. Because it is necessary to create a two-dimensional table with the same size as the input image to store the optimal seam line position of each pixel, the space complexity is usually $O(m * n)$, that is, $O(n^2)$.

The basic idea of the Greedy algorithm is to select the best choice under the current situation at each step, hoping to achieve the global optimal seam line through a series of local optimal choices. The Greedy algorithm often makes the optimal choice at every step without considering how subsequent choices may affect the outcome. Therefore, the time complexity of the Greedy algorithm is usually linear, that is, $O(n)$, where n represents the height of the image. Furthermore, Greedy algorithm can operate directly on the original image without using additional data structures to store intermediate results or solution spaces. Therefore, the space complexity of the Greedy algorithm is usually $O(1)$, which is a constant.

The Greedy algorithm may not guarantee the global optimal seam line when selecting the local optimal seam line at each step. Therefore, although the Greedy algorithm may be smaller in its time and space complexity, the results may be limited by local optimal selection, resulting in the poorer fusion of images. In image fusion problems, the dynamic programming algorithm requires more computing and storage resources due to its higher computational complexity. However, they can provide better fusion results by finding the optimal global solution [34].

3. Hardware Algorithm Design

The dynamic programming algorithm is widely used in seam line search because of its good real-time performance and fusion effect. Based on the original algorithm, this study made improvements based on the hardware circuit design requirements. By optimizing the calculation method and search strategy of the energy function, we effectively reduced the consumption of computing resources and further improved the real-time performance of the calculation.

For the convenience of explanation, it is assumed that the two image objects to be fused have achieved image registration and are aligned in the ordinate direction, as shown in Figure 1. In the diagram, i is the ordinate of the image pixel, j is the abscissa of the image pixel, j_{lp1} represents the left edge abscissa of image 1 in the result image coordinate system, j_{lp2} represents the left edge abscissa of image 2 in the result image coordinate system coordinate. The searched seam lines will be located in the overlapping area of the two images.

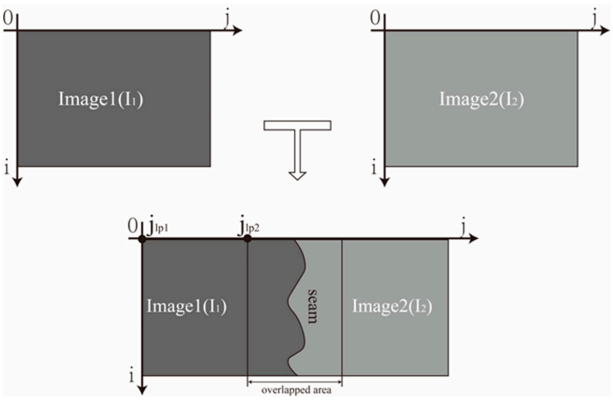

Figure 1. Image stitching from two horizontally aligned images.

3.1. Energy Value Calculation

In this study, an energy function for optimal stitching line search is constructed to meet the needs of hardware circuits. Energy function [29] construction is usually based on the difference in pixel values of two images within the overlapping area, that is, the combination of color difference, gradient value and a special distance term. However, the space complexity of dynamic programming algorithm for image stitching is $O(n^2)$, resulting in a heavy computational burden and excessive storage resource consumption. In fact, in the overlapping area after strict registration, the main difference between the images is the brightness, and in the fused image, the difference in image brightness may lead to a large fusion error, and the difference in light and dark of the image is the main perception of the human eye. Therefore, it is proposed to use the absolute value of the difference between the brightness channels Y in the overlapping area as the energy value, as shown in Equation (1):

$$E(i,j) = |Y_1(i,j) - Y_2(i,j)| \tag{1}$$

Y_1 and Y_2 represent the brightness values in the overlapping area. By using this energy function, the best seam line can be accurately determined in image stitching, achieving high-quality image fusion effects, and saving two-thirds of storage resources compared to the construction of traditional energy value functions.

In view of the distribution of RGB color components in the image and other important indicators, a weighted average process was performed using psychological formulas to determine the relationship between the three-color components of red (R), green (G), and blue (B) in the image and the brightness channel Y, as shown in Equation (2):

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{2}$$

In order to avoid the overhead of low-speed floating point operations and division operations in practical applications, this equation is modified to make the operation more efficient, as shown in Equation (3):

$$Y = (306 \times R + 601 \times G + 117 \times B)/1024 \tag{3}$$

3.2. Seam Line Search Strategy

In dynamic programming seam search, starting from the starting point of the seam, the image is scanned line by line in order from left to right. During the scanning process, the cumulative energy value from each pixel point to the starting point of the seam line is calculated. This value is the sum of all pixel values between the pixel value of the current point and the starting point. Then, in the opposite direction (backtracking from the end point to the starting point), a selection is made based on the accumulated energy value to find the adjusted optimal seam line.

The pseudocode for this process is shown in Algorithm 1.

Algorithm 1. The pseudocode of seam line search strategy

```
Function calculateSeamLine (image)
        Input: E—a 2D array for energy values of each pixel in the image
        Output: seamLine—an array representing the path of the minimum energy seam
        Define M as a 2D array for accumulated energy values
        For j from 0 to (width of image − 1)
M(0, j) = E(0, j)
        For i from 1 to (height of image − 1)
For j from 0 to (width of image − 1)
M(i, j) = E(i, j) + min(M(i − 1, j − 1), M(i − 1, j), M(i − 1, j + 1))
        minEnergyColumn = column j with minimum value in M(last row)
        seamLine = array to store seam path
Set currentColumn = minEnergyColumn
For i from (height of image − 1) down to 0
If I > 0
Add currentColumn to seamline
currentColumn = find column j with min adjacent value in M(i − 1)
        Return seamline
End Function
```

The specific implementation steps are as follows:

1. As shown in Figure 2, $M(i,j)$ is defined as the accumulated energy value of the seam node in the i-th row and j-th column, and $E(i,j)$ is the energy value of the pixel in the i-th row and j-th column.

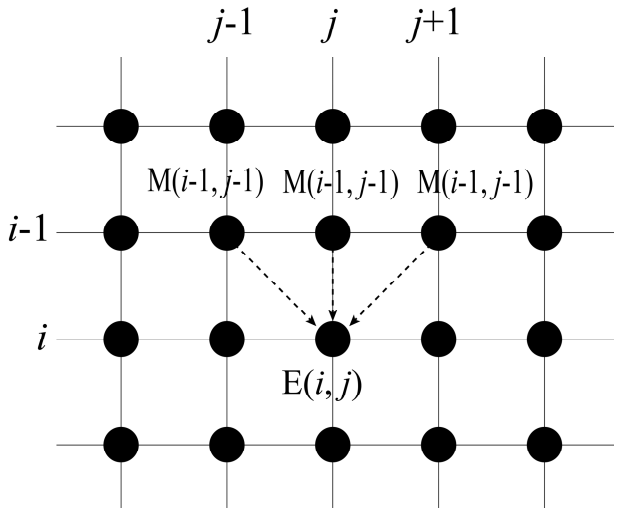

Figure 2. Dynamic programming algorithm diagram.

