International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume –04, Issue – 05, May 2019, PP –28-32

Reconstruction of Binary Images with Gray Value

M. Anantha Lakshmi [1], Dr. Prof. P. Kailasapathi[2], Dr. A. Sanjeevi Kumar[3]
1.(Electronics and Communication Engineering/ Research Scholar /Annamalai University, Annamalainagar – 608 002. Tamilnadu, India.)
2.(Electronics and Communication Engineering / Annamalai University, Annamalainagar – 608 002. Tamilnadu, India.)
3.(Electronics and Communication Engineering /Meenakshi Academy of Higher Education and Research, India)

Abstract: The algebraic reconstruction technique (ART) is a class of iterative algorithms used in computed tomography. These reconstruct an image from a series of angular projections (a sinogram). and showed its use in image reconstruction; whereas the method is known as Kaczmarz method in numerical linear algebra.
ART can be considered as an iterative solver of a system of linear equations . The values of the pixels are considered as variables collected in a vector and the image process is described by a matrix . The measured angular projections are collected in a vector

1. Introduction

1.1 SDART (Search Based Algorithm)

A* (pronounced "A star") is a computer algorithm that is widely used in path finding and graph traversal, which is the process of finding a path between multiple points, called "nodes". It enjoys widespread use due to its performance and accuracy. However, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance, although other work has found A* to be superior to other approaches.

Peter Hart, Nils Nilsson and Bertram Raphael of Stanford Research Institute (now SRI International) first published the algorithm in 1968. It can be seen as an extension of Edsger Dijkstra's 1959 algorithm. A* achieves better performance by using heuristics to guide its search.

A* is an informed search algorithm, or a best-first search, meaning that it is formulated in terms of weighted graphs: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost (least distance travelled, shortest time, etc.). It does this by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until its termination criterion is satisfied.

At each iteration of its main loop, A* needs to determine which of its paths to extend. It does so based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal. Specifically, A* selects the path that minimizes where n is the next node on the path, $g(n)$ is the cost of the path from the start node to n, and $h(n)$ is a heuristic function that estimates the cost of the cheapest path from n to the goal. A* terminates when the path it chooses to extend is a path from start to goal or if there are no paths eligible to be extended.The heuristic function is problem-specific. If the heuristic function is admissible, meaning that it never overestimates the actual cost to get to the goal, A* is guaranteed to return a least-cost path from start to goal.

Typical implementations of A* use a priority queue to perform the repeated selection of minimum (estimated) cost nodes to expand. This priority queue is known as the open set or fringe. At each step of the algorithm, the node with the lowest $f(x)$ value is removed from the queue, the f and g values of its neighbours are updated accordingly, and these neighbours are added to the queue. The algorithm continues until a goal node has a lower f value than any node in the queue (or until the queue is empty). The f value of the goal is then the cost of the shortest path, since h at the goal is zero in an admissible heuristic.

The algorithm described so far gives us only the length of the shortest path. To find the actual sequence of steps, the algorithm can be easily revised so that each node on the path keeps track of its predecessor. After this algorithm is run, the ending node will point to its predecessor, and so on, until some node's predecessor is the start node.

As an example, when searching for the shortest route on a map, $h(x)$ might represent the straight-line distance to the goal, since that is physically the smallest possible distance between any two points.

If the heuristic h satisfies the additional condition $h(x) \le d(x, y)+ h(y)$ for every edge (x, y) of the graph (where d denotes the length of that edge), then h is called monotone, or consistent. In such a case, A* can be implemented more efficiently roughly speaking, no node needs to be processed more than once and A* is equivalent to running Dijkstra's algorithm with the reduced cost $d'(x, y) = d(x, y) + h(y) - h(x)$.

