Seminar Report: Newest Trends in High-Performance Data Analytics

Computational Performance Characterization of GPU-accelerated Image Analysis

Ayan Gupta

Supervisor: Michael Bidollahkhani

Georg-August-Universität Göttingen
Institute of Computer Science

March 31, 2024

Contents

List of Figures **2**

List of Listings **2**

1 Introduction **1**

1.1 Image Analysis 1

1.2 CPUs vs GPUs: Architectures Tailored for Different Needs 3

1.3 GPU Acceleration: Parallelism and High Memory Bandwidth 4

2 Literature Review **7**

3 Experiments **8**

3.1 U-Net based Brain tumor segmentation from 2D MRI scans 8

3.2 Real time video analysis for object detection and tracking using YOLOv7 . 9

4 Tools and Technologies **10**

5 Results and Discussion **11**

6 Conclusion **14**

References **15**

List of Figures

1 Image Formation . . . 2
2 What the human eye sees . . . 2
3 What computers see . . . 2
4 CPU Fetching data from system memory . . . 4
5 GPU Fetching data from system memory . . . 5
6 Parallelism in GPU . . . 5
7 Working of SIMD . . . 6
8 Block multiplication . . . 7
9 U-Net architecture [RFB15] . . . 9
10 Working of YOLO model . . . 10
11 Training and validation execution times by processing unit . . . 11
12 Train and validation throughputs of CPU and GPU . . . 11
13 Training and Validation losses for CPU . . . 12
14 Training and Validation losses for GPU . . . 12
15 Training and validation IOUs for CPU . . . 12
16 Training and validation IOUs for GPU . . . 12
17 CPU and GPU execution times for YOLOv7 . . . 13
18 CPU and GPU processing FPS for YOLOv7 . . . 13
19 Visual results from U-Net brain tumor segmentation task . . . 14
20 Visual results from YOLOv7 object detection and tracking task . . . 14

List of Listings

1 Scalar and big matrix operation on CPU . . . 3
2 Scalar and big matrix operation on GPU . . . 3

1 Introduction

In the era of big data, the ability to extract meaningful information from vast amounts of unstructured data, particularly images, has become increasingly crucial across various domains, ranging from healthcare and scientific research to surveillance and autonomous systems. Image analysis, the process of extracting valuable insights from digital images through automated techniques, has emerged as a powerful tool for addressing this challenge. However, manual analysis of large-scale image datasets is often error-prone, labor-intensive, and time-consuming, necessitating the development of efficient and accurate computational approaches. One such approach that has emerged as a transformative force in recent years is GPU (Graphics Processing Unit) acceleration. Unlike traditional CPUs (Central Processing Units), which are designed for general-purpose computing, GPUs are specialized hardware architectures optimized for parallel processing and high-throughput computations. The massively parallel nature of GPUs, coupled with their high memory bandwidth and fast memory access, makes them particularly well-suited for image analysis tasks that involve extensive matrix operations and data-parallel computations. The primary objective of this research is to evaluate the computational performance characterization of GPU-accelerated image analysis techniques and compare them with their CPU-based counterparts. By conducting experiments and analyzing the results, we aim to demonstrate the superiority of GPUs in accelerating image analysis algorithms, leading to significant performance gains in terms of execution time, throughput, and overall efficiency. In this report, we will first provide an overview of image analysis and GPU acceleration, establishing the context and significance of our research. We will then review relevant literature, highlighting the state-of-the-art techniques and findings in GPU-accelerated image analysis. Subsequently, we will describe the experiments conducted, detailing the algorithms and methodologies employed, as well as the tools and technologies used. The results obtained from our experiments will be presented and analyzed, with a focus on key performance metrics such as execution time, throughput, and resource utilization, as well as task-oriented metrics. These findings will be used to substantiate our argument for the superior computational performance of GPU-accelerated image analysis approaches compared to CPU-based methods. Finally, we will conclude by summarizing our key findings, discussing the implications of our research, and outlining potential future directions for further exploration in this field.

1.1 Image Analysis

To grasp the essence of image analysis, one must first comprehend the intricate nature of images themselves. An image is a two-dimensional representation of a three-dimensional scene, capturing and encapsulating it into a digital format through a process known as image formation. This process involves the acquisition of a 3D scene by an imaging device, such as a camera or a scanner, which then converts the captured information into a 2D array of pixels. Each pixel in this array contains numerical values that define various properties of the scene, such as color, brightness, and depth or distance from the imaging device. Interestingly, the perception and interpretation of images can vary vastly between humans and computers.

To the human eye, an image can reveal a wealth of information – colors, actions, objects, segments, and even emotions like happiness, freshness, or positivity. For instance, an image of a lush green meadow on a sunny day might evoke feelings of serenity and

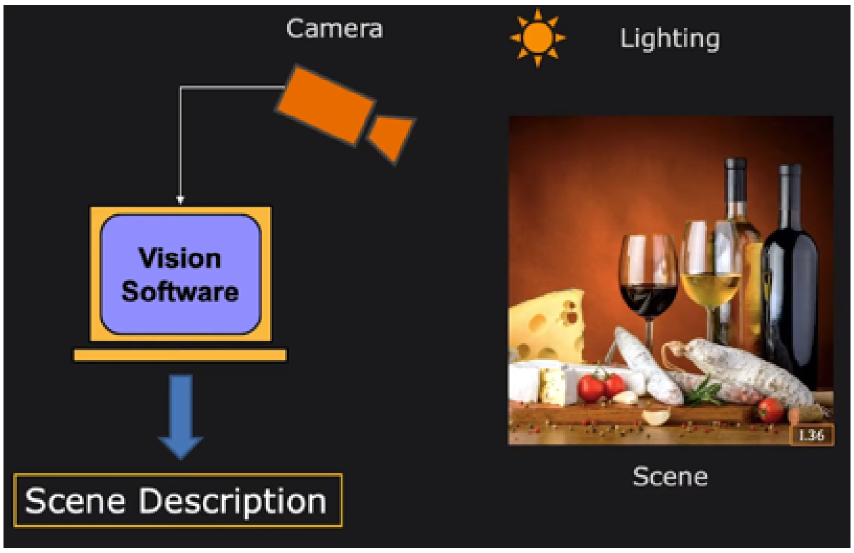
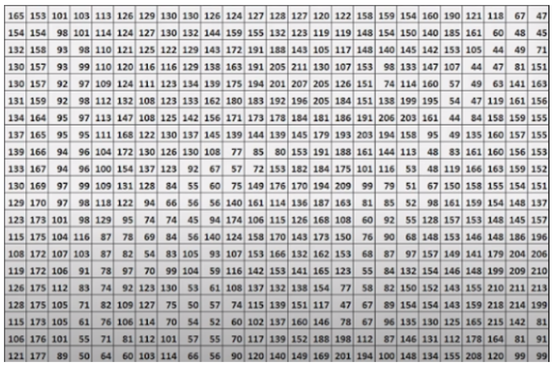


