Spatially Invariant Vector Quantization

Egon Elbre
Hans Mäesalu

Abstract
Currently pattern matching requires considerable amount of knowledge about those methods and analysis to be usable and customizable. In this project we implemented SIVQ\textsuperscript{1} algorithm with a web interface that is easy to use and search parameters are easy to define. Also we provide an idea how to implement this algorithm with vectors.

SIVQ algorithm
SIVQ algorithm is a simple algorithm for matching a certain pattern in an image. This method works with search patterns that are in the same scale in the images. Those areas include pathology and aerial maps, as shown in the SIVQ paper.

The search pattern is defined as a ring vector (or multiple of them). This ring vector is a sampling of pixels around a point lying on a circle. This gives a search pattern that can be easily rotated and therefore can be used to determine similarity between parts. To determine the heatmap corresponding to the search pattern the same size ring vector is sampled around that pixel and compared with all possible rotations to the search pattern. This gives a difference value that gives the similarity around that point.

Throughout formulas we use two different ways of vector (and matrix) indexing \( p[i] = p_i \), they can be considered equivalent.

Formally, vector ring is a sampling in an image
\[
V(I,x,y)[i] = p_i = I[x_o + r \cos(\varphi^*i), y_o + r \sin(\varphi^*i)],
\]
where \( \varphi \) is the sampling stride and \( r \) is ring radius.
\([x_o,y_o]\) is the center of the ring vector.
The vector has \( n = |p| = \tau / \varphi \) elements.

\textsuperscript{1}“Spatially Invariant Vector Quantization: A pattern matching algorithm for multiple classes of image subject matter including pathology” [http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3049270/](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3049270/)
Also we define right circular shift matrix:
\[
R[n-1,0] = 1
\]
\[
R[1:n-1;1:n-1] = I_{n-1}, \text{ where } I_{n-1} \text{ is identity matrix of size } n-1.
\]

R is o in any other part

This matrix acts like reindexing with an offset:
\[
(p(R^o))[i] = p[i + o \mod n]
\]

R-matrix helps to write formulas concisely, although in implementation reindexing should be used instead.

SIVQ algorithm finds heatmap \( H \)
\[
H[x,y] = \min ||(V(I,x,y) - pR^o)||^2 / n, \text{ where } o \in \{0..n-1\},
\]
on an image that has normalized values.

In case of two ring vectors:
\[
H[x,y] = \min \left( ||(V(I,x,y) - pR^o)||^2 / ||p|| + ||(V(I,x,y) - qR^o)||^2 / ||q|| \right) / 2,
\]
assuming \(|q| \geq |p|\) then
\[
o = \text{floor} \left( |p| \cdot s / ||q|| \right)
\]
Similarly we can extend it to contain arbitrary number of ring vectors.

This method is simple, yet effective.

**Web interface**

In order to make using our application easier we developed web based interface using several new technologies defined in HTML5 specification. This also means that application can run in separate dedicated more powerful server and users can access it from remote computers.

Because our application is meant to be used by non-technical people that do not know much about image processing then we cannot assume that user knows what parameters give the best results for specific image in some certain situation. So to make it easy to find the correct parameters without reading long manuals or understanding how exactly the algorithm works we developed special parameters adjusting process, that helps user find these parameters. User simply has to select point of interest or previously saved vector and then application calculates several images for each parameter from which user has to select the best. When all parameters have been selected the result should be near optimal and additional tweaking can be done by manually adjusting the parameters.

Communication between server and client is done over WebSocket. This allows exchanging messages between server and client while the image is being processed, for example server can send status updates and client can stop the process. But at the time of writing this report, only latest Chrome, Firefox, Opera and Safari versions support WebSocket and by default it has been disabled in Firefox and Opera. But WebSocket is a standard defined in HTML5
specification and over the coming years will be implemented and enabled in all modern browsers.

**Parameters**
Vector location - determines where the ring vector will be sampled.
Vector radius - determines the size of the smallest ring vector.
Vector ring count - how many vector rings will be used.
Vector radius increment - by how much is the next ring vector larger than previous.
Vector rotation stride - by how much will the ring vector rotated during comparison.