1. Properties

Like breadth-first search, A* is complete and will always find a solution if one exists provided for fixed .If the heuristic function *h* is admissible, meaning that it never overestimates the actual minimal cost of reaching the goal, then A* is itself admissible (or optimal) if we do not use a closed set. If a closed set is used, then *h* must also be monotonic (or consistent) for A* to be optimal. This means that for any pair of adjacent nodes *x* and *y*, the length of the edge between them, we must have: This ensures that for any path *X* from the initial node to *x*: where *L* is a function that denotes the length of a path, and *Y* is the path *X* extended to include *y*. In other words, it is impossible to decrease (total distance so far + estimated remaining distance) by extending a path to include a neighboring node. (This is analogous to the restriction to nonnegative edge weights in Dijkstra's algorithm.) Monotonicity implies admissibility when the heuristic estimate at any goal node itself is zero, since (letting $P = (f,v_1,v_2,...,v_n,g)$ be a shortest path from any node *f* to the nearest goal *g*):

A* is also optimally efficient for any heuristic *h*, meaning that no optimal algorithm employing the same heuristic will expand fewer nodes than A*, except when there are multiple partial solutions where *h* exactly predicts the cost of the optimal path. Even in this case, for each graph there exists some order of breaking ties in the priority queue such that A* examines the fewest possible nodes.

2. Special cases

Dijkstra's algorithm, as another example of a uniform-cost search algorithm, can be viewed as a special case of A* General depth-first search can be implemented using A* by considering that there is a global counter *C* initialized with a very large value. Every time we process a node we assign *C* to all of its newly discovered neighbors. After each single assignment, we decrease the counter *C* by one. Thus the earlier a node is discovered, the higher its value. Both Dijkstra's algorithm and depth-first search can be implemented more efficiently without including and value at each node.

3. Implementation details

There are a number of simple optimizations or implementation details that can significantly affect the performance of an A* implementation. The first detail to note is that the way the priority queue handles ties can have a significant effect on performance in some situations. If ties are broken so the queue behaves in a LIFO manner, A* will behave like depth-first search among equal cost paths (avoiding exploring more than one equally optimal solution).

When a path is required at the end of the search, it is common to keep with each node a reference to that node's parent. At the end of the search these references can be used to recover the optimal path. If these references are being kept then it can be important that the same node doesn't appear in the priority queue more than once (each entry corresponding to a different path to the node, and each with a different cost). A standard approach here is to check if a node about to be added already appears in the priority queue. If it does, then the priority and parent pointers are changed to correspond to the lower cost path. A standard binary heap based priority queue does not directly support the operation of searching for one of its elements, but it can be augmented with a hash table that maps elements to their position in the heap, allowing this decrease-priority operation to be performed in logarithmic time. Alternatively, a Fibonacci heap can perform the same decrease-priority operations in constant amortized time.

4. Admissibility & Optimality

A* is admissible and, in some circumstances, considers fewer nodes than any other admissible search algorithm with the same heuristic. This is because A* uses an "optimistic" estimate of the cost of a path through every node that it consider optimistic in that the true cost of a path through that node to the goal will be at least as great as the estimate. But, critically, as far as A* "knows", that optimistic estimate might be achievable.

To prove the admissibility of A*, the solution path returned by the algorithm is used.When A* terminates its search, it has found a path whose actual cost is lower than the estimated cost of any path through any open node. But since those estimates are optimistic, A* can safely ignore those nodes. In other words, A* will never overlook the possibility of a lower-cost path and so is admissible.

Suppose now that some other search algorithm B terminates its search with a path whose actual cost is *not* less than the estimated cost of a path through some open node. Based on the heuristic information it has, Algorithm B cannot rule out the possibility that a path through that node has a lower cost. So while B might consider fewer nodes than A*, it cannot be admissible. Accordingly, A* considers the fewest nodes of any admissible search algorithm.

This is only true if both:

- A* uses an admissible heuristic. Otherwise, A* is not guaranteed to expand fewer nodes than another search algorithm with the same heuristic.
- A* solves only one search problem rather than a series of similar search problems. Otherwise, A* is not guaranteed to expand fewer nodes than incremental heuristic search algorithms.[12]

5. Bounded relaxation

While the admissibility criterion guarantees an optimal solution path, it also means that A* must examine all equally meritorious paths to find the optimal path. To compute approximate shortest paths, it is possible to speed up the search at the expense of optimality by relaxing the admissibility criterion. Oftentimes we want to bound this relaxation, so that we can guarantee that the solution path is no worse than $(1 + \varepsilon)$ times the optimal solution path. This new guarantee is referred to as ε-admissible.