2. For the first line of the image, each pixel is used as the starting point of the seam line, and the corresponding accumulated energy value is the energy value of the pixel.
3. Starting from the second row (i-th row), for each pixel point, select a point with the smallest cumulative energy value from the three adjacent pixel points in the $(i-1)$-th row as the best stitching path node. As shown in Equation (4), add the minimum value to the energy value of the current point to obtain the accumulated energy value of the current point, and then iteratively update the accumulated energy value:

$$\mathrm{M(i,j)} = \mathrm{E(i,j)} + \min\{\mathrm{M(i-1,j-1)}, \mathrm{M(i-1,j)}, \mathrm{M(i-1,j+1)}\} \quad (4)$$

4. Repeat the above steps until reaching the last line of the image and traverse all pixels.
5. Finally, compare the cumulative energy values of all different seam lines, select the seam line with the minimum cumulative energy value, find the end node of the best seam line, and obtain the entire best seam line path through backtracking.

3.3. Image Fusion

After determining the coordinates of the seam line, the pixels in the overlapping area of the two images need to be processed to obtain a complete image. Commonly used pixel fusion methods include the direct average method and the weighted average method. The direct average method [35] adds the pixel values in the overlapping area and divides them by two, but the transition effect it presents on the image lacks natural smoothness, and the human eye can easily distinguish the difference in brightness. The weighted average method [36] can make full use of the human eye's relatively slow response to gradient information by adjusting the weighting coefficient so that the overlapping areas of the image present a smoother transition. Compared with other complex image fusion algorithms, the weighted fusion operation consumes fewer computing resources. In the hardware implementation, fixed-point number operations are used instead of floating-point number operations to determine the weights, further improving the calculation speed. This article selects three pixels around the seam line for weighted fusion processing.

The weighted fusion algorithm is shown in Equation (5):

$$I(i,j) = \begin{cases} I_1(i,j), & j < j_{seam} - 3 \\ I_1(i,j) \times \omega_1(j) + I_2(i,j) \times \omega_2(j), & |j - j_{seam}| \leq 3 \\ I_2(i,j), & j > j_{seam} + 3 \end{cases} \quad (5)$$

In the formula, $I_1(i,j)$, $I_2(i,j)$ and $I(i,j)$ represent the pixel values of image 1, image 2 and the stitched image, respectively; j_{seam} represents the abscissa value of the optimal seam line; $\omega_1(j)$ and $\omega_2(j)$, respectively, represent the weights of image 1 and image 2 in weighted fusion, and they are determined using Equations (6) and (7):

$$\omega_1(j) = \frac{j_{seam} - j + 4}{8} \quad (6)$$

$$\omega_2(j) = \frac{j - j_{seam} + 4}{8} \quad (7)$$

4. Hardware Circuit Design

4.1. Image Fusion System Design Framework

The system circuit consists of an energy value calculation module, a dynamic planning seam line search module, an image fusion module, and an image data cache module, as shown in Figure 3.

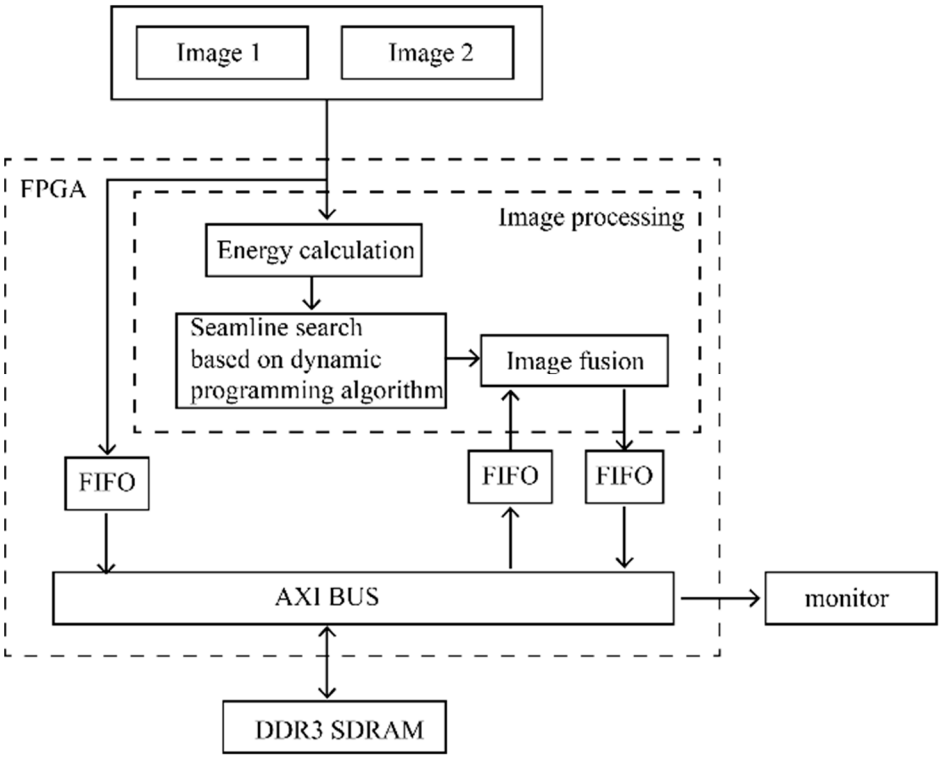

Figure 3. FPGA-based image fusion system.

The pixel data of the two registered images are sent to the asynchronous FIFO and written into DDR3 SDRAM through the AXI BUS protocol; at the same time, they are sent to the image processing module (including energy value calculation, seam line search based on dynamic programming algorithm, and image fusion). The energy value calculation module calculates the energy value based on the difference in the pixel values of the overlapping area images and outputs it to the seam line search module. The seam line search module completes the optimal seam line search of the dynamic programming algorithm and outputs the abscissa and image fusion enable signal of each row of seam

lines in the overlapping area image. The asynchronous FIFO reads and caches the pixel data of the two images cached in DDR3 SDRAM through the AXI BUS protocol. After the image fusion module receives the seam line coordinates, it reads the pixel data from the asynchronous FIFO, performs a weighted fusion operation on the pixels of the two images based on the seam line coordinates, sends the results to another asynchronous FIFO cache, and writes the fused pixels into DDR3 SDRAM through the AXI BUS protocol.

4.2. Energy Value Calculation Module

The energy value calculation module consists of three parts: pixel value conversion, line data cache, and energy value calculation.

In order to reduce the circuit design complexity and low-speed inefficiency caused by floating-point number operations, the decimals in the brightness value calculation formula are converted into fixed-point numbers for calculation, and the effective part is intercepted while ensuring the calculation's accuracy. In Equation (3), the division operation is implemented through a 10-bit right shift in hardware, thereby reducing the resource overhead caused by division in the FPGA. The specific calculation is as shown in Equation (8):

$$Y = (306 * R + 601 * G + 117 * B) \gg 10 \quad (8)$$

In the formula, $\gg$ represents a right shift operation.