Figure 1: Image Formation

tranquility, while an image of a bustling city street might convey a sense of energy and vibrancy. However, to a computer, an image is initially perceived as nothing more than a non-informative collection of sparse pixels – a grid of numerical values devoid of inherent meaning or context. This stark contrast between human and computer perception highlights the fundamental challenge of image analysis: enabling computers to interpret and derive meaningful information from images, akin to the way humans effortlessly perceive and understand visual data. It is this challenge that underscores the significance of image analysis and the pursuit of developing sophisticated computational techniques to mimic and augment human visual capabilities.

Figure 2: What the human eye sees



165	153	101	103	113	126	129	130	130	126	124	127	128	127	120	122	158	159	154	160	190	121	118	67	47
154	154	98	101	114	124	127	130	132	144	159	155	132	123	119	119	148	154	150	140	185	161	60	48	45
132	158	93	98	110	121	125	122	129	143	172	191	188	143	105	117	148	140	145	142	153	105	44	49	71
130	157	93	99	110	120	116	116	129	138	163	191	205	211	130	107	153	98	133	147	107	44	47	81	151
130	157	92	97	109	124	111	123	134	139	175	194	201	207	205	126	151	74	114	160	57	49	63	141	163
131	159	92	98	112	132	108	123	133	162	180	183	192	196	205	184	151	138	199	195	54	47	119	161	156
134	164	95	97	113	147	108	125	142	156	171	173	178	184	181	186	191	206	203	161	44	84	158	159	155
137	165	95	95	111	168	122	130	137	145	139	144	139	145	179	193	203	194	158	95	49	135	160	157	155
139	166	94	96	104	172	130	126	130	108	77	85	80	153	191	188	161	144	113	48	83	161	160	156	153
133	167	94	96	100	154	137	123	92	67	57	72	153	182	184	175	101	116	53	48	119	166	163	159	152
130	169	97	99	109	131	128	84	55	60	75	149	176	170	194	209	99	79	51	67	150	158	155	154	151
129	170	97	98	118	122	94	66	56	56	140	161	114	136	187	163	81	85	52	98	161	159	154	148	137
123	173	101	98	129	95	74	74	45	94	174	106	115	126	168	108	60	92	55	128	157	153	148	145	157
115	175	104	116	87	78	69	84	56	140	124	158	170	143	173	150	76	90	68	148	153	146	148	186	196
108	172	107	103	87	82	54	83	105	93	107	153	166	132	162	153	68	87	97	157	149	141	179	204	206
119	172	106	91	78	97	70	99	104	59	116	142	153	141	165	123	55	84	132	154	146	148	199	209	210
126	175	112	83	74	92	123	130	53	61	108	137	132	138	154	77	58	82	150	152	143	155	210	211	213
128	175	105	71	82	109	127	75	50	57	74	115	139	151	117	47	67	89	154	154	143	159	218	214	199
115	173	105	61	76	106	114	70	54	52	60	102	137	160	146	78	67	96	135	130	125	165	215	142	81
106	176	101	55	71	81	112	101	57	55	70	117	139	152	188	198	112	87	146	131	112	178	164	81	91
121	177	89	50	64	60	103	114	66	56	90	120	140	149	169	201	194	100	148	134	155	208	120	99	99

Figure 3: What computers see

Images are not merely collections of pixels; they are rich vessels of information that can convey complex concepts, emotions, and narratives fundamental to human communication. Each pixel contains values defining properties like color, brightness, and distance, collectively forming a visually dense tapestry of data. While humans effortlessly perceive and interpret images through the intricate workings of the visual cortex, recognizing edges, patterns, objects, and colors, computers initially see only non-informative sparse pixels. This is because human visual perception is a highly complex and nuanced process, involving the integration of various cognitive and neurological processes that enable us to make sense of visual stimuli. To bridge this gap, we must ask: What is the significance of enabling computers to analyze images akin to human perception? The answer lies in the exponential growth of unstructured image data and images' ability to convey information more efficiently than text. However, manual analysis is error-prone, slow for complex scenes, and struggles to scale. Computers can interpret images as arrays of pixels defining properties which can be analyzed to detect patterns mimicking our visual capabilities. Automating this process through image analysis algorithms provides key benefits - accelerating labor-intensive tasks, reducing errors, and enabling scalability. Consequently, the field of image analysis aims to develop techniques that leverage modern hardware and software to empower computers to analyze and comprehend images on par with human perception. This powerful synthesis unlocks myriad opportunities across domains reliant on extracting insights from visual data.

1.2 CPUs vs GPUs: Architectures Tailored for Different Needs

Central Processing Units (CPUs) are designed as general-purpose processors, excelling at sequential operations and scalar computations. They typically consist of a few cores (usually between 2 and 16) optimized for executing a wide range of tasks efficiently. However, CPUs are not well-suited for data-parallel workloads that involve performing the same operation on large datasets simultaneously. On the other hand, Graphics Processing Units (GPUs) are specialized hardware architectures designed specifically for parallel processing and high-throughput computations. GPUs are composed of thousands of smaller, more efficient cores (often in the range of tens of thousands) optimized for executing the same instruction on multiple data elements simultaneously, a computing paradigm known as Single Instruction, Multiple Data (SIMD). This massive parallelism, combined with a high memory bandwidth and faster memory access, makes GPUs exceptionally efficient at handling data-parallel tasks that involve extensive matrix and vector operations. Such operations are prevalent in various domains, including image analysis, scientific simulations, and deep learning, where the same computation needs to be performed on large datasets concurrently.

```
%%timeit
x = torch.randn(1,1)
out = torch.matmul(x,x)
# Output: 7.22 µs ± 302 ns
```

```
%%timeit
x = torch.randn(10000,10000)
out = torch.matmul(x,x)
# Output: 34 s ± 676 ms
```

