OPTIMIZATION OF ANT COLONY EDGE DETECTION ALGORITHM

Hui Zhou

DMAVT
ETH Zürich
Zürich, Switzerland

ABSTRACT

In this paper, an ant-colony edge detection algorithm is implemented using the blocking strategy and non-repeated random number generator. An image is taken as an array to decrease the memory access and cache load of this algorithm. This approach speeds up the algorithm and can be extended in a parallel way.

1. INTRODUCTION

Ant Colony Edge Detection (ACED) is a heuristic approach for edge detection in image processing. Originated from ant colony optimization, it uses the pheromone trails generated by artificial ants to label the edges. In general, the ants are randomly distributed in an image and then moved by iterations. The ant movement is according to the transition probability [1, 2, 3],

$$p^n_{ij\to\Omega_{ij}} = \begin{cases} \frac{\left(\tau^{n-1}_{rs}\right)^{\alpha}(\eta_{rs})^{\beta}}{\sum_{rs}\left(\tau^{n-1}_{rs}\right)^{\alpha}(\eta_{rs})^{\beta}} & \text{if } (r,s)\in\Omega_{ij} \\ 0 & \text{otherwise} \end{cases}$$

where τ^n_{ij} is the pheromone of the nth construction step at (i,j) pixel position of the image and η_{ij} is the heuristic information at that location. Ω_{ij} is the neighborhood of pixel (i,j). α and β are input parameters with typical value ≈ 2.0. From this transition rule, we can see that the pixel with high heuristic information and pheromone will be frequently visited by the ants. After the movements of all ants, the pheromone τ^n_{ij} will be updated by

$$\tau^n_{ij} \leftarrow (1-\rho)\tau^n_{ij} + \lambda$$

where ρ is the pheromone evaporation rate and λ is the pheromone left by the ants which visited this site. A typical setting of λ [1, 3] is

$$\lambda = \begin{cases} \sum_k \eta_{ij} & \text{if } k\text{th ant visited } ij \text{ and } \eta_{ij} \geq t \\ 0 & \text{otherwise} \end{cases}$$

where t is a user-defined value to avoid noise. Due to the evaporation, the pheromone on the sites with low ant visiting frequency will eventually vanish so as to highlight the edges. In addition, each ant has memory length, which is to avoid the ant visiting previous positions stored in its memory. Besides this non-overlapping ant movement constraint, the initial ant position can also be randomly distributed over the region containing high heuristic information (e.g. $\eta_{ij} \geq t$) in a non-overlapping way.

In literature, Liu et al. [3] concluded that ACED yields better results than traditional edge detectors like Canny and Sobel but slower running speed. Some parallel ant colony optimizations are implemented using MPI [4], OpenMP [5] and GPU [6]. We refer to [7] for a review of parallel approaches. In this paper, we used the blocking strategy to implement the ACED proposed in [3]. The algorithm is run on a single core and can be extended in parallel.

2. BACKGROUND

The definition of heuristic information, according to [3], is

$$\eta_{ij} = \frac{\max|I_{i-u,j-v} - I_{i+u,j+v}|}{I_{\max}} \quad u,v \in \{-2,-1,0,1,2\}$$

where $I_{\max}$ is the maximum intensity value of gray-scale image I and I_{ij} is the intensity value of node (i,j) in image I. Note that the range of (u,v) value should be adjusted to $\{-1,0,1\}$ for the node (i,j) near or at the boundary of image. For instance, the heuristic information at the upper boundary of image (without 2 corners at the ends) degenerates to the normalized central difference $\eta_{ij} = \frac{|I_{i,j-1} - I_{i,j+1}|}{I_{\max}}$. Besides, the heuristic information is set to 0 at the corners of image. Similarly, Ω_{ij}, the neighborhood of node (i,j), is defined as a 1-ring around the node (i,j),

$$\Omega_{ij} = \{(r,s) \mid \max\{|u|,|v|\} = 1, u = r-i, v = s-j\}$$

which means each ant has 8 degrees of freedom for motion. However, it will decrease to 5 at the boundaries and 3 at the corners.

Combined all the ingredients mentioned before, with the proposed input parameter setting listed in Tab.1, the implementation follows Alg.1, where the Random function is implemented under the non-overlapping constraint as shown in Alg.2.