Matching value stride - which colors are being changed. For example [R1,G1,B1,R2,G2,B2] with
stride 2 only [R1,B1,G2] will be compared. Can be used to reduce the amount of computation
required.
Matching offset - starting point of ring vector components comparison. If using stride 3, this can
be used to determine which color component is being used for comparison.

Average bias - mixes the SIVQ algorithm result pixel with average value over the ring vectors
around that pixel according to the specified bias. Smoothens the result and reduces circular
defects that arise in SIVQ.
Gamma adjust - adjusts the gamma value of the image. Can be used to highlight the matching
parts of the image.
Threshold - replaces values that are below the threshold with black. Can be used to remove
non-matching parts of the image.
Applications

Here we show example applications either as the end result or as a pre-processing step for other algorithms. We have used a red arrow to point out the pattern ring vector sampling location.

Shape tests

This is an example of recognizing a rotated letter “A” and finding the border of shape. Here we can see that this searching method has rotational invariance.
Cell images

Finding tumor cells. Here we highlight the cancerous cells.

Also we can use it as a pre-processing step for extracting the background.
Microorganisms

Identification of microorganisms having stained tissue.

River

Example of highlighting a water body on a satellite image.
Sky

Finding the sky on a hemispheric image.

As we can see this algorithm can be easily used for other purposes as well.
GPU optimization

SIVQ algorithm requires a considerable amount of simple vector computation. For batch processing this requires a lot of CPU time. Also if you want to easily modify the parameters then real-time visualization is a big benefit as parameters can be tweaked faster and therefore it's possible to find a better search pattern faster.

As most of the calculations are based on vectors and little data means that this algorithm should be easily implemented on the a GPU. This will improve the performance due to multicore architecture of GPUs.

Our primary tests so far have only improved the program speed about 25% (4 Core i7 CPU-s against 192 core Quadro 2000M GPU). This result is not impressive therefore there was no reason to do further testing on this implementation. The expected execution speed should be at least a magnitude higher, because similar convolution examples can improve the performance about 20x.

This GPU algorithm was probably bottlenecked by the memory access speed. The ring vector sampling requires a lot of non-local accesses to the memory and therefore does not use the caches properly. To overcome this problem we developed a pure way of using 1D vectors with little sampling per iteration and therefore less non-local accesses.

Image can be described by one dimensional vector $I$ by concatenating each row of the image to each other. The search pattern $p$ is a vector sampled on a circle. The points of the circle maintain constant offsets from the center during the concatenation of the image rows. This means that the search pattern $p$ has also an associated offset vector $o$.

The regular approach would be to find
\[
\min \sum_i (p_i - I_{c-i(i+k) \mod n})^2, \text{ where } k \in \{1..n\}
\]
for all pixel positions $c$. As can be seen this requires non-local accesses to the image $I$.

Matching to a certain position on a pattern can also be translated over the entire image. Considering an accumulator (for the difference) $A_{n,r}$ - where $r$ is the rotation of the pattern and the $n$ count of elements in the search vector. Then we can find the difference of pattern on each point during a certain rotation with:
\[
A_{i,r} = A_i + (I - p_{i+r})^\ast R[i]^\ast
\]
where $R[i]^\ast$ is the right circular shift matrix, and $^\ast$ is element-wise squaring operation. This multiplication is only necessary to write the formula in actual implementation this can be replaced by reindexing the values in the vector.

And to find the heat map we just need to find $D_n$ with formula:
\[
D_i = M(A_{n,i}, D_{i-1})
\]
where $M$ is element-wise minimum operation. This method should give us better memory locality with expense of some memory.

**Conclusion**

We achieved our primary goal to implement SIVQ algorithm and provide a simple interface to it. We did not achieve our secondary goal to make it real-time and usable for large scale batch processing by using GPUs for computation due to lack of understanding of GPU architecture. We shall still try to make it real-time outside of this project.

The code is available at [https://github.com/egonelbre/sivq](https://github.com/egonelbre/sivq).