Listing 1: Scalar and big matrix operation on CPU

```
%%timeit
x = torch.randn(1,1).to('cuda')
out = torch.matmul(x,x)
# Output: 74.7 µs ± 53.4 µs
```

```
%%timeit
x = torch.randn(10000,10000).to('cuda')
out = torch.matmul(x,x)
# Output: 1.22 s ± 215 ms
```

Listing 2: Scalar and big matrix operation on GPU

The code snippets above illustrate this fundamental difference in performance between CPUs and GPUs. For a scalar operation, which is a simple computation on a single value, the CPU outperforms the GPU, as visible in Listing 1. However, when the operation involves larger matrices or tensors, the GPU's parallel processing capabilities result in significantly faster execution times compared to the CPU, as shown in Listing 2. GPUs achieve this performance advantage through their highly parallel architecture, where thousands of cores can work simultaneously on different portions of the data. Additionally, GPUs have a higher memory bandwidth, allowing them to fetch and process data from memory more efficiently, which is crucial for data-intensive tasks like image analysis. While CPUs are designed to handle a wide range of general-purpose tasks efficiently, GPUs are specialized for specific types of workloads that can take advantage of their parallel processing capabilities. This specialization enables GPUs to outperform CPUs in domains like image analysis, where the same operations need to be applied to large datasets concurrently. It's important to note that GPUs and CPUs are not mutually exclusive; they are often used in tandem, with CPUs handling the sequential and control-flow tasks, while offloading the computationally intensive data-parallel workloads to GPUs.

1.3 GPU Acceleration: Parallelism and High Memory Bandwidth

As mentioned earlier, many image analysis algorithms involve extensive matrix and vector operations, such as edge detection, feature extraction, image segmentation, denoising, and deep learning. These operations are computationally intensive and benefit greatly from both hardware and software advantages offered by GPUs. To understand how GPUs accelerate these algorithms, let's consider the analogy of a CPU as a high-performance Ferrari and a GPU as a massive dump truck. The task at hand is to fetch data from the system memory, load it into their respective memories, and perform the necessary operations. In the first case, we perform scalar multiplication.

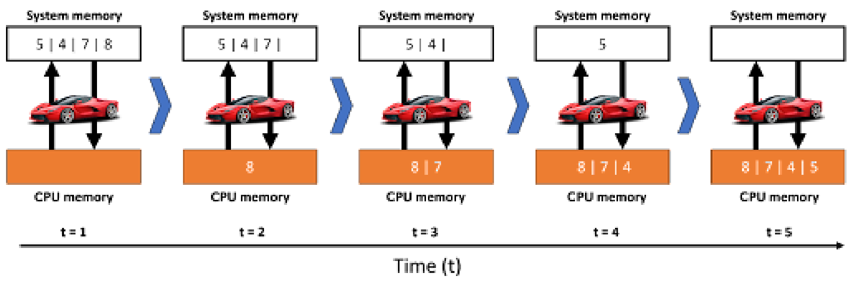


Figure 4: CPU Fetching data from system memory

Here, the CPU swiftly retrieves data via a fetch operation to memory. Although the amount of data fetched at a time is relatively small, it's sufficient for fetching floating-point numbers within a couple of trips, akin to the speed of a Ferrari. Conversely, GPUs possess a broader data transmission capability. However, their efficiency isn't optimized for swift operations. Consequently, while they can handle larger volumes of data, their processing of smaller data chunks can be comparatively slower. Thus, for scalar multiplications, GPUs may sometimes lag behind CPUs in terms of performance. In the second case, we're looking at multiplying two large matrices. The CPU can fetch data really quickly, but it can only grab a tiny bit of data each time it goes back and forth to memory. Consequently, it may require thousands of trips to fetch all the necessary data. While it's doing this, the CPU's memory gets processed and ready for the next set of data. With their superior memory bandwidth, GPUs can efficiently fetch significantly larger amounts

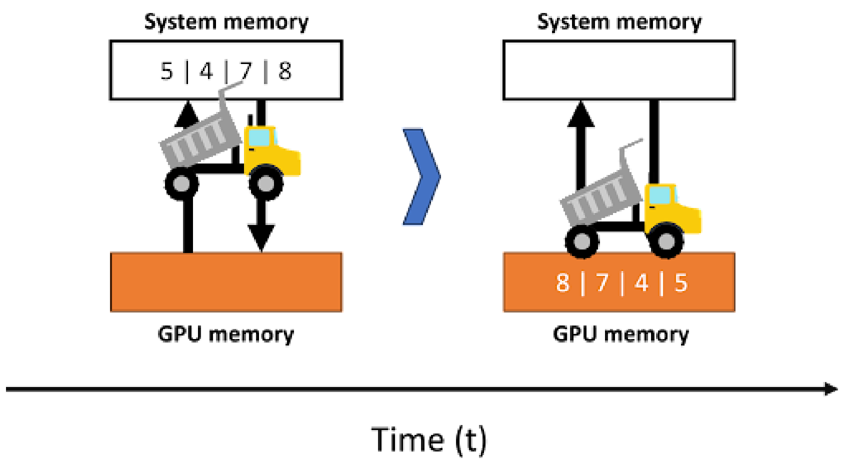


Figure 5: GPU Fetching data from system memory

of data from RAM into their memory. As a result, the GPU circumvents the need for as many trips, enabling expedited matrix multiplication operations.

Bandwidth refers to the amount of data that can be moved to and from memory in a single trip, and it's a key reason why GPUs outperform CPUs in handling large matrix multiplication tasks. However, even with a high bandwidth, GPU processors sometimes end up waiting for data. To address this, GPUs employ parallelization, which is like having a fleet of trucks instead of a single one. While the GPU is processing the current matrix chunks, it is fetching more chunks from the system memory ensuring that the GPU is never idle. This ensures that GPU processors are continuously fed with data to work on, reducing idle time.