Input: Grayscale image of size $\text{height} \times \text{width}$
Output: Binary image of the edges
initialize the parameters, pheromone matrix τ_{ij} and heuristic information matrix η_{ij};
for *k=0; k<K; k++* **do**
  $\text{ants}_{k0} = \text{Random}(k)$;
end
for *n=0;n¡N;n++* **do** construction iterations
  for *k=0; k<K; k++* **do** get current position of ant
    $q = \text{ants}_{k0}$;
    $i = q/\text{width}$;
    $j = q\%\text{width}$;
    for *l=1; l<L; l++* **do** move L-1 steps
      for $rs \in \Omega_{ij}$ **do** calculate the transition probability
        $p^n_{ij\to\Omega_{ij}}$;
      end
      if $ij_{\text{old}} \in \Omega_{ij}$ **then** avoid the position in memory
        $p^n_{ij\to ij_{\text{old}}} = 0$;
      end
      $p^n_{ij_{\max}} = \max\{p^n_{ij\to\Omega_{ij}}\}$;
      if $p^n_{ij_{\max}} == 0$ or $\eta_{ij_{\max}} < t$ **then**
        $ij_{\text{new}} = \text{Random}(k)$;
      else
        $ij_{\text{new}} = ij_{\max}$;
      end
      $\text{ants}_{kl} = i_{\text{new}} * \text{width} + j_{\text{new}}$;
    end
  end
  $\tau_{ij} \leftarrow (1-\rho) * \tau_{ij} + \sum_{kl} \eta_{\text{ants}_{kl}}$;
  $\text{ants}_{k0} \leftarrow \text{ants}_{k(l-1)}$;
end

Algorithm 1: Ant Colony Edge Detection algorithm

Function $\text{Random}(k)$
  $c = 0$;
  $q = 0$;
  $r = \text{rand}()\%A$;
  while $q < k$ and $c < A$ **do**
    $r = \text{rand}()\%A$;
    for *q=0; q<k; q++* **do**
      if $\text{ants}_{q0} == r$ *or* $\eta_r < t$ **then**
        break;
      end
    end
    c++;
  end
return r;

Algorithm 2: Random function for the ant distribution

image area	$A = \text{height} * \text{width}$
image perimeter	$P = 2 * (\text{height} + \text{width})$
initial τ_{ij}	$\tau_0 = 1\text{e}-4$
number of ants	$K = \lfloor\sqrt{A}\rfloor$
ant moving steps	$L = \lfloor 3\sqrt{A}\rfloor$
memory length	$M = \lfloor\sqrt{P}\rfloor$
τ power factor	$\alpha = 2$
η power factor	$\beta = 2$
evaporation rate	$\rho = 0.02$
η threshold	$t = 0.1$
construction iterations	$N = 3$

Table 1. Proposed parameter values for initialization, $\lfloor\cdot\rfloor$ is the floor function

To speed up the algorithm, the image I is taken as a 1-D array rather than a 2-D matrix. In other words, the pixel position (i, j) is zipped as $ij = i * \text{width} + j$. This is beneficial for storing the positions of ants and generating the non-repeated random numbers. In detail, the ants_{kl} matrix stores the position of the kth ant at the l th step in only 1 float ij rather than 2 floats (i, j). When the movement of ant need to be conducted, the float ij can be unzipped to 2 floats (i, j) by integer division $i = ij/\text{width}$ and integer mod $j = ij\%\text{width}$. This will increase the floating point operations meanwhile decrease the memory access and the cache load. Moreover, the random number generated by $ij = \text{rand}()\%A$ will have less chance of overlapping than 2 separate random numbers generated by $i = \text{rand}()\%\text{height}$ and $j = \text{rand}()\%\text{width}$ because the period $A = \text{height} * \text{width}$ is larger than the periods height and width.

In addition, the binary image of the edges is obtained by using the threshold τ_0. The sites with the higher pheromone $\tau_{ij} > \tau_0$ are labeled as the edges (black) otherwise the region (white).

3. BLOCKING STRATEGY

To further speed up the algorithm, it is necessary to slice the large input image so that the matrices can be fit into the cache. Therefore, Alg.1 is wrapped into a Block function with arguments η, (x_0, y_0) and $(width, height)$. The block is defined by $[x_0, x_0 + width] \times [y_0, y_0 + height]$. Note that the heuristic information matrix η_{ij} is initialized outside the Block function so that the heuristic information at the boundaries of sliced image is complete. It is better to check the value of A and η to make sure the block area $A > 0$ and the block contains more than 1 color type (i.e. $\#\{\eta_{ij} \mid \eta_{ij} > t, (i, j) \in [x_0, x_0 + width] \times [y_0, y_0 + height]\} > 0$). Finally, the algorithm with blocking follows the structure in Alg.3, where the block size is $height \times width$ in comparison with the image size Height $\times$ Width.

