

# Efficient Extraction and Matching of LISF Features in GPU

Leonardo Chang<sup>1,2</sup>, Miguel Arias-Estrada<sup>1</sup>, L. Enrique Sucar<sup>1</sup> and José Hernández-Palancar<sup>2</sup>

## 1. ABSTRACT

- LISF (Invariant Local Shape Features)[1] is an invariant local shape features descriptor which can be used to obtain a discriminative and compact representation of an object.
- For high occlusion levels (up to 60%) LISF method outperformed other popular shape description methods, with about 20% higher bull's eye score and 10% higher accuracy in classification.
- In this work, we present a massively parallel implementation in GPU of the two most time expensive parts of LISF algorithm, i.e. feature extraction and feature matching steps.
- A 32x speedup is achieved compared to the CPU implementation when extracting features in a 1000 points object contour.
- A 34x speedup is achieved compared to the CPU implementation when matching 290 vs 290 LISF features.
- In feature matching, GPU implementation shows linear scaling while increasing number of features to match, in contrast to CPU implementation that shows an exponential growth.

## 2. THE INVARIANT LOCAL SHAPE FEATURES

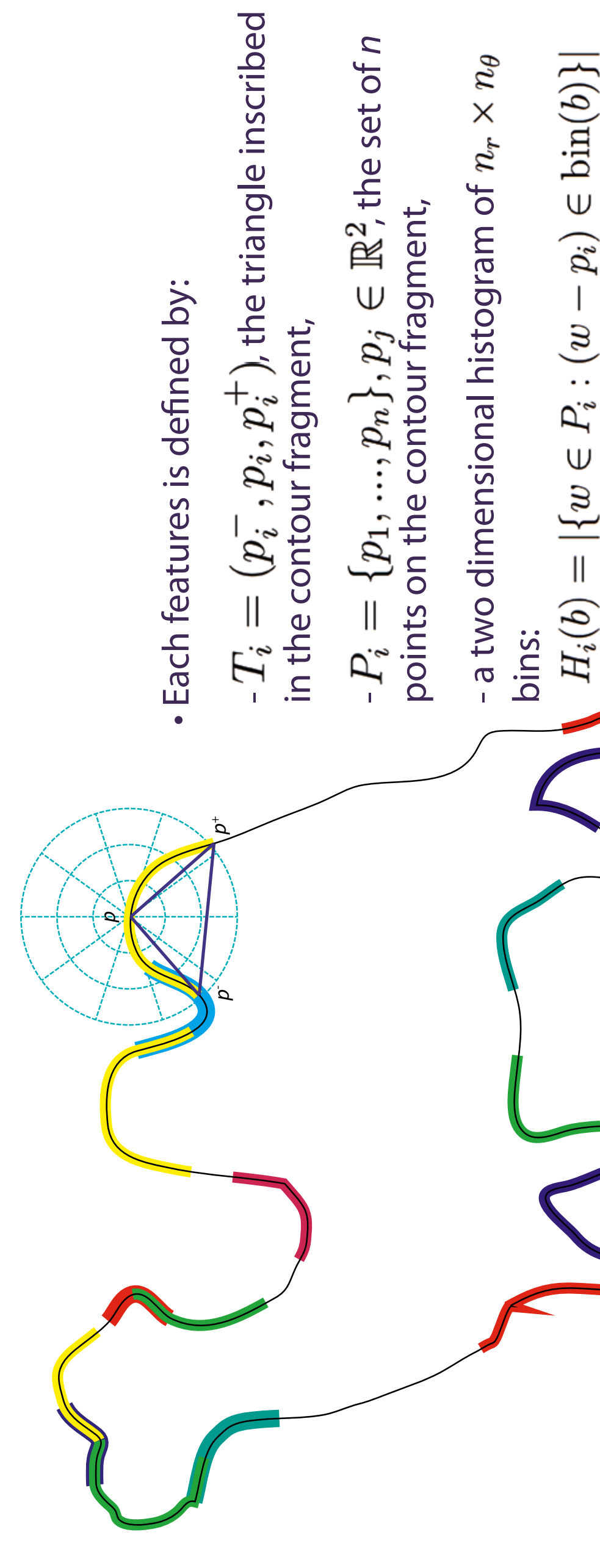
- In LISF [1], boundary and internal edge fragments are used to represent the shape of an object.
- When using contour fragments, we do not have to deal with the inherent limitations of using global features.
- In addition, there is not restriction about only dealing with closed contours or silhouettes, the features are extracted from both the contour and the internal edges of the object.
- LISF extraction and description methods are invariant to rotation, translation, scale and present certain robustness to partial occlusion.