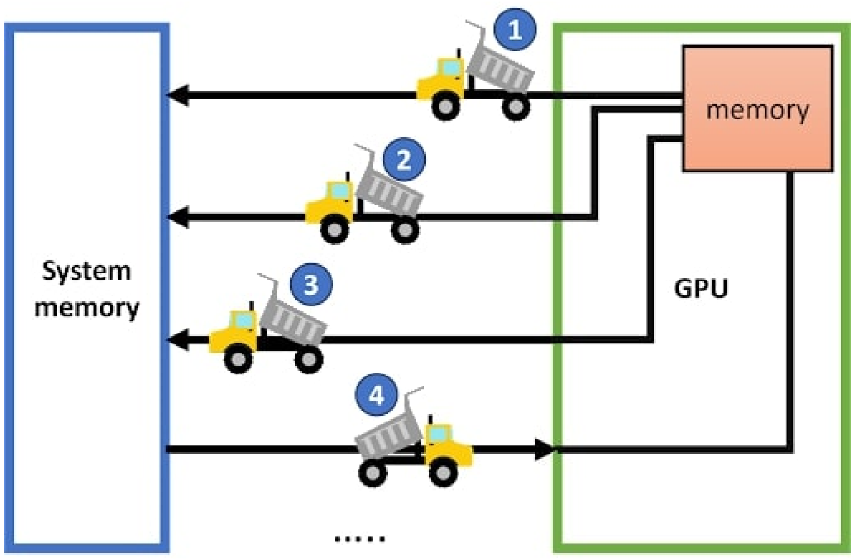


Figure 6: Parallelism in GPU

Additionally, GPUs boast a larger number of registers compared to CPUs, which further enhances computational speed. GPUs also employ SIMD (Single Instruction, Multiple Data) techniques, which allow them to perform the same operation on multiple data elements simultaneously. For instance, in image analysis tasks like filtering or color space conversion, the same operation needs to be applied to each pixel. With SIMD, a GPU can process multiple pixels in parallel using the same instruction, significantly accelerating these types of operations compared to traditional scalar processing on CPUs. The combination of these factors—high memory bandwidth, parallelization, and faster memory access, enables GPUs to outperform CPUs for many image analysis algorithms that involve extensive matrix and vector operations on large datasets. A GPU has a few streamlined multiprocessors, which constitute the processing part of the GPU. It has a global memory that is shared by these multi processors. They also have local memory that is specific to the multi processors but cannot be shared with other streamlined multiprocessors. This hardware can only be utilized to its highest potential if the algorithms are designed to exploit this hardware well. Here, we will take the example of matrix multiplication, which is an important part of Image Analysis along with other matrix operations. The brute force approach to solve matrix multiplication is by multiplying every pair of numbers one at a time and summing it up. The issue here is the time complexity of this approach. Now, leveraging the capabilities of GPUs would be ideal in this situation, however, simply throwing more processing power and faster data transfer at the problem isn't the whole story. To truly unlock the potential of GPUs, a technique called "block multiplication" comes into play. Here, the large matrices are strategically divided into smaller sub-blocks that fit comfortably within the shared memory of individual GPU cores. This shared memory acts like a smaller, high-speed workspace readily accessible to the core, reducing the need to constantly fetch data from the main memory. This strategy not only accelerates processing but also optimizes resource utilization, making efficient use of the GPU's computational power.

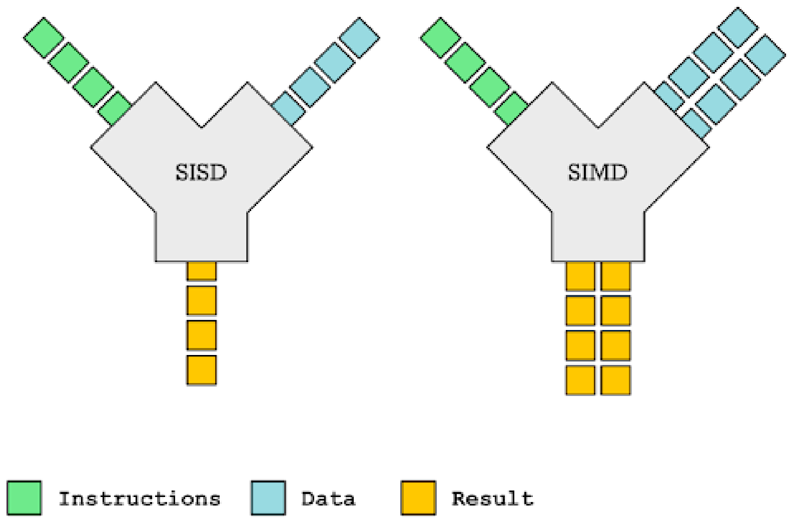


Figure 7: Working of SIMD

In summary, the evolution of GPU technology has significantly enhanced computational efficiency, enabling the swift execution of complex algorithms previously constrained by CPU limitations. This breakthrough empowers fields like medical and satellite image analysis to tackle large-scale data challenges with unprecedented speed and accuracy. By leveraging high memory bandwidth, parallelization, and innovative hardware utilization strategies like block multiplication, GPUs have become indispensable tools for real-time processing in applications such as surveillance and autonomous driving. This transforma-

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C_{ij} = \sum_{k=1}^{n\_blocks} A_{ik}B_{kj}$$

$$C = A.B$$

$$C = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

Figure 8: Block multiplication

tive capability marks a paradigm shift, driving advancements in diverse domains reliant on intensive data analysis and computation.

2 Literature Review

In the evolving field of image analysis, the transition from CPU-based to GPU-accelerated computation has marked a significant leap towards addressing the challenges of processing large-scale image datasets efficiently. The study by Chang and Sheu [CS20] on GPU acceleration for patient-specific airway image segmentation exemplifies this transition. Their work demonstrates a groundbreaking speedup of 61.8 times over CPU-based methods by leveraging GPU parallelism and optimization techniques, such as memory arrangement and block settings adjustment. This not only underscores the computational benefits of GPUs but also highlights the importance of algorithmic adjustments to harness these benefits fully. Stefaniga's [Seb21] contribution to medical image analysis through CUDA-based acceleration of feature extraction algorithms, specifically Canny Edge Detection and Hough Transform, further reinforces the GPU's prowess. By focusing on lung tumor detection in radiographic images, Stefaniga showcases reductions in computation times by 75% to 99%, emphasizing the synergy between algorithmic innovation and GPU computing capabilities. In a comparative study by Mounir et al.[Arn+98] across multiple platforms, including CPU, GPU, Raspberry Pi, and FPGA, for standard image processing algorithms, FPGA emerged as the superior platform, with performance ten times higher than its counterparts. This research, centered around a mobile explorer robot, signifies the importance of selecting the appropriate hardware platform based on the computational demands of specific image processing tasks, thereby optimizing robotic navigation and obstacle avoidance capabilities. Hitchcock et al.[Hit+21] introduce PyTorchDIA in their study, a GPU-accelerated method for Differential Item Functioning analysis within the PyTorch framework. By leveraging GPU computing, the approach significantly reduces computation times, enhancing the efficiency of statistical analyses in educational testing. This development underscores the potential of GPU acceleration in the domain of large-scale data analysis, offering a scalable and efficient solution for educational researchers and practitioners. The study by Barreiros Jr. et al. [Bar+22] introduces a cost-aware data partitioning strategy for microscopy image analysis on CPU-GPU systems, addressing the challenge of efficiently processing irregularly distributed data. By achieving significant improvements in computational efficiency, the study highlights the potential of integrating advanced data partitioning techniques with GPU acceleration to enhance the performance of image analysis tasks. This approach opens new avenues for the development of high-throughput imaging technologies, facilitating faster and more accurate scientific discoveries. Won Jo et al.[JG22] present a novel GPU-accelerated approach for Riemannian metric optimization crucial for surface analysis in medical images. The study demonstrates how GPU computing can improve the speed and efficiency of geometric operations, critical for accurate image analysis. This methodological advancement highlights the expanding role of GPU acceleration in medical imaging, offering new perspectives on disease diagnosis and monitoring through enhanced image processing capabilities. In summary, the shift to GPU-accelerated computation has proven to be a cornerstone in image analysis, significantly enhancing precision and efficiency, and setting a new benchmark for future advancements in the field.