The calculated and converted pixel value data are sent to the line buffer module, as shown in Figure 4. In order to improve the efficiency of the module, the line cache module is composed of four RAM modules, which alternately cache the pixel value data of the same two lines from the two pictures to meet the ping-pong operation and cope with the possible data overflow during high-speed data transmission. In Figure 4, Flag A and Flag B are line count state controllers. RAM A receives the brightness value data of the overlapping part in image 1, and RAM B waits. After completing writing a row of image data, Flag A changes, and the Ctrol1 module writes the next row of image data into RAM B. At the same time, Flag B changes, and the Ctrol2 module reads the data in RAM A. After RAM B receives a line of image data, Flag A changes. The Ctrol1 module writes the next line of image data into RAM A. Flag B changes. The Ctrol2 module reads the data in RAM B. The RAM C, RAM D, Flag C, Flag D, Ctrol3, and Ctrol4 modules process the brightness value data of the overlapping portion in image 2 with the same functions as above.

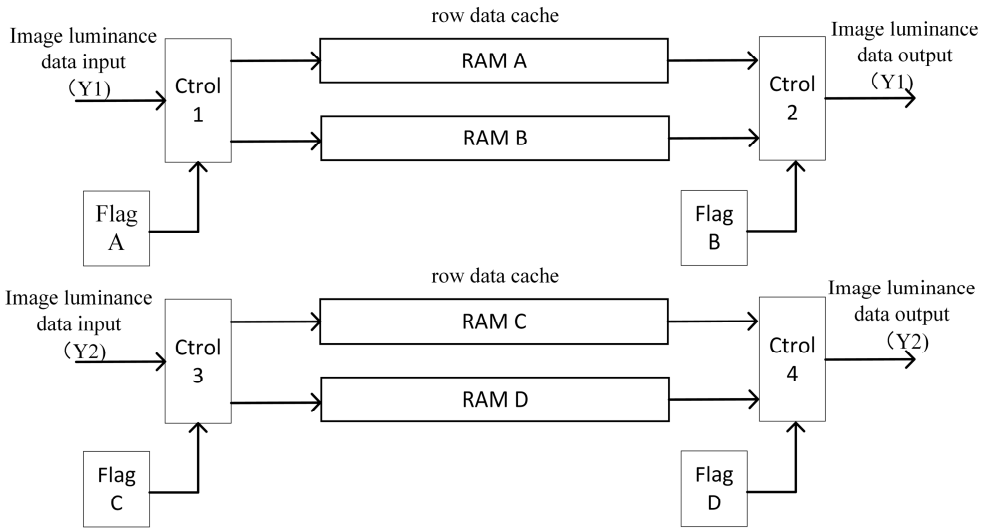

Figure 4. Schematic diagram of data cache design of energy value calculation module.

The circuit for energy value calculation is shown in Figure 5:

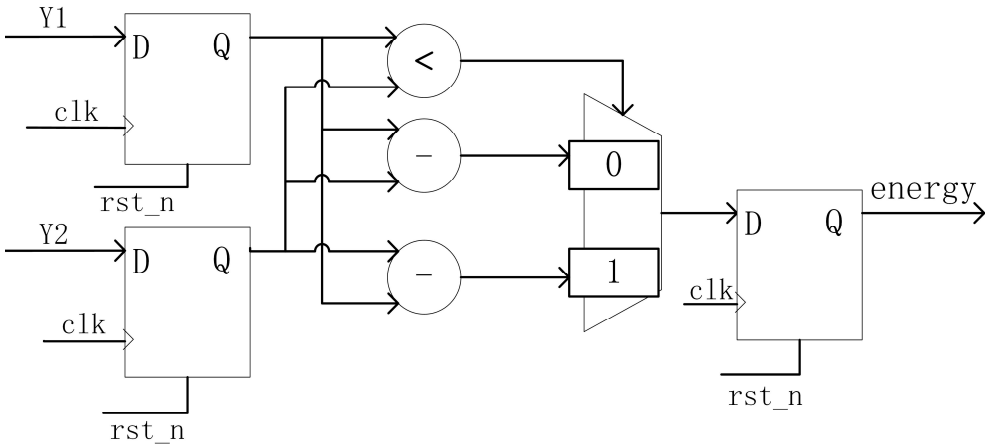

Figure 5. Energy value calculation circuit.

The input pixel value data are used as the minuend and the subtrahend, respectively, to complete the subtraction operation, and the comparison operation is performed at the same time. According to the output of the comparison operation as the selection condition of the selector, the operation results of the two subtractors are selected and output.

4.3. Design of Seam Line Search Module Based on Dynamic Programming Algorithm

Based on the obtained energy value, a dynamic programming algorithm is used to find a seam line to avoid cutting objects in the overlapping area as much as possible. This seam line has the characteristic of minimizing the cumulative energy value to ensure that important information is preserved during the fusion process.

The seam line search module is divided into three parts: data cache module, cumulative energy value iteration module, and seam line coordinate calculation module. As shown in Figure 6, when calculating the minimum cumulative difference value row by row, in order to effectively store the difference value between pixel positions and the cumulative difference value of the current row, the data cache module adopts the RAM group memory strategy. Specifically, the RAM group includes H rows of dual-port synchronous RAM with a depth of W (where H is the overlapping area image height; W is the overlapping image width; $RAM[i][j]$ means storing the elements of the i-th row and j-th column), supporting the efficient storage and reading of data. The accumulated energy value iteration module consists of an RW control, RAME, comparator, adder, and arbiter, which are used to control the reading and writing of the RAM group, store the accumulated energy values that are traversed to the current row, compare and calculate the minimum value among three adjacent accumulated energy values, iteratively update the accumulated energy value (the sum of its minimum value and the energy value of the corresponding coordinate pixel in the next row is rewritten into RAME), and output the relative path, respectively. After the accumulated energy value iterates through all pixels, the seam line coordinate calculation module calculates the minimum accumulated energy value and obtains the best seam line coordinates based on the relative path backtracking.

Since each RAM will inevitably go through three stages of operations, energy value input, energy value output, and relative path writing that covers each row of pixels, each three RAMs are designed to process three rows of pixels' energy value data, respectively, to achieve a three-level pipeline. The specific process is that the energy value of the pixel in row $i+1$ and column j in the overlapping area is written to $RAM[i+1][j]$; RAME sends three consecutive adjacent accumulated energy values to the comparator to compare and obtain the minimum value. The RW control module reads the pixel energy value $RAM[i][j]$ of the i-th row and j-th column, adds the minimum value to it, and rewrites it to $RAME[j]$; the arbiter module generates relative path data based on the position of the minimum value of comparator module and writes it to $R[i-1][j]$. There is a counter in the seam line search module. i and j start counting from 0. Every time the data are read or written, the counter j

increases by 1. When j reaches $W+1$, it returns to zero. At the same time, i increases by 1 and the cycle repeats until i equals H.