### 2.1. FEATURES EXTRACTION AND DESCRIPTION

- High curvature contour fragments are used as shape features.
- Feature extraction is based on Chetverikov method [5].
- Candidate contour fragments are those where a triangle  $(p_i^-, p_i, p_i^+)$  can be inscribed such that:

$$\begin{aligned} d_{min} &\leq \|p^- - p^+\| \leq d_{max}, \\ d_{min} &\leq \|p^- - p^-\| \leq d_{max}, \\ \alpha_{min} &\leq \alpha \leq \alpha_{max}. \end{aligned}$$

- Several triangles can be found over the same point or over adjacent points. The triangle with the highest curvature in a neighborhood is selected.



<sup>1</sup> Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)  
Luis Enrique Erro No.1, C.P. 72840, Tonantzintla, Puebla, Mexico.  
{lchang, ariasmo, esucar}@ccc.inaoep.mx

<sup>2</sup> Advanced Technologies Application Centre (CENATAV)  
7<sup>th</sup>A No. 21406, Playa, C.P. 12200, La Habana, Cuba.  
{lchang, jpalancar}@cenatav.co.cu

## 3. FEATURE MATCHING

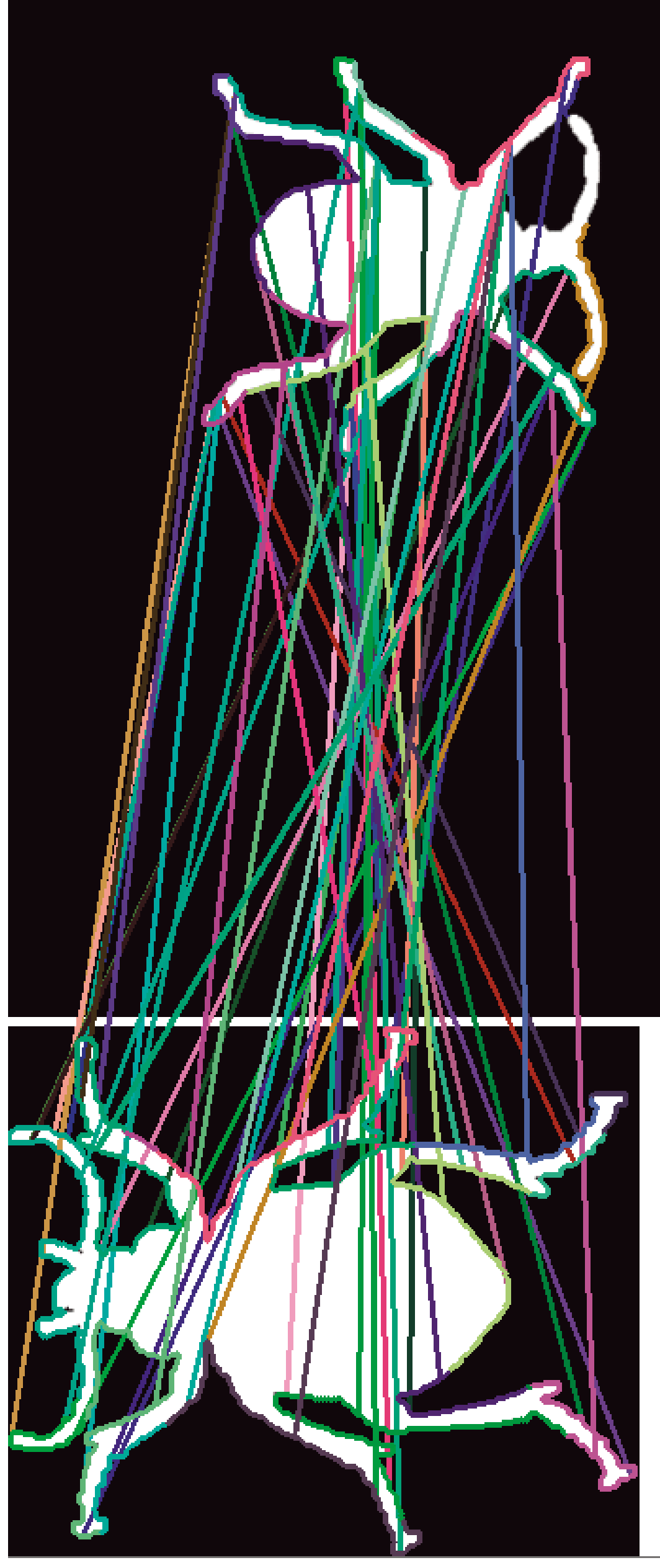
- Feature matching takes into account the structure and spatial organization of the features.
- Matches are validated by rejecting casual or wrong matches.