3 Experiments

Since we have established theoretical suppositions that GPUs are better than CPUs performing operations which are heavily used in Image Analysis, Now, we shall conduct experiments that involve extensive datasets including images and videos. The main objective of all experiments will be to evaluate the Computational Performance Characterization by conducting algorithms that can take advantage of GPU acceleration and comparing them with CPU performance. Two algorithms are evaluated in this experiment:

3.1 U-Net based Brain tumor segmentation from 2D MRI scans

In the field of medical imaging, efficient and high-accuracy segmentation results is particularly important in scenarios where precise delineation of lesions or abnormalities is critical for proper diagnosis and treatment planning. One algorithm that has gained widespread recognition for its effectiveness in medical image segmentation is the U-Net[RFB15], a fully convolutional encoder-decoder network architecture designed specifically for this task. In the context of brain tumor segmentation from MRI scans, the U-Net's performance is invaluable, as it can accurately identify and segment tumor regions, enabling radiologists and clinicians to make informed decisions about diagnosis and treatment strategies. The U-Net's architecture is characterized by its unique structure, which consists of a contracting path (encoder) that captures contextual information and an expanding path (decoder) that enables precise localization. Additionally, skip connections between the encoder and decoder paths allow the network to preserve fine-grained details and merge semantic features from high-resolution features, resulting in accurate segmentation results. U-Net's powerful architecture faces challenges with large medical datasets and real-time applications due to its computational intensity. This is where GPU acceleration offers a transformative solution, leveraging parallel processing in modern GPUs. Since the U-Net is a fully convolutional network, its layered processing benefits significantly from GPU acceleration. The convolution operations, which are at the core of the network's functionality, can be parallelized across the thousands of cores available in modern GPUs. This parallelization allows for simultaneous processing of multiple image regions, skip connections and the concatenation operations involved in merging feature maps from different levels of the network, significantly reducing the overall computation time compared to traditional CPU-based approaches. By accelerating the U-Net algorithm on GPUs, radiologists and clinicians can analyze large volumes of medical imaging data in a timely manner, enabling faster diagnosis and treatment planning. Additionally, the high accuracy of the U-Net's segmentation results ensures that critical details are not overlooked, potentially preventing misdiagnosis or suboptimal treatment strategies.

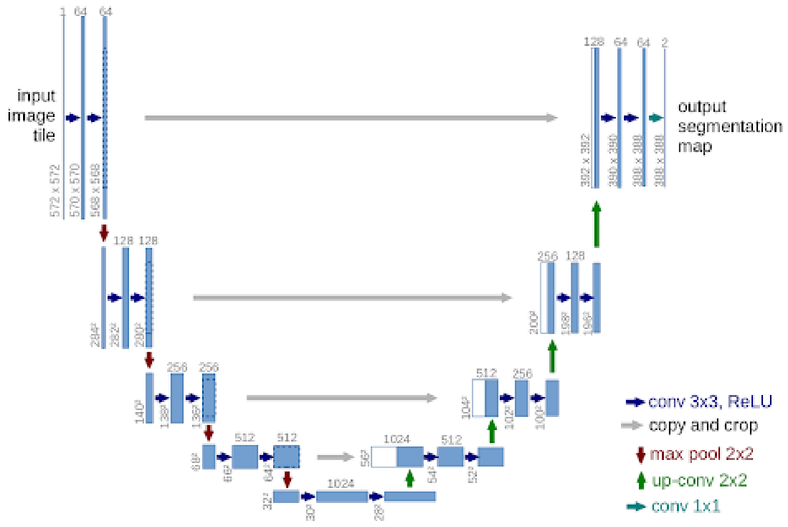


Figure 9: U-Net architecture [RFB15]