There are a number of ε-admissible algorithms:

- Weighted A*/Static Weighting. If $h_a(n)$ is an admissible heuristic function, in the weighted version of the A* search one uses $h_w(n) = \varepsilon\ h_a(n), \varepsilon > 1$ as the heuristic function, and perform the A* search as usual (which eventually happens faster than using h_a since fewer nodes are expanded). The path hence found by the search algorithm can have a cost of at most εtimes that of the least cost path in the graph.
- Dynamic Weighting uses the cost function , where is the depth of the search and N is the anticipated length of the solution path.
- Sampled Dynamic Weighting uses sampling of nodes to better estimate and debias the heuristic error.
- uses two heuristic functions. The first is the FOCAL list, which is used to select candidate nodes, and the second h_F is used to select the most promising node from the FOCAL list.
- A_ε selects nodes with the function , where A and B are constants. If no nodes can be selected, the algorithm will backtrack with the function , where C and D are constants.
- AlphA* attempts to promote depth-first exploitation by preferring recently expanded nodes. AlphA* uses the cost function , where λ and Λ are constants with $\pi(n)$ is the parent of n, and $\tilde{n}$ is the most recently expanded node.

6. Complexity

The time complexity of A* depends on the heuristic. In the worst case of an unbounded search space, the number of nodes expanded is exponential in the depth of the solution (the shortest path) d: $O(b^d)$, where b is the branching factor (the average number of successors per state). This assumes that a goal state exists at all, and is reachable from the start state; if it is not, and the state space is infinite, the algorithm will not terminate.

The heuristic function has a major effect on the practical performance of A* search, since a good heuristic allows A* to prune away many of the b^d nodes that an uninformed search would expand. Its quality can be expressed in terms of the effective branching factor b*, which can be determined empirically for a problem instance by measuring the number of nodes expanded.

1).The time complexity is polynomial when the search space is a tree, there is a single goal state, and the heuristic function h meets the following condition: where h^* is the optimal heuristic, the exact cost to get from x to the goal. In other words, the error of h will not grow faster than the logarithm of the "perfect heuristic" h^*that returns the true distance from x to the goal.

7. APPLICATIONS

A* is commonly used for the common pathfinding problem in applications such as games, but was originally designed as a general graph traversal algorithm. It finds applications to diverse problems, including the problem of parsing using stochastic grammars in NLP. Other cases include an Informational search with online learning.

8. RELATIONS TO OTHER ALGORITHMS

A* apart from a greedy best-first search algorithm is that it takes the cost/distance already traveled, $g(n)$, into account.Some common variants of Dijkstra's algorithm can be viewed as a special case of A* where the heuristic for all nodes; in turn, both Dijkstra and A* are special cases of dynamic programming. A* itself is a special case of a generalization of branch and bound and can be derived from the primal-dual algorithm for linear programming.

1. Simultaneous Iterative Reconstruction Technique

A represents the action of the scanner, also known as the forward projection. Each row of AA contains the coefficients of an equation that corresponds to a single ray. It describes how the pixels are combined into ray sums. The transposed matrix, AT *back projects* the projection images onto the reconstruction area. Given a ray sum, it describes which pixels are hit by that ray.

SIRT alternates forward and back projections. Its *update equation* is

$$x(t+1)=x(t)+CATR(b-Ax(t))$$

where C and R are diagonal matrices that contain the inverse of the sum of the columns and rows of the system matrix, so cjj=1/∑iaij and rii=1/∑jaijrii=1/∑jaij These matrices compensate for the number of rays that hit each pixel and the number of pixels that are hit by each ray.

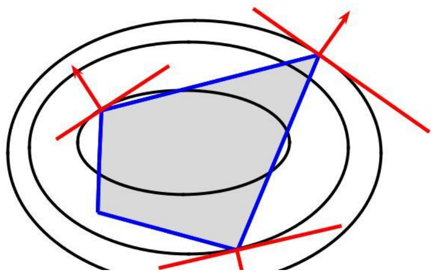
Iterations start with x(0)=0 The update equation then does the following things.

1. The current reconstruction x(t) is forward projected: Ax(t).
2. The result is subtracted from the original projections: b−Ax(t).
3. This difference is then back projected. In essence this is done by multiplying with AT, but weighted with C and R.
4. This results in the correction factor .
5. The correction factor is then added to the current reconstruction, and the whole process is repeated from.

