Cartoonizer: report
High Performance Computing Project

Emilio Garzia [mat. 0124/314]
Luigi Marino [mat. 0124/320]

2025

Contents

Chapter 0

Introduction

Cartoonizer aims primarily to create a graphic effect in "cartoon" style applied to digital images. This effect is achieved by reducing the number of colors in the image using the **k-means clustering** method. This algorithm groups the colors of the image into a predefined number of clusters, allowing each pixel to be represented by the color of the centroid of the cluster it belongs to. The result is a simplified image, with sharper color transitions and a visually appealing appearance similar to an animated drawing. The implementation of the Cartoonizer was developed using **CUDA** (*Compute Unified Device Architecture*) technology, designed to leverage the parallel computing power of **GPUs** (*Graphics Processing Units*), with the primary goal of significantly improving performance compared to a sequential version of the algorithm, enabling the processing of large images in reduced time. A comprehensive evaluation will be presented in the report, showcasing the performance of the algorithm across various **NVIDIA** devices. The analysis will include comparisons based on different parameters, such as thread numbers and device configurations, providing detailed insights into the scalability and optimization of the algorithm. This section will delve into how varying these factors influences both execution time and overall performance. The results will be meticulously detailed, illustrating the impact of parallelization and hardware-specific optimizations in real-world scenarios. The idea for this project stems from two foundational research papers that inspired its development. The first paper discusses various methods to generate cartoonized painterly effects on grayscale and colored images. It introduces the concept of vector quantization to achieve the painterly effect and compares algorithms such as LBG, KPE, and KMCG based on their processing time and visual results. These methods have applications in fields such as movie-to-comic conversions and digital art software [1]. The second paper focuses specifically on the use of k-means for color quantization. While k-means has historically been viewed as computationally expensive and sensitive to initialization, the paper demonstrates that with efficient implementations and appropriate initialization strategies, k-means can serve as a highly effective color quantization method. The experiments conducted on diverse images highlight the algorithm's per-

formance and its potential in image processing [2]. In the following chapters and sections, we will examine in detail the approach adopted, starting from the description of the problem and the techniques used, moving through the implementation in CUDA, and finally evaluating the performance achieved. The results highlight the benefits of parallelization, demonstrating how GPU computing can be utilized for creative and high-impact visual applications.

Chapter 1

Theoretical background

1.1 K-Means algorithm

K-means clustering is a fundamental algorithm in unsupervised machine learning, widely employed for partitioning a dataset into a predefined number of clusters, K, based on feature similarity. The algorithm is iterative and aims to minimize the variance within each cluster, leading to compact and well-separated groups of data points. The K-means algorithm operates by alternating between assignment and update steps until convergence. The process can be described as follows:

1. **Initialization**: Select K initial centroids randomly from the dataset.

2. **Assignment Step**: Assign each data point to the nearest centroid using a distance metric, commonly the Euclidean distance.

3. **Update Step**: Calculate the new centroids by taking the mean of all data points assigned to each cluster.

4. **Convergence**: Repeat the assignment and update steps until the centroids stabilize, or a maximum number of iterations is reached.

The objective of K-means is to minimize the sum of squared distances between each data point and the centroid of its assigned cluster. Mathematically, this is represented as:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbf{1}(y_i = k) \|\mathbf{x_i} - \mu_k\|^2$$

Where:

- J: The within-cluster sum of squared distances (WCSSD).

- $\mathbf{x_i}$: The i-th data point.

- μ_k: The centroid of the k-th cluster.

- $\mathbf{1}(y_i = k)$: An indicator function that equals 1 if data point $\mathbf{x_i}$ is assigned to cluster k, and 0 otherwise.

The algorithm alternates between minimizing J by assigning data points to their closest centroids and recalculating centroids based on these assignments.
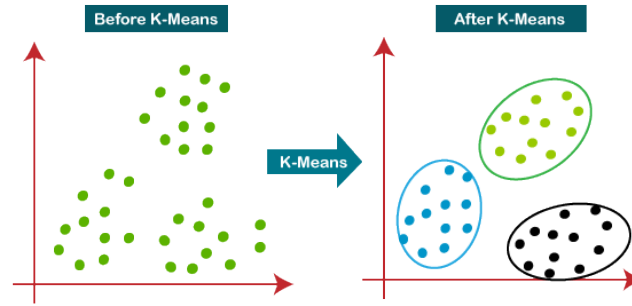

Figure 1.1: Set of data before and after the application of the K-Means algorithm

1.1.1   Advantages and Limitations

K-means is valued for its simplicity, efficiency, and scalability:

- **Simplicity**: Easy to implement and interpret.

- **Efficiency**: Computationally efficient for moderate values of K and large datasets.

- **Versatility**: Applicable to a wide range of domains, including image compression and customer segmentation.

Despite its strengths, K-means has several limitations:

- **Choice of** K: Requires prior knowledge of the number of clusters.

- **Initialization Sensitivity**: Poor initialization can lead to suboptimal clustering.

- **Cluster Shape Assumption**: Assumes clusters are spherical and of equal size, which may not hold for all datasets.

- **Outliers**: Sensitive to outliers, which can distort cluster centroids.

1.1.2   Applications

K-means clustering has numerous applications, including:

- **Image Processing**: Color quantization and compression by reducing the number of colors in an image.

- **Market Segmentation**: Grouping customers based on purchasing behavior for targeted marketing.

- **Data Summarization**: Simplifying datasets by grouping similar data points.

- **Anomaly Detection**: Identifying outliers as data points that do not fit into any cluster well.

1.1.3   Conclusion

K-means clustering remains a foundational algorithm in machine learning due to its simplicity and effectiveness in solving clustering problems. While it has limitations, various extensions and initialization strategies, such as K-means++, have been developed to address these issues and improve the algorithm's robustness.

1.2   CUDA

Cuda

1.3   Color quantization

Color quantization

Bibliography

[1] Archana B. Patankar, Purnima A. Kubde, and Ankita Karia. "Image cartoonization methods". In: *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*. 2016, pp. 1–7. DOI: 10.1109/ICCUBEA.2016.7860045.

[2] M. Emre Celebi. "Improving the performance of k-means for color quantization". In: *Image and Vision Computing* 29.4 (2011), pp. 260–271. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2010.10.002. URL: https://www.sciencedirect.com/science/article/pii/S0262885610001411.