### 3.1. FINDING CANDIDATE MATCHES

- For each feature in the query, its  $K$  nearest neighbors in the database image are found by comparing their descriptors.
- In this work we use chi-squared distance to compare histograms.  $\chi^2 = \sum_i \frac{(m_i - n_i)^2}{m_i + n_i}$
- To provide the method with rotation invariance the feature descriptors are normalized in terms of orientation, (i.e., all features are set to orientation zero).

### 3.2. REJECTING CASUAL MATCHES

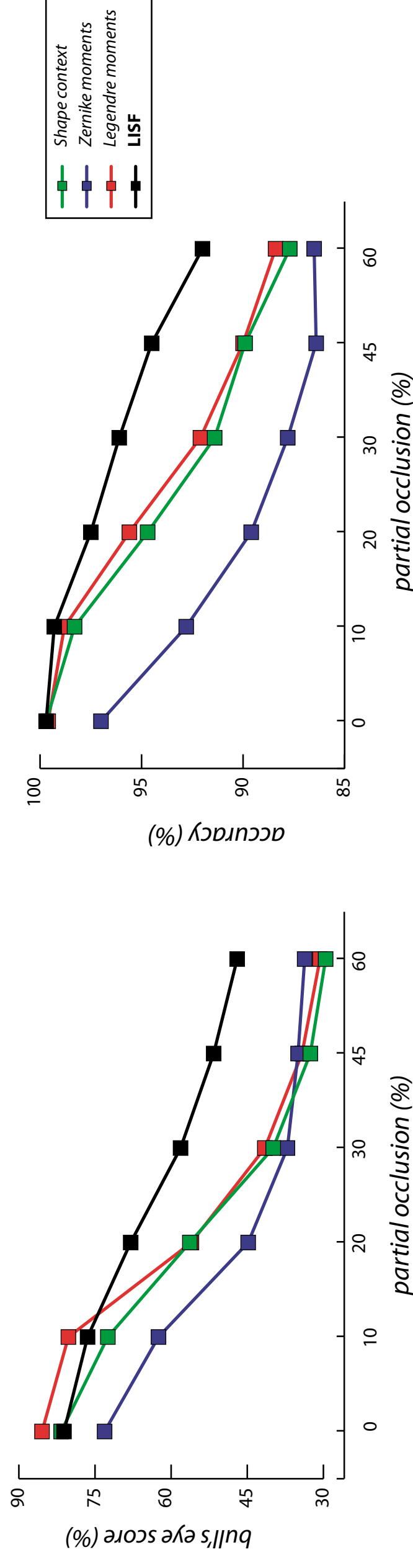
- For each candidate match, features on both images are aligned according to its corresponding scale, rotation and translation.
- For each aligned feature in the query image, its nearest neighbor in the database image is found.
- This nearest neighbor votes for a center position in the candidate image.
- If the candidate match is a correct one, most of the center votes should be concentrated around the object centroid.
- This idea is what allows us to accept or reject a candidate match, using a measure of dispersion.



Matches between local shape descriptors in two images. It can be seen how these matches were found even in presence of rotation, scale and translation changes.

## 4. LISF EXPERIMENTAL RESULTS

- MPEG-7 CE-Shape-1 dataset.
- 10 classes, 20 images per class.
- Artificially introduced different levels of partial occlusion (0%, 10%, 20%, 30%, 45%, 60%).
- 2 experiments, shape retrieval and classification.
- Comparison with Zernike moments [2], Legendre moments [3] and shape context [4].