3.2 Real time video analysis for object detection and tracking using YOLOv7

The rise of cameras and video sources across domains has heightened the need for rapid and accurate data processing, making real-time video analysis essential for swift decision-making in today's data-driven world. Applications such as autonomous vehicles, security monitoring, and retail analytics require the ability to detect and track objects in real-time, enabling timely actions and informed decision-making. YOLOv7 [WBL23] addresses this need by providing a highly efficient and accurate solution for real-time video analysis. YOLOv7 (You Only Look Once) is a state-of-the-art model that is designed to perform object detection and tracking in real-time on video streams, making it an invaluable tool for a wide range of applications. Deep learning models, such as those used in object detection and tracking, are computationally intensive and require a significant amount of processing power. GPUs, with their thousands of parallel cores, are well-suited for these types of tasks, allowing for faster processing times and higher throughput compared to traditional CPUs. YOLOv7 is designed to take advantage of GPU acceleration, enabling it to process video frames at extremely high rates. By offloading the computationally intensive tasks to the GPU, YOLOv7 can process multiple frames simultaneously, ensuring smooth and consistent real-time performance. This GPU acceleration capability is crucial for applications that demand real-time analysis, such as autonomous driving, where split-second decisions can mean the difference between safety and potential accidents. On a modern CPU, processing a single frame for object detection and tracking using YOLOv7 could take several hundred milliseconds, depending on the complexity of the scene and the CPU's capabilities, resulting in significant lag for a video stream at 30 frames per second (FPS); however, GPU acceleration can reduce processing time to just a few milliseconds, enabling real-time performance at high frame rates, with YOLOv7 processing video streams at over 100 FPS on a modern GPU like the NVIDIA RTX 3080, ensuring smooth and responsive real-time analysis. YOLOv7 employs a single neural network to perform object detection and tracking simultaneously, eliminating the need for separate models for each task. The YOLO algorithm takes an input image and divides it into a grid of cells. The image is passed through a convolutional neural network (CNN) backbone, such as Darknet or ResNet, to extract features at multiple scales. For each grid cell, YOLO predicts bounding boxes, confidence scores, and class probabilities. Each bounding box is represented by its center coordinates, width, height, and confidence score indicating the likelihood that the bounding box contains an object. Class probabilities represent the likelihood of each detected object belonging to different predefined classes. After predictions are made for all grid cells, YOLO applies post-processing techniques such as non-maximum suppression (NMS) to remove redundant bounding boxes and keep only the most confident detections. The final output consists of bounding boxes along with their associated class labels and confidence scores, indicating the presence of objects and their types in the image. YOLOv7 uses a novel architecture called CSPNet (Cross-Stage Partial Network), which combines features from different layers of the network to enhance the model's accuracy and performance. One of the key innovations in YOLOv7 is the introduction of new techniques, such as Auxiliary Classifiers, to improve the model's ability to detect small objects, a common challenge in object detection tasks. Additionally, YOLOv7 incorporates advanced techniques like Mosaic data augmentation and Self-Adversarial Training (SAT) to enhance the model's robustness and generalization capabilities. In the context of real-time video analysis, YOLOv7 processes each frame of the video stream independently, detecting and tracking objects within each frame.

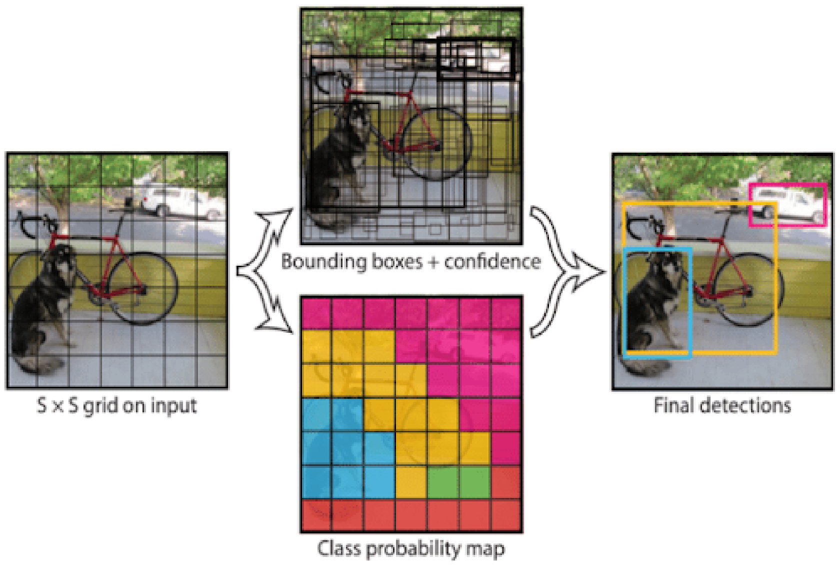


Figure 10: Working of YOLO model

4 Tools and Technologies

In the programming environment, Python serves as the primary language, supported by essential libraries such as PyTorch, NumPy, Subprocess, SciPy, and Scikit-learn, alongside specialized frameworks like segmentation-pytorch-models and YOLOv7. The processing units environment encompasses GPUs like the NVIDIA Tesla P100 on Kaggle and NVIDIA Tesla T4 on both Colab and Kaggle, along with CPUs including the Intel Core i9-10900X CPU running at 3.70GHz and the Intel Xeon CPU operating at 2.20GHz on Colab. Key concepts involve Convolutional Neural Networks (CNNs), Single Shot Detectors (SSDs), Encoder-Decoder segmentation, Floating Point Operations (FLOPs), and Parallelization techniques, all integral to the implementation and optimization of the above mentioned tasks. The data used for the task of brain tumor segmentation contains brain MR images together with manual FLAIR abnormality segmentation masks. These images were obtained from The Cancer Imaging Archive (TCIA). They correspond to 110 patients included in The Cancer Genome Atlas (TCGA) lower-grade glioma collection. For the task of object detection and tracking, sample videos of 10-15 seconds, including pedestrians and cars in open areas, were taken from the internet on which inference was carried out using the YOLOv7 model.

5 Results and Discussion

This section explores the results obtained from the experiments. The results will include task specific metrics such as qualitative results, loss curves and IOU scores, as well as computational performance metrics such as execution time, throughput, processing FPS curves.

