**Programming Assignment #2**

**Introduction**

With the rapid increase of the information that is handed nowadays, it is not uncommon to see huge datasets for all different types of applications, for instance, speech recognition, market studies, etc. Clustering techniques come as an aid to obtain the most information out of this extensive gathered data, by separating data into meaningful groups that can be analyzed more easily. The purpose of the following assignment is to experiment with different clustering algorithms, showing the cases where they perform the best and the worst; all while also being compared to each other with different kinds of metrics.

**Objectives**

The main objectives of this assignment are the following:

* Implement some of the most known clustering algorithms discussed in the lectures.
* Experiment with different parameters of the algorithms to see different results.
* Compare the results of own implementations with those made by popular clustering libraries.

**Review of the methods used**

Formally speaking, clustering algorithms are the ones that separate every instance of the data into groups called clusters, in which every instance is as similar as possible from other instances in their cluster, is as different as possible from instances in other clusters, and the final separation of clusters in based on a clear and meaningful measurement criterion.

* Datasets

To test the algorithms’ performance on a variety of scenarios, very different datasets were used during the experiments shown in the next section. The datasets used can be seen in

Scatter chart

Description automatically generated

Figure 1 From left to right, top to bottom: (a)Noisy circles (b)Noisy moons (c)Blobs with varied variances (d)Anisotropicly distributed blobs (e)Equally sized blobs (f)Unclustered data

* Metrics
  + Internal
    - Davies Bouldin Index (DB Index)
    - Silhouette Index
    - calinski\_harabasz\_score
  + External

**Explanation of the experiments done**

A picture containing text

Description automatically generated

**Discussions**

The results obtained from the Canny edge detector are good, but the algorithm does not make it simple to adjust its parameters to fit each image. For this reason, the usage of an improved algorithm is highly encouraged, to save time in the fine tuning of the algorithm and get great results.

It is also interesting to note the effect of the Laplacian sharpening filter, that even when the original image seems almost unaffected to the human eye, the Canny algorithm still is affected a lot by the changes made to the image, even around areas that do not seem to have any edge on the original image. Like the sky on Figure 9.

**Main References**

* Xu, D., Tian, Y. A Comprehensive Survey of Clustering Algorithms. Ann. Data. Sci. 2, 165–193 (2015). <https://doi.org/10.1007/s40745-015-0040-1>
* Yingke Feng, Jinmin Zhang, and Siming Wang , "A new edge detection algorithm based on Canny idea", AIP Conference Proceedings 1890, 040011 (2017) https://doi.org/10.1063/1.5005213
* Ranita Biswas, & Jaya Sil (2012). An Improved Canny Edge Detection Algorithm Based on Type-2 Fuzzy Sets. Procedia Technology, 4, 820-824.
* W. Rong, Z. Li, W. Zhang and L. Sun, "An improved Canny edge detection algorithm," 2014 IEEE International Conference on Mechatronics and Automation, 2014, pp. 577-582, doi: 10.1109/ICMA.2014.6885761.
* Gonzalez, R. C. (2018). *Digital Image Processing* (pp. 716-737). Pearson.
* Sofiane Sahir, Canny Edge Detection Step by Step in Python — Computer Vision, 2019. https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123. (Accessed December 2021).

**Code**