Applying Petri Nets to Approximation of the Euclidean Distance with the Example of SIFT

Jan Henrik Röwekamp and Michael Haustermann

University of Hamburg, MIN Faculty, Department of Informatics, Theoretical Computer Science Group and Cognitive Systems Group, Hamburg, Germany
http://www.informatik.uni-hamburg.de/TGI/
http://kogs-www.informatik.uni-hamburg.de/

Abstract. SIFT (Scale Invariant Feature Transform) is a complex image processing procedure for matching objects or patterns in images. Involving the computation of Euclidean distances in high dimensional space of many pairs of points and matching them against a threshold, this work proposes a speedup for the procedure utilizing colored Petri nets. Being sped up more pairs can be evaluated within reasonable time leading to better overall results.

Keywords: SIFT, object recognition, image processing, Euclidean distance, performance evaluation, binary squaring, complexity reduction, colored Petri nets

1 Introduction

Looking at the recent development of society towards omnipresence of computers coming in form of smart phones, handhelds, embedded systems and more the necessity of semi and fully automated processing of images to handle the amount of data and related requests produced by people becomes more and more severe. Efficient algorithms – in both running time as well as quality regard – cut down time spent by users to achieve their goals. SIFT [1](short for Scale Invariant Feature Transform) is a very powerful process for recognition of real world objects in an – from the queries point of view – unknown, heterogeneous image environment. At one critical point within the procedure it is necessary to compute the Euclidean distance between several pairs of high dimensional vectors and compare it against a certain threshold. Since this task is rather demanding in regards to computation time when done in the traditional way by just computing the distance, this work presents a method using Petri nets to lower the computation time for each distance calculation and comparison.

Object recognition as well as image processing in general is an extensive field of research with a lot of applications. Examples of applications benefiting from improved running times are **panorama photos** which a lot of modern mobile devices (smart phones) are able to take, **Optical character recognition (OCR)** in which an application for a live translator that automatically translates text seen in the real world is imaginable, **automated counting** of cars, people, coins, . . . using images and/or video streams and also **assignment of photos** in a sense where objects depicted on a photo known to an online database, but unknown to the user, are matched.

As several other image processing algorithms, the procedure of SIFT involves calculating the Euclidean distance in high dimensional spaces of a significant amount of points.

Binary Squaring

One of the basic ideas in this work is the bit-wise squaring $bsq(n)$ of binary numbers which will be called "binary squaring". Assume $n = [42]_{10} = [0000101010]_2 = 2^1 + 2^3 + 2^5$. The binary square would be: $bsq(n) = [1092]_{10} = [0100100100]_2 = 2^2 + 2^6 + 2^{10}$. Obviously this value does not coincide with the correct squared value of 42 (1792).

There will follow some propositions related to binary squaring, which (analytical) proofs are omitted due to space constraints. For the factor $R_f(n)$ between binary square and conventional square of an integer n it holds: $1 \leq R_f(n) < 3$. The average sum of binary squares of uniform distributed random integers between two powers of two differs from the sum of conventional squares by factor 1.944 with high probability. This also holds for values between 0 and a power of two. The computation of the Euclidean distance can be approximated using the sum of binary squares times 1.944. The major advantage of the binary square version of this computation is, that summands (on the binary level) can be interchanged *before* computing the squared value. Thus the computation of the Euclidean distance using binary squares can be rearranged to handle most significant bits first being able to recognize distances above the threshold very fast.

The Petri Net Model

The general idea is to model the computation of the binary square based Euclidean distance with a colored Petri net. By doing so some of the computations can be done by the net design itself, for example using edge weights / (computations) and also the distance calculation can be split up and parallelized utilizing Petri nets' inherent concurrency by using the concept of binary squaring. The net consists of three major parts: The bit-splitting component, the bit analyzer component and a decision component (which decides whether the points are below or above the threshold). The bit splitter utilizes the results above and splits the differences in each dimension of two points into its bits, subsequently placing all bits of same significance i over all dimensions into the same place of the net k_i. The analyzer component contains a place which again contains the (classic) squared value divided by 1.944 of the desired threshold. The analyzer component continuously removes marks from the pool for each mark removed from a pool k_i. As soon as the pool runs out of marks, but there are still marks in one of the k_i to be processed the decision component decides "false" otherwise it decides "true".

As there are only a very few pairs below the threshold and the majority above (depends on the algorithm, that requires to calculate the Euclidean distance, but in most cases this holds), most of the computations will end in only a few firings instead of d squares and additions when looking at a d-dimensional space.

References

1. David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–1157, Washington, DC, USA, 1999. IEEE Computer Society.