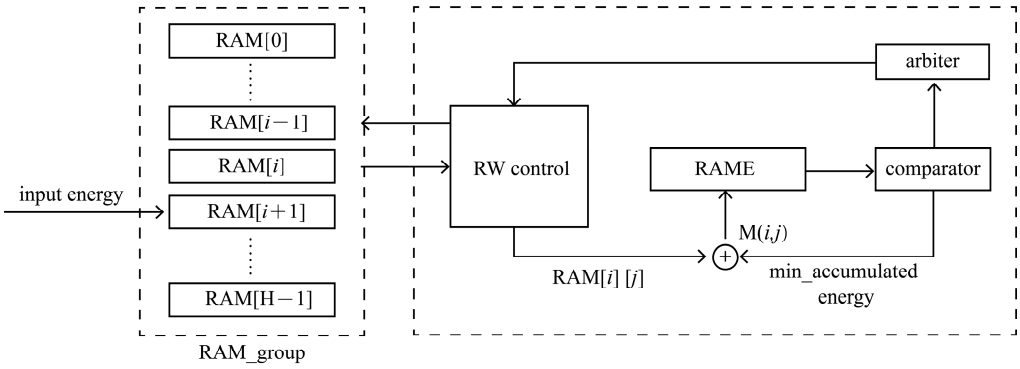

Figure 6. Hardware design diagram of seamline search module.

The following is a detailed description of the steps to find the best seam line path for the entire iteration:

1. Energy value input: the energy value calculation module sends and stores the processed energy value of the pixel point in the overlapping area with coordinates $(i,\ j)$ to the j-th column element of the i-th row of RAM, that is, $RAM[i][j]$=$E(i,j)$.
2. Comparison operation: comparator module sequentially reads three consecutive adjacent points $RAME[j-1],\ RAME[j],\ RAME[j+1]$ ($RAME[j]$ represents the j-th column element in the memory RAME) from the RAME module, representing the minimum accumulated energy value $M(i-1,j-1), M(i-1,j), M(i-1,j+1)$ of the upper left, right upper and upper right of pixel point (i,j), respectively. The minimum value among them is obtained by sequentially comparing in comparator module, and its relative position is recorded through the arbiter module, that is, the direction of upper left, right upper and upper right, and the corresponding relative path (2, 1, 0) is output. These path data are stored in $RAM[i-1][j]$, covering the original energy value data for saving storage resources, as shown in Equation (9):

$$RAM[i-1][j] = \begin{cases} 2, & M_{min} = M(i-1,j-1) \\ 1, & M_{min} = M(i-1,j) \\ 0, & M_{min} = M(i-1,j+1) \end{cases} \tag{9}$$

3. Iteration: read $RAM[i][j]$ from RAM_group, that is, the energy value of the current pixel $E(i,j)$, and add it to the minimum cumulated energy value obtained above to obtain the cumulated energy value of the current pixel $M(i,j)$, and write the sum to $RAME[j]$ to iteratively update the accumulated energy value of the current stitching node. As shown in Equation (10):

$$RAME[j] = RAM[i][j] + M_{min} \tag{10}$$

4. Repeat steps 1, 2, and 3: continue to iterate the above steps until the point in the last row of the overlapping area image of the current point of the seam line is processed. The circuit is shown in Figure 7.
5. Optimal seam line operation stage: compare the minimum cumulated energy values of all paths through the comparator, obtain the minimum value, and output the abscissa corresponding to the current pixel point as the abscissa of the last line of the optimal seam line.
6. Backtracking stage: in a bottom-up order, the abscissa values of each node of the optimal seam line are calculated from the abscissa coordinates of the last row of the best seam line that was obtained, and the relative path data recorded in the memory

RAM group. Output the abscissa of the complete optimal seam line to the image fusion module.

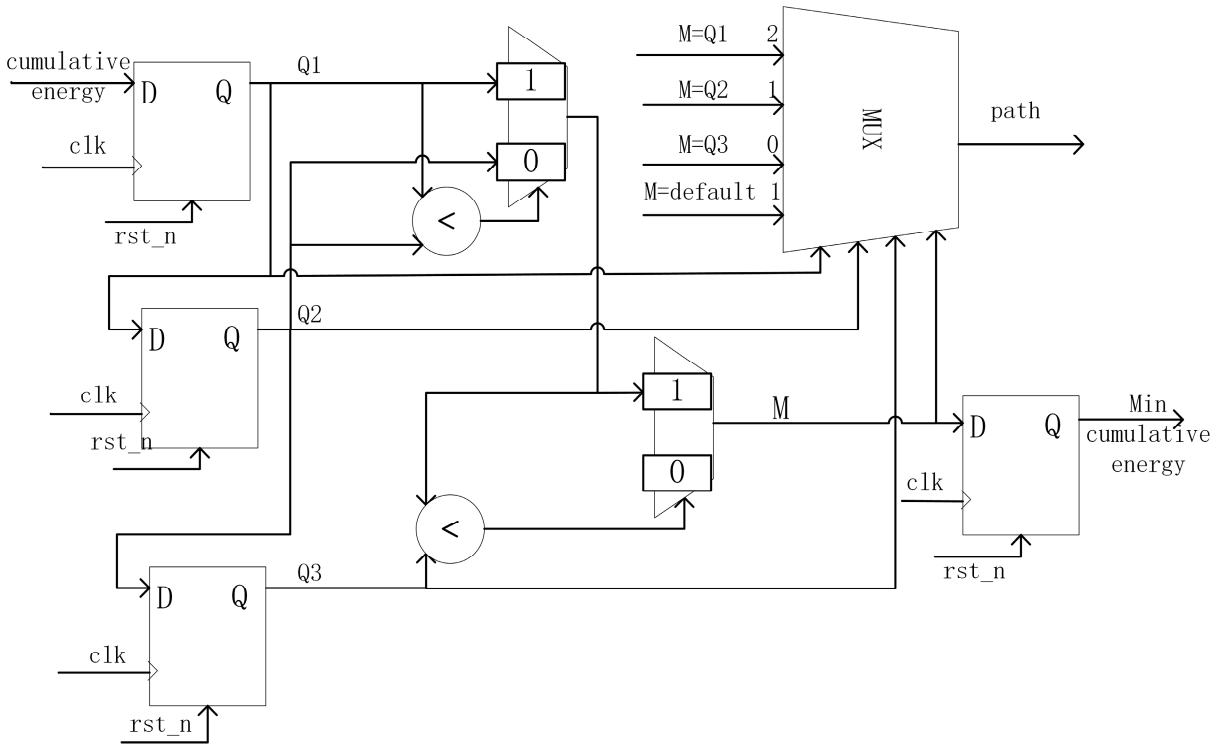

Figure 7. Circuit diagram of cumulative energy value iteration.