## REFERENCES

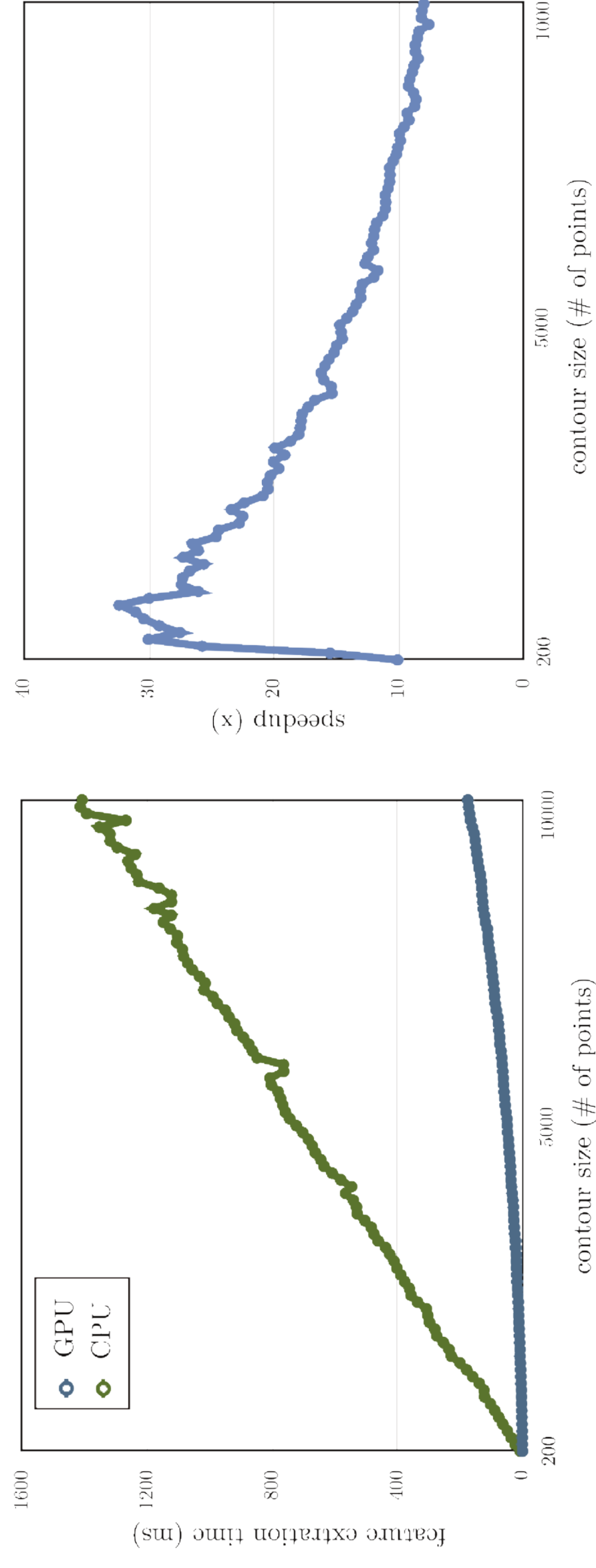
- [1] L. Chang, M. Arias-Estrada, L. E. Sucar and J. Hernández-Palancar. LISF: An Invariant Local Shape Features Descriptor Robust to Occlusion. *International Conference on Pattern Recognition Applications and Methods, ICPRAM 2014*.
- [2] A. Khotanzad and Y. H. Hong. Rotation invariant pattern recognition using zernike moments. *9th International Conference on Pattern Recognition*, 1988, pages 326–328 vol.1, 1988.

- [3] C.-W. Chong, P. Raveendran, and R. Mukundan. Translation and scale invariants of legendre moments. *Pattern Recognition (PR)*, 37(1):119–129, 2004.
- [4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.
- [5] D. Chetverikov. A Simple and Efficient Algorithm for Detection of High Curvature Points in Planar Curves. *Proceedings of the 23rd Workshop of the Austrian Pattern Recognition Group*, pages 746–753, 2003.