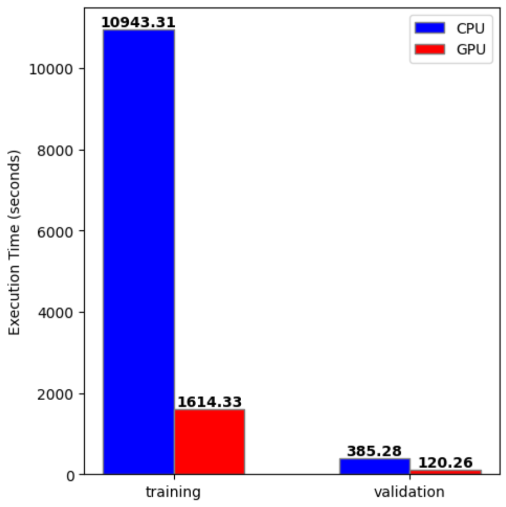


Figure 11: Training and validation execution times by processing unit

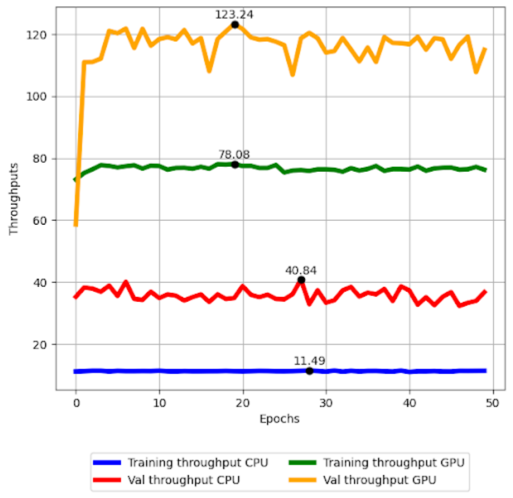


Figure 12: Train and validation throughputs of CPU and GPU

Figure 11 visually confirms the expected benefit of GPU acceleration for U-Net brain tumor segmentation, with the GPU execution time being approximately 6.8 times faster than the CPU. This finding aligns with the strengths of GPUs in parallel processing for computationally intensive tasks. However, the validation execution times might not show a substantial difference due to the potentially less demanding nature of validation calculations, data transfer overheads, or a small validation dataset size. As per Figure 12, the GPU exhibits considerably higher throughput compared to the CPU for both training and validation stages. The higher throughput of the GPU translates to processing image data at a faster rate, leading to potentially faster training times and improved efficiency during task execution. Now for the task-specific metrics, the GPU's training loss curve starts converging from a very low value compared to the CPU, as visible in Figure 13 and 14. This is also evident in the curves for IOU score in Figure 15 and 16, where the GPU starts achieving higher values for these metrics from the beginning, and gradually both the CPU and GPU converge to the same values for all metrics. This suggests the model trained on the GPU begins learning effectively much quicker during the initial epochs. This aligns with the expected advantages of GPUs in parallel processing, allowing them to grasp patterns in the training data faster.

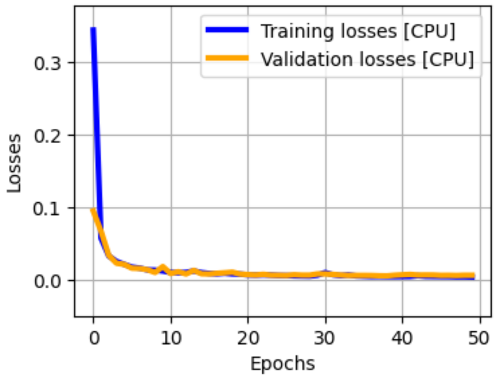


Figure 13: Training and Validation losses for CPU

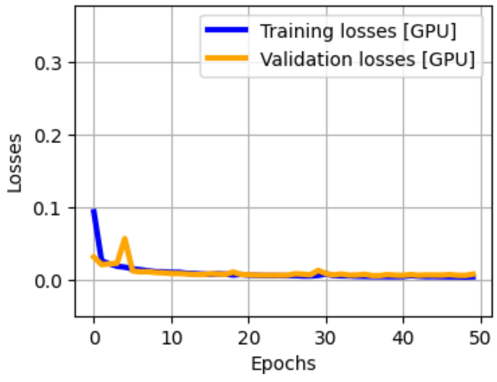


Figure 14: Training and Validation losses for GPU

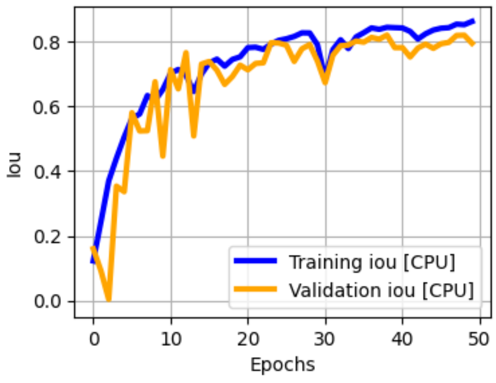


Figure 15: Training and validation IOUs for CPU

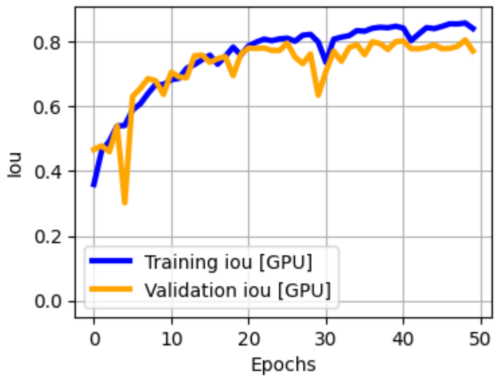


Figure 16: Training and validation IOUs for GPU

For the task of object detection with YOLOv7, it is again evident from Figure 17 that the CPU execution time is several times greater than the GPU execution time. This translates to a substantial speedup factor in favor of the GPU for this task. processing individual image regions and tracking objects likely benefit significantly from the parallel processing architecture of GPUs.

As visible in Figure 18, The GPU processing FPS is several times greater than the CPU processing FPS. The higher processing FPS on the GPU translates to a faster rate of processing video frames, which is crucial for real-time object detection tasks. This improved frame rate allows for smoother detection and tracking of people and objects within the defined Region of Interest. However, the GPU's FPS curve shows noticeable jaggedness, likely due to variations in scene complexity and data transfer overhead between CPU and GPU memories. These fluctuations can disrupt the smooth processing pipeline and lead to inconsistencies in frame processing time.