4.4. Image Fusion Module

The coordinates of the best seam line are cached in the image fusion module, and the two required image data are read from DDR3 by using the AXI bus protocol and asynchronous FIFO cache. Based on the nodes of the best seam line, an image fusion operation is performed. During the fusion process, for each row of pixels, three points on the left and right of the column coordinate point corresponding to the seam line are weighted and averaged to achieve image fusion.

In the image fusion module, a separate design of the seam line search module and fusion module is adopted to reuse existing seam lines within a certain number of frames, thereby effectively improving the image processing speed. This separation design is particularly suitable for fixed content scenes, which can reduce power consumption, achieve efficient acceleration effects, and eliminate the problem of fusion image ghosting caused by frequent seam line changes.

In a specific relatively fixed content scene, since the color and texture differences of the overlapping area images change little, and their positions do not change much, it can be assumed that the position of the optimal seam line is relatively fixed. In this case, previously calculated seam lines can be effectively reused, resulting in a reduced power consumption and efficient acceleration. During specific implementation, the registered pixel values are directly input to the image fusion module, and the image fusion is performed based on the optimal seam line coordinates calculated in previous frames stored in the image fusion module.

4.5. Image Data Cache Module

The image data cache module, a critical component of our system, is designed to address the challenges posed by the mismatch in clock and data transmission rates during image processing. This module comprises First-In, First-Out (FIFO) cache modules and a DDR3 control module.

4.5.1. Function and Importance

FIFO cache modules buffer pixel data after image registration and before image fusion, aligning the different speeds of data production and consumption. This ensures smooth data flow and minimizes processing delays.

DDR3 control module: This module works with the FIFO cache to manage large volumes of image data, crucial for maintaining high processing efficiency.

4.5.2. Interface Conversion and Operational States

An essential part of this system is the interface conversion module (AXI BUS), bridging the FIFO and DDR3. This transforms the FIFO interface into an AXI interface, facilitating effective data movement between the FIFO cache and DDR3 memory.

The module operates in several states—IDLE, CHECK FIFO, and TRANS—to efficiently manage data transfers. This state machine design ensures a balanced data flow for both reading and writing operations.

Incorporating this module into our system is critical for mitigating the discrepancies in data rates and clock synchronization, a common challenge in advanced image processing. It addresses the specific needs of real-time, high-quality image stitching, making the process more efficient and reliable. The module not only streamlines data management but also significantly contributes to the system's overall performance, especially in handling extensive image data with minimal latency.

4.5.3. Detailed Technical Workflow

In the DDR3 controller interface conversion module, a judgment and arbitration process is required for read/write operations.

During the reading operation, the system initially remains in the IDLE state. Once the calculation of the optimal seam line is completed, the seam line search module sends a data reading signal to the AXI module, signaling it to start the reading process. The system then transitions to the CHECK FIFO state, where it checks the volume of data items in the FIFO, and proceeds to perform burst transmissions in the TRANS state.

The pseudocode for this process is shown in Algorithm 2.

Algorithm 2. The pseudocode of data cache operation

```
SET state = IDLE
WHILE true
    IF state == IDLE
        WAIT for optimal seam line calculation completion
        ON completion, seam line search module sends data reading signal to AXI module
        SET state = CHECK_FIFO
    ELSE IF state == CHECK_FIFO
        IF number of data items in FIFO < image width
            SET state = TRANS
        ELSE
            CONTINUE checking FIFO
    ELSE IF state == TRANS
        PERFORM burst transmission from DDR3 to FIFO
        SET state = CHECK_FIFO
```

The writing steps mirror the reading process, where the data to be stored in DDR3 are sent to the FIFO, and AXI burst transmission is initiated when the FIFO data exceed a single burst's length.

The rationale behind this design is to enhance the system's ability to process large image data sets in real time with high efficiency, which is vital for practical applications in fields.

5. Experimental Results and Analysis

The circuit verification device is Altera's Cyclone IV xc7k325tffg676-2 FPGA, the development language is Verilog, the development software is Vavado 2020.2, and the simulation tool is Modelsim 2018. The designed circuit has configurable parameters, allowing for it to flexibly adapt to the stitching and fusion of images of different sizes.

By adjusting parameters, customized configurations of the image stitching and fusion process can be achieved to meet diverse application requirements. The fusion effect of the verification test is shown in Figure 8.