Input: Grayscale image of size Height $\times$ Width
Output: Binary image of the edges
assert($I_{\max} > 0$);
initialize heuristic information matrix η_{ij};
$b_x = width$;
$b_y = height$;
$x_1 = \text{Width}/width * width$;
$y_1 = \text{Height}/height * height$;
for $y_0 = 0$; $y_0 < y_1$; $y_0 += b_y$ **do**
  for $x_0 = 0$; $x_0 < x_1$; $x_0 += b_x$ **do**
    Block$(x_0, y_0, b_x, b_y, \eta)$;
  end
  for $x_0 = x_1$; $x_0 < \text{Width}$; $x_0 += b_x$ **do**
    Block$(x_0, y_0, \text{Width} - x_1, b_y, \eta)$;
  end
end
for $y_0 = y_1$; $y_0 < \text{Height}$; $y_0 += b_y$ **do**
  for $x_0 = 0$; $x_0 < x_1$; $x_0 += b_x$ **do**
    Block$(x_0, y_0, b_x, \text{Height} - y_1, \eta)$;
  end
  for $x_0 = x_1$; $x_0 < \text{Width}$; $x_0 += b_x$ **do**
    Block$(x_0, y_0, \text{Width} - x_1, \text{Height} - y_1, \eta)$;
  end
end

Algorithm 3: Blocking strategy for ACED

4. EXPERIMENTAL RESULTS

The image testing is run on Intel i5-4690 CPU (3.50GHz, L1d cache: 32K, L1i cache: 32K, L2 cache: 256K, L3 cache: 6144K) with gcc 6.3.1 compiler (-O0 flag). The result of ACED algorithm is shown in Fig.1. ACED with or without blocking yields nearly the same run time when the image size is small (e.g. $1s/1s$ for 512×512 lena image) but significant run time difference when the image size is large (e.g. $5s/25s$ for 1024×1024 lena image). We can also see that the image color given by ACED is deeper than that given by its blocking. This is due to the fact that the ant movements are constrained inside each block rather than walking around the whole image. Therefore the overall pheromone distribution of ACED with blocking is more sparser than the original ACED. Despite of this, the blocked ACED still captures the features of image and has a faster speed than the original one.

Furthermore, ACED with blocking is tested by an extreme case, that is, one black pixel image as shown in Fig.2. We can hardly locate the black pixel by naked eyes, however, with the help of ACED, we can easily pinpoint its location. This has potential application in cosmology.

Fig. 1. lena image of size 1024×1024 (left); ACED (middle) and its blocking (right) run by Intel i5-4690 CPU with gcc 6.3.1 compiler -O0 flag

Fig. 2. One black pixel image of size 325×446(left); blocked ACED circle out the black pixel in red color (right)

5. CONCLUSIONS

The blocked ACED algorithm achieved comparable results and faster speed than the original one. It can be applied in cosmology to locate the stars from the image of galaxies. For the further optimization of this algorithm, some parallel approaches (e.g. MPI) can be used.

6. REFERENCES

[1] Hossein Nezamabadi-Pour, Saeid Saryazdi, and Esmat Rashedi, "Edge detection using ant algorithms," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 10, no. 7, pp. 623–628, 2006.

[2] Jing Tian, Weiyu Yu, and Shengli Xie, "An ant colony optimization algorithm for image edge detection," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on.* IEEE, 2008, pp. 751–756.

[3] Xiaochen Liu and Suping Fang, "A convenient and robust edge detection method based on ant colony optimization," *Optics Communications*, vol. 353, pp. 147–157, 2015.

[4] Marcus Randall and Andrew Lewis, "A parallel implementation of ant colony optimization," *Journal of Parallel and Distributed Computing*, vol. 62, no. 9, pp. 14211432, 2002.

[5] Pierre Delisle, Michaël Krajecki, Marc Gravel, and Caroline Gagné, "Parallel implementation of an ant colony optimization metaheuristic with openmp," in *Proceedings of the 3rd European Workshop on OpenMP (EWOMP01), Barcelona, Spain*, 2001.

[6] Laurence Dawson and Iain A Stewart, "Accelerating ant colony optimization-based edge detection on the gpu using cuda," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 1736–1743.

[7] Martín Pedemonte, Sergio Nesmachnow, and Héctor Cancela, "A survey on parallel ant colony optimization," *Applied Soft Computing*, vol. 11, no. 8, pp. 5181–5197, 2011.

[8] N.J. Higham, *Handbook of Writing for Mathematical Sciences*, SIAM, 1998.

[9] W. Strunk Jr. and E.B. White, *Elements of Style*, Longman, 4th edition, 2000.