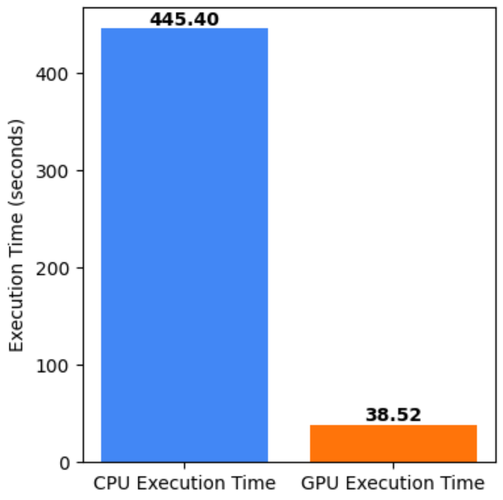


Figure 17: CPU and GPU execution times for YOLOv7

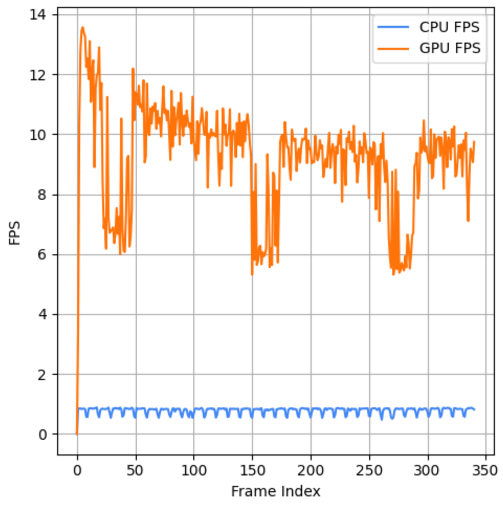


Figure 18: CPU and GPU processing FPS for YOLOv7

The task-specific qualitative results from both the tasks performed on the GPU have also yielded accurate results. Analyzing the qualitative results from the U-Net model's brain tumor segmentation, as visible in Figure 19, it's evident that the model achieves varying degrees of alignment with the ground truth. In some instances, the tumor boundaries are predicted with remarkable accuracy, capturing the curves and locality of the tumor with a high degree of precision, although with occasional slight jaggedness suggesting potential overfitting. However, in other cases, the model exhibits minor deviations, such as underfitting that leads to less detailed perimeters, or over-segmentation where false positives appear due to the model's sensitivity to textures resembling tumor tissues. The model also shows a tendency to miss subtleties in the tumor's perimeter in certain cases, and in one instance, introduces an internal pore not present in the ground truth, likely due to over-sensitivity to intratumoral variations. Overall, the predictions demonstrate the GPU-accelerated model's capability to learn complex patterns effectively, yet they also highlight areas for further refinement, particularly in enhancing boundary smoothness and reducing false positives for irregularly shaped tumors.

The resultant video frame, as shown in Figure 20, from the inference process offers a qualitative assessment of YOLOv7's object detection and SORT's tracking capabilities on a GPU. The system has successfully detected and tracked multiple individuals represented by rectangular bounding boxes with unique ID numbers assigned to each person. The output demonstrates the capabilities of modern object detection and tracking algorithms in handling complex scenarios involving multiple moving objects. It highlights the potential applications of such systems in crowd monitoring, surveillance, or pedestrian analytics which frequently involve densely packed artifacts within images and videos. To do such analytics in real-time, it is necessary to employ GPU acceleration for higher efficiency.

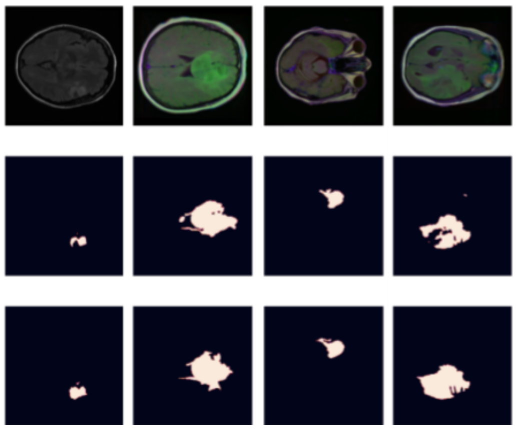


Figure 19: Visual results from U-Net brain tumor segmentation task

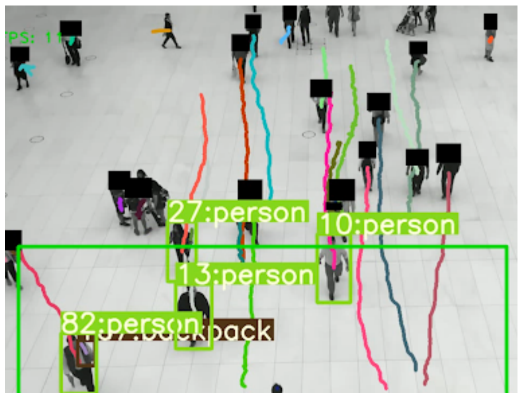


Figure 20: Visual results from YOLOv7 object detection and tracking task

6 Conclusion

Our experiments explored the computational performance of GPUs for image analysis tasks. The results clearly demonstrate the significant advantages of GPUs compared to CPUs in terms of speed and efficiency. This was evident in both brain tumor segmentation and real-time object detection with tracking using YOLOv7 and SORT algorithms. Faster execution times, higher throughputs, and overall improved performance were achieved with GPU acceleration. This highlights the importance of GPUs in processing the vast amount of image data available. They offer a powerful and efficient alternative to manual analysis, particularly for tasks that involve complex calculations. Looking forward, advancements in deep learning algorithms like Transformers are expected to further enhance the capabilities of GPU-accelerated image analysis, leading to wider applications and improved accuracy. Additionally, the development of energy-efficient GPUs aligns with the push for greener AI solutions, addressing current limitations in energy consumption. Furthermore, neuromorphic computing, inspired by the human brain's efficiency, holds promise for a future revolution in computing power and image analysis capabilities. In conclusion, this exploration of GPU-accelerated image analysis showcases the significant progress made in this field and paves the way for exciting future developments. Continuous improvements in algorithms and hardware will undoubtedly unlock new possibilities in image analysis, allowing us to better understand the visual world around us.

References

[Arn+98] A. S. Arnold et al. "A Simple Extended-Cavity Diode Laser". In: *Review of Scientific Instruments* 69.3 (Mar. 1998), pp. 1236–1239. URL: http://link.aip.org/link/?RSI/69/1236/1.

[Bar+22] Willian Barreiros et al. "Efficient microscopy image analysis on CPU-GPU systems with cost-aware irregular data partitioning". In: *Journal of Parallel and Distributed Computing* 164 (2022), pp. 40–54. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2022.02.004. URL: https://www.sciencedirect.com/science/article/pii/S0743731522000466.

[CS20] Yu-Wei Chang and Tony Sheu. "GPU acceleration of a patient-specific airway image segmentation and its assessment". In: (Dec. 2020).

[Hit+21] James A Hitchcock et al. "PyTorchDIA: a flexible, GPU-accelerated numerical approach to Difference Image Analysis". In: *Monthly Notices of the Royal Astronomical Society* 504.3 (2021), pp. 3561–3579. DOI: 10.1093/mnras/stab1114.

[JG22] Jeong Won Jo and Jin Kyu Gahm. "G-RMOS: GPU-accelerated Riemannian Metric Optimization on Surfaces". In: *Computers in Biology and Medicine* 150 (2022), p. 106167. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.106167. URL: https://www.sciencedirect.com/science/article/pii/S0010482522008757.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4.

[Seb21] Stefaniga Sebastian-Aurelian. "Methods of acceleration for feature extractions in medical imaging using GPU processing". In: *2021 23rd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. 2021, pp. 234–241. DOI: 10.1109/SYNASC54541.2021.00047.

[WBL23] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 7464–7475. DOI: 10.1109/CVPR52729.2023.00721.