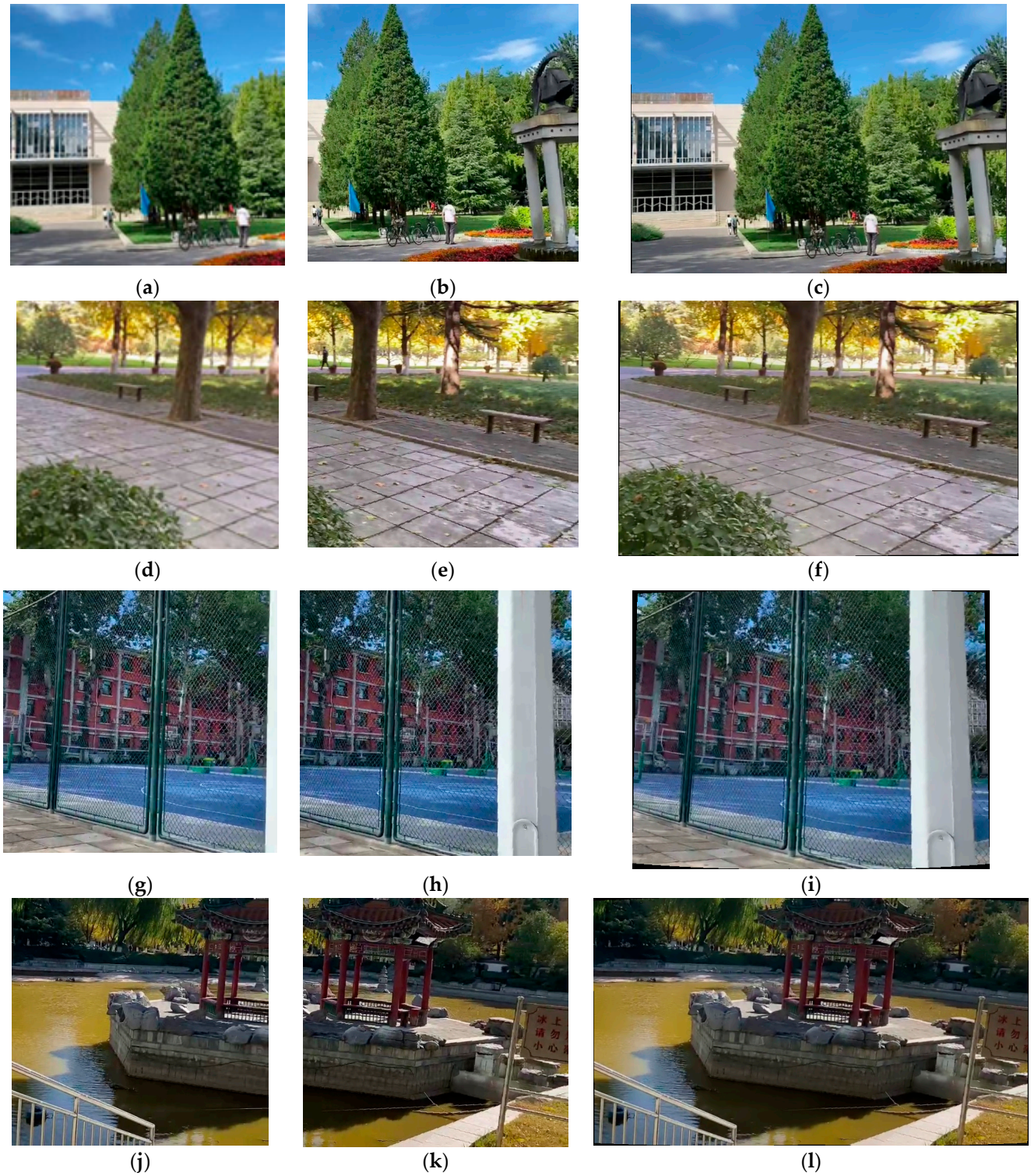

Figure 8. Test images and fusion results: (**a**,**d**,**g**,**j**) the original left image before fusion; (**b**,**e**,**h**,**k**) the original right image before fusion; (**c**,**f**,**i**,**l**) the result of fusion.

The visual effect after image fusion has the characteristics of smooth transition and natural connection. This result further proves the effectiveness of the hardware imple-

mentation of the algorithm, which successfully achieves the fusion of target objects while maintaining image continuity.

The FPGA resource usage is shown in Table 1. This table clearly shows the usage of each resource in the design, as well as the percentage occupancy of the corresponding resource. The results show that the occupancy rate of each resource is less than 80%, which proves that the system meets the reliability requirements. At the same time, the system also makes full use of the available system resources to avoid resource waste when using high-performance FPGAs.

Table 1. Memory resource utilizations of the image stitching system on FPGA.

Resource	Used Number	Total Number	Utilization Percentage (%)
LUT	36,839	203,800	18.08
BRAM	321.5	445	72.25
FF	23,643	407,600	5.80

The simulation test results are shown in Figure 9. When the input clock frequency is 100 MHz and the AXI BUS clock frequency is 200 MHz, two images (486 × 643) are stitched and fused into one image (579 × 643), and the overlapping area size is 393 × 643, as shown in Figure 9, and the total time required is 7.043795 ms. According to theoretical calculations, the processing rate of the circuit reaches 142 FPS. In comparison, the software implementation (Python 3.10, CPU: AMD Ryzen 9 7945 HX, Memory: 64 GB at 5200 MHz) took 99.948 ms to run the same algorithm. This signifies that the circuit design execution time is only 7.05% of the software implementation time, representing a 92.95% reduction in time compared to the software implementation.

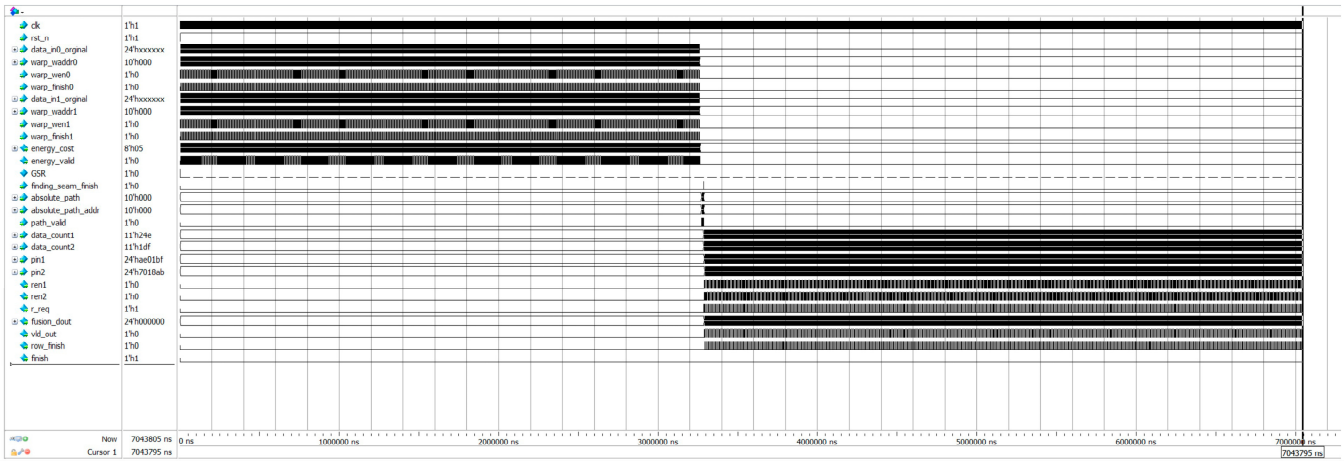

Figure 9. Simulation timing diagram of the complete process.

In a relatively fixed content scene, by reusing the best seam lines of previous frames for image fusion, a single image processing can take 3.758890 ms, and the theoretical rate reaches 266 FPS. As shown in Figure 10, these results demonstrate that the designed system exhibits a reliable performance and efficiency in the image processing process.

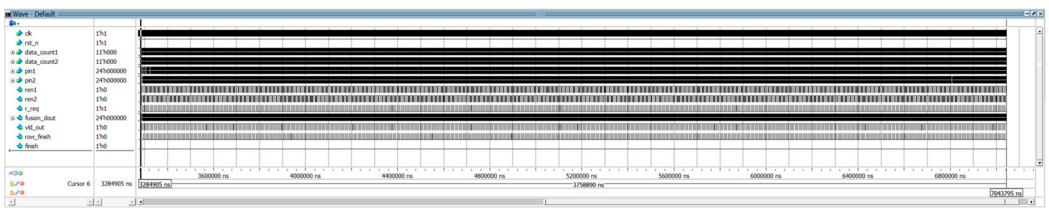

Figure 10. Simulation timing diagram of reuse seam line process.

Table 2 shows a comparison of the image size and achieved frame rate of circuits designed in the different literature studies. As can be seen from Table 2, for processing images smaller than or approximately the size of this article, the image fusion circuit designed in this article achieves a higher frame rate than other circuits.

Table 2. Quantitative comparison between previous work and our system.

	Input Image Resolution	Output Image Resolution	Frame Rate/FPS
X. Q. Yin [37]	640 × 480	/	25
X. Yang [11]	320 × 240	460 × 192	53
Z. Chuang [38]	640 × 480	/	49
S. Yifan [39]	640 × 360	/	31
Ours	486 × 643	579 × 643	142

Xu Yang and others' work [11] supports an image resolution of 320 × 240, with a frame rate of 53 FPS and an input clock of 24 MHz. The design proposed in this paper, while processing images significantly larger than those handled in their work, achieves a frame rate that is approximately 1.68 times higher.