## 5. IMPLEMENTATION USING CUDA

### FEATURE EXTRACTION - IMPLEMENTATION KEY ISSUES

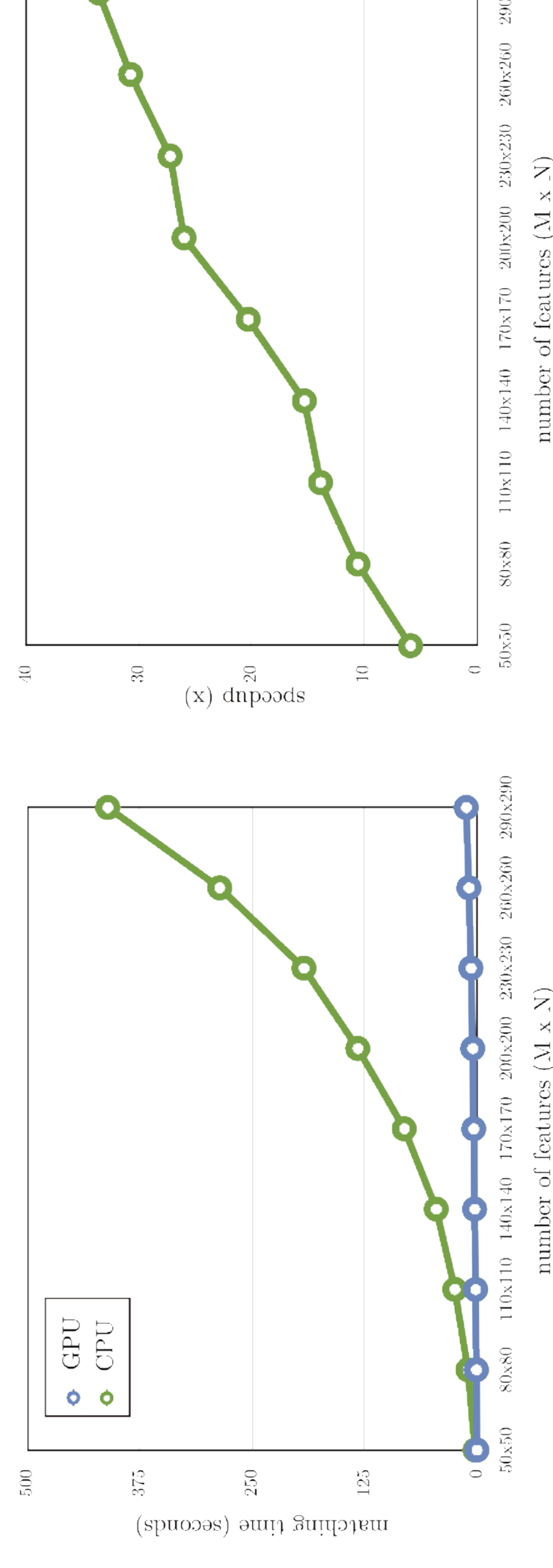
- For each point  $p_i$  in the contour, up to  $P$  triangles are evaluated, where  $P$  is the contour size.
- Candidate triangles (triangles that fulfill constraints in 2.1) of point  $p_i$  are computed in a block.
- Constrains of each candidate triangle are evaluated in a thread.
- Each block selects the triangle of highest curvature among candidate triangles of corresponding point  $p_i$ .
- Computed on a NVIDIA GeForce GT 610.
- Compared to single-threaded Intel CPU Processor at 3.4 GHz.
- Up to 32x speedup, 16x average speedup.



### FEATURE MATCHING - IMPLEMENTATION KEY ISSUES

- Finding candidate matches involves  $M \times N$  chi-squared comparisons of feature descriptors.
- Also, rejecting casual matches needs  $M \times N$  chi-squared comparisons after alignment.
- Inputs:  $m$ , descriptors of query image features ( $M$  histograms of 64 bins concatenated).
- $n$ , descriptors of database image features ( $N$  histograms of 64 bins concatenated).
- Each value in  $m$  is used  $N$  times.
- In order to increase data reutilization and decrease global memory accesses,  $m$  and  $n$  are tiled into the shared memory.
- For values of  $M$  and  $N$ , such that the features and comparison results do not fit in the device memory, the matrix could be partitioned and the kernel launched several times.

- Computed on a NVIDIA GeForce GTX 480.
- So, grid size is  $\lceil (M+7)/8 \rceil$ ,  $\lceil (N+7)/8 \rceil$  and the number of threads per block is  $8 \times 64$ .



- GPU implementation linear scaling vs. CPU exponential scaling.
- A 34x speedup is achieved compared to the CPU implementation when matching 290 vs 290 LISF features.

## ACKNOWLEDGEMENTS

- This project was supported in part by CONACYT grant No. 103878. L. Chang was supported in part by CONACYT scholarship No. 240251.