Zhao Chuang's work is capable of processing images with a resolution of 640 × 480 at up to 49 FPS [38]. In comparison, the design presented in this paper improves the frame rate by 1.89 times.

Song Yifan's approach, which utilizes a Dijkstra-based optimal seam line fusion algorithm, can perform image stitching on 640 × 360 images at 31 FPS [39]. This frame rate is significantly lower than the frame rate achieved by the work in this paper for processing images of similar size, which showed an improvement of about 3.58 times.

Yin Xiaoqing proposed a seam line search module based on a second-order dynamic programming algorithm that can be transplanted to FPGA, which is theoretically capable of processing 640 × 480 video streams at 25 FPS [37]. This frame rate is substantially lower than the processing frame rate of the work in this paper, which achieves an improvement of approximately 4.68 times.

6. Conclusions

Image stitching is widely used in the field of image processing. There have been many algorithm studies, but there are few dedicated circuits for image stitching. On the premise of combining the hardware algorithm complexity and resource utilization, we optimized the energy value calculation method and designed a real-time image stitching circuit based on the fusion algorithm of dynamic programming for a seam line search.

In order to achieve a high real-time performance, multiple asynchronous FIFOs and DDR3 were used in the design to cache and synchronize the two image data; ping-pong operation and pipeline design were used to improve computing efficiency; the overhead of low-speed floating point operations and division operations is avoided, making the operation more efficient; the seam line search module and fusion module were designed separately to improve parallelism; and seam lines were reused in relatively fixed scenes to further increase the image processing rate. After a series of experiments, it was proved that the designed circuit successfully achieved high-quality, real-time image stitching and fusion processing under relatively limited hardware resources. Not only did its performance significantly surpass the four reference objects, but it also achieved significant improvements in frame rate.

The real-world applications of this technology are extensive, ranging from surveillance and medical imaging to autonomous vehicles and panoramic photography. For instance, in surveillance, our method offers improved monitoring capabilities with enhanced image detail and accuracy. In medical imaging, this can aid in creating comprehensive scans, potentially improving diagnostics.

Importantly, our circuit's efficiency in resource-limited environments makes it suitable for embedded systems and mobile devices, paving the way for broader application in

compact, cost-effective products. This not only advances the field of image stitching but also opens possibilities for integrating advanced image processing in various practical settings.

7. Future Work

In future developments, we propose an optimization of the energy value function to enhance seam line navigation in image stitching, particularly in high-contrast areas. This improvement aims to address the current limitations caused by the reliance on grayscale values, which often leads to ghosting and distortion when encountering stark contrasts. Advanced image recognition techniques will be integrated into the energy calculation to allow for a more intelligent circumvention of objects.

Additionally, we plan to expand the node selection process for seam lines. Moving beyond the current method of choosing from three adjacent nodes, we will explore a dynamic programming approach that considers a larger set of nodes, potentially enhancing the system's ability to navigate around larger objects and achieve more natural stitching results. This expansion may include the application of machine learning algorithms for optimized path prediction in complex scenarios.

On the hardware front, we are considering the use of a heterogeneous FPGA integrated with a CPU processing unit. This integration is expected to facilitate complex image recognition tasks and improve efficiency by offloading tasks to the CPU. We also aim to explore adaptive algorithms for dynamic resource allocation between the FPGA and CPU, depending on the task's complexity, to enhance both performance and energy efficiency.

Author Contributions: Conceptualization, Y.J. and R.W.; methodology, Y.J.; software, Y.J.; validation, Y.J., R.W. and X.J.; formal analysis, Y.J.; investigation, Y.J.; resources, X.J.; data curation, Y.J.; writing—original draft preparation, Y.J.; writing—review and editing, Y.J.; visualization, Y.J.; supervision, X.J.; project administration, X.J.; funding acquisition, X.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by National Science Foundation of China under Grant 61072135, 81971702, the Fundamental Research Funds for the Central Universities under Grant 2042017gf0075, 2042019gf0072, and Natural Science Foundation of Hubei Province under Grant 2017CFB721.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Glossary

Image Stitching: Combines multiple images with overlapping fields to produce a single high-resolution image. Common in panoramic photography and surveillance.

Ghosting Effect: This refers to a visual artifact in image stitching where duplicate or misaligned features appear, typically due to movement or misalignment in overlapping areas. This is a common challenge in the creation of seamless panoramas.

Seamlines: These are the lines along which images are joined in stitching processes. The optimal seamline minimizes visual discrepancies, ensuring a smooth and cohesive transition between stitched images.

Dynamic Programming: An algorithmic method used for efficiently finding optimal seam lines in image stitching, minimizing visual discrepancies.

Field-Programmable Gate Array (FPGA): A customizable integrated circuit used for parallel data processing, ideal for high-speed image analysis and real-time tasks.

Asynchronous First-In, First-Out (FIFO): A queue structure for data synchronization, allowing data reading and writing at different speeds, crucial for handling image data from varied sources.

DDR3 Memory: A high-speed memory technology used for efficient storage and access of large image data in real-time processing.

Ping-Pong Operation and Pipeline Design: Alternating between two buffers (ping-pong operation) and simultaneous processing stages (pipeline design) to enhance efficiency.

Seam Line Search and Fusion Module: Identifies optimal paths for stitching and merges images along these lines, improving speed and performance through parallel processing.

References

1. Cheng, H.L.; Hao, Q.; Hu, Y.; Cao, J.E.; Wang, S.P.; Li, L. Design of optoelectronic imaging system with high resolution and large field of view based on dual CMOS. In Proceedings of the Conference on Optoelectronic Imaging and Multimedia Technology IV, Beijing, China, 12–13 October 2016.
2. Gu, L.Y.; Zeng, A.J.; Hu, S.Y.; Yuan, Q.; Cheng, W.L.; Zhang, S.H.; Hu, G.H.; He, H.B.; Huang, H.J. Imaging ellipsometer with large field of view. In Proceedings of the Conference on Optical Metrology and Inspection for Industrial Applications IV Held as Part of SPIE/COS Photonics Asia Conference, Beijing, China, 12–14 October 2016.
3. Korompili, G.; Kanakaris, G.; Ampatis, C.; Chronis, N. A portable, optical scanning microsystem for large field of view, high resolution imaging of biological specimens. *Sens. Actuators A-Phys.* **2018**, *279*, 367–375. [CrossRef]
4. Kannala, J.; Brandt, S.S. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 1335–1340. [CrossRef] [PubMed]
5. Gennery, D.B. Generalized camera calibration including fish-eye lenses. *Int. J. Comput. Vis.* **2006**, *68*, 239–266. [CrossRef]
6. Yan, W. The Realization of Real-Time Stitching Imaging System Based on Binocular Camera. Master's Thesis, Xidian University, Xi'an, China, 2022.
7. Perazzi, F.; Sorkine-Hornung, A.; Zimmer, H.; Kaufmann, P.; Wang, O.; Watson, S.; Gross, M. Panoramic video from unstructured camera arrays. *Comput. Graph. Forum* **2015**, *34*, 57–68. [CrossRef]
8. Matzen, K.; Cohen, M.F.; Evans, B.; Kopf, J.; Szeliski, R. Low-cost 360 stereo photography and video capture. *ACM Trans. Graph.* **2017**, *36*, 1–12. [CrossRef]
9. Wang, M.; Liang, J.B.; Zhang, S.H.; Lu, S.P.; Shamir, A.; Hu, S.M. Hyper-lapse from multiple spatially-overlapping videos. *IEEE Trans. Image Process.* **2018**, *27*, 1735–1747. [CrossRef]
10. Zhang, J.D.; Xiu, Y. Image stitching based on human visual system and SIFT algorithm. *Vis. Comput.* **2023**. [CrossRef]
11. Xu, Y.; Wang, X.; Zhu, J.; Liu, P.; Jiang, H. Research and design of image mosaic technology based on FPGA. *J. Chang. Univ. Sci. Technol. (Nat. Sci. Ed.)* **2018**, *41*, 94–98+103.
12. Yong, L.; Lei, W.; Hanlin, Q. The design and implementation for the splicing of the panoramic video images based on FPGA. *Electron. Des. Eng.* **2018**, *26*, 80–83. [CrossRef]
13. Brown, M.; Lowe, D.G. Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vis.* **2007**, *74*, 59–73. [CrossRef]
14. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
15. Yang, Z.; Hu, C.; Liu, D. FPGA Image Stitching Design Based on Improved SURF Algorithm. In Proceedings of the 2022 International Conference on Artificial Intelligence and Computer Information Technology (AICIT), Yichang, China, 16–18 September 2022; pp. 1–4.
16. Hussain, M.; Ali, N.; Hong, J.-E. Vision beyond the field-of-view: A collaborative perception system to improve safety of intelligent cyber-physical systems. *Sensors* **2022**, *22*, 6610. [CrossRef]
17. Gao, H.; Huang, Z.; Yang, H.; Zhang, X.; Cen, C. Research on Improved Multi-Channel Image Stitching Technology Based on Fast Algorithms. *Electronics* **2023**, *12*, 1700. [CrossRef]
18. Hussain, M.; Hong, J.-E. Enforcing Safety in Cooperative Perception of Autonomous Driving Systems through Logistic Chaos Map-based End-to-End Encryption. In Proceedings of the 2022 16th International Conference on Open Source Systems and Technologies (ICOSST), Lahore, Pakistan, 14–15 December 2022; pp. 1–6.
19. Fischler, M.A.; Bolles, R.C. Random sample consensus—A paradigm for model-fitting with applications to image-analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
20. Xu, X.; Yuan, S.; Wang, J.; Dai, Y.P. Image stitching method based on global and local features. *Trans. Beijing Inst. Technol.* **2022**, *42*, 502–510. [CrossRef]
21. Bayrak, M.; Kilinç, O.; Arica, N. Real-time image stitching for multiple camera panoramic video shoot: A case study in football matches. In Proceedings of the 28th Signal Processing and Communications Applications Conference (SIU), Electr Network, Gaziantep, Turkey, 5–7 October 2020.
22. Burt, P.J.; Adelson, E.H. The laplacian pyramid as a compact image code. *IEEE Trans. Commun.* **1983**, *31*, 532–540. [CrossRef]
23. Burt, P.J.; Adelson, E.H. A multiresolution spline with application to image mosaics. *ACM Trans. Graph.* **1983**, *2*, 217–236. [CrossRef]
24. Duplaquet, M.L. Building large image mosaics with invisible seam-lines. In Proceedings of the Visual Information Processing VII, Orlando, FL, USA, 13–14 April 1998; pp. 369–377.
25. Davis, J.; IEEE Comp, S.O.C. Mosaics of scenes with moving objects. In Proceedings of the 1998 IEEE Computer-Society Conference on Computer Vision and Pattern Recognition, Santa Barbara, CA, USA, 23–25 June 1998; pp. 354–360.
26. Geman, S.; Geman, D. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.* **1984**, *6*, 721–741. [CrossRef]

27. Boykov, Y.Y.; Jolly, M.P.; IEEE Computer, S. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In Proceedings of the 8th IEEE International Conference on Computer Vision (ICCV 2001), Vancouver, BC, Canada, 7–14 July 2001; pp. 105–112.
28. Boykov, Y.; Kolmogorov, V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 1124–1137. [CrossRef]
29. Lee, K.W.; Jung, S.W.; Kim, S.K.; Ko, S.J. A novel content-aware stitching algorithm for real-time video sequences. *IEICE Trans. Inf. Syst.* **2011**, *E94D*, 357–362. [CrossRef]
30. Efros, A.A.; Freeman, W.T. Image quilting for texture synthesis and transfer. In Proceedings of the Siggraph 2001, Los Angeles, CA, USA, 12–17 August 2001; pp. 341–346.
31. Avidan, S.; Shamir, A. Seam carving for content-aware image resizing. *ACM Trans. Graph.* **2007**, *26*, 10-es. [CrossRef]
32. Gu, H.; Yu, Y.; Sun, W.D. A new optimal seam selection method for airborne image stitching. In Proceedings of the IEEE International Workshop on Imaging Systems and Techniques, Shenzhen, China, 11–12 May 2009; pp. 159–163.
33. Yin, X.Q.; Li, W.L.; Wang, B.; Liu, Y.; Zhang, M.J. A novel video stitching method for multi-camera surveillance systems. *Ksii Trans. Internet Inf. Syst.* **2014**, *8*, 3538–3556. [CrossRef]
34. Altahir, A.A.; Asirvadam, V.S.; Hamid, N.H.B.; Sebastian, P.; Saad, N.B.; Ibrahim, R.B.; Dass, S.C. Optimizing Visual Sensor Coverage Overlaps for Multiview Surveillance Systems. *IEEE Sens. J.* **2018**, *18*, 4544–4552. [CrossRef]
35. Lindeberg, T. Scale-space for discrete signals. *IEEE Trans. Pattern Anal. Mach. Intell.* **1990**, *12*, 234–254. [CrossRef]
36. Hwang, J.W.; Lee, H.S. Adaptive image interpolation based on local gradient features. *IEEE Signal Process. Lett.* **2004**, *11*, 359–362. [CrossRef]
37. Yin, X.Q.; Li, W.L.; Liu, Y.; Wang, B.; Zhang, M.J. FPGA-based real time video stitching method for video surveillance. *Optik* **2015**, *126*, 2804–2808. [CrossRef]
38. Chuang, Z. Design and implementation of Real Time Image Stitching System Based on FPGA. Master's Thesis, Tianjin University, Tianjin, China, 2020.
39. Yifan, S. Design and Implementation of Omnidirectional Binocular Vision System Based on FPGA. Master's Thesis, South China University of Technology, Guangzhou, China, 2021.