The wavefront reconstruction is formulated as an inverse problem where the complex exponent or the amplitude and phase of this exponent are assumed to admit sparse representations in suitable sparsifying transforms (dictionaries). The sparse modeling is a form of regularization of the inverse problem. For design of these overcomplete sparsifying dictionaries we use Block Matching 3D (BM3D) and learning dictionary techniques. Various optical setups (interferometric and non-interferometric) are considered with algorithms developed for Gaussian and Poissonian noise in intensity measurements. Algorithms:

2. Non Convex Optimization Techniques

Linearization of basic nonconvexities of the problem under scrutiny and, consequently, reduction of the problem to a family of (partially) linearized problems. Application of convex optimization methods for solving linearized problems and, as a consequence, "within" special local search methods. Construction of "good" approximations (resolving sets) of level surfaces/epigraph boundaries of convex functions.



3. Gray Value Estimation

Grayscale images are distinct from one-bit bi-tonal black-and-white images, which in the context of computer imaging are images with only the two colors, black, and white (also called *bilevel* or binary images). Grayscale images have many shades of gray in between.

4. Results and Discussion

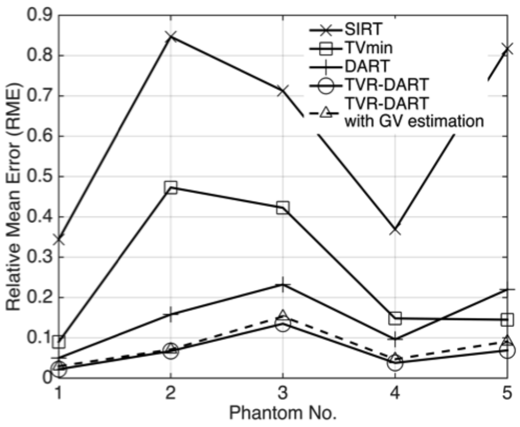


Fig No:1 Comparison of the Relative Mean Error (RME) of reconstructions among SIRT, TVmin, DART, TVR-DART with and without gray value estimation for all 5 phantoms.

Conclusions

ART over other reconstruction methods (such as filtered backprojection) is that it is relatively easy to incorporate prior knowledge into the reconstruction process. ART falls into the category of Iterative reconstruction techniques.

Acknowledgments

The authors would like to thank Department of Electronics and Communication, Faculty of Engineering and Technology, Annamalai University, Annamalai Nagar for his support within our joint project. They thank Electronics and Communication, Meenakshi Academy of Higher Education and Research,Chennai-78.

[1]. P. A. Midgley and R. E. Dunin-Borkowski, “Electron tomography and holography in materials science,” Nature Mater., vol. 8, pp. 271–280, Apr. 2009.
[2]. F. Natterer, The Mathematics of Computerized Tomography. Philadelphia, PA, USA: SIAM, 2001.
[3]. D. L. Donoho, “Compressed sensing,” IEEE Trans. Inf. Theory, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.
[4]. E. J. Candès, J. Romberg, and T. Tao, “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information,” IEEE Trans. Inf. Theory, vol. 52, no. 2, pp. 489–509, Feb. 2006.
[5]. T. Schüle, C. Schnörr, S. Weber, and J. Hornegger, “Discrete tomography by convex–concave regularization and D.C. programming,” Discrete Appl. Math., vol. 151, nos. 1–3, pp. 229–243, Oct. 2005.
[6]. [6] A. Chambolle and T. Pock, “A first-order primal-dual algorithm for convex problemswith applications to imaging,” J. Math. Imag. Vis., vol. 40, no. 1, pp. 120–145, May 2011.
[7]. D. L. Donoho, M. Vetterli, R. A. DeVore, and I. C. Daubechies, “Datacompression and harmonic analysis,” IEEE Trans. Inf. Theory, vol. 44,no. 6, pp. 2435–2476, Oct. 1998.
[8]. Y. Meyer, Wavelets and Operators. Cambridge, U.K.: CambridgeUniv. Press, 1993.
[9]. D. L. Donoho, “Unconditional bases are optimal bases for data compression and for statistical estimation,” Appl. Comput. Harmonic Anal., vol. 1, pp. 100–115, 1993.
[10]. A. Pinkus, “n-widths and optimal recovery in approximation theory,” in Proc. Symp. Applied Mathematics, vol. 36, C. de Boor, Ed., Providence, RI, 1986, pp. 51–66.
[11]. J. F. Traub and H. Woziakowski, A General Theory of Optimal Algorithms. New York: Academic, 1980.
[12]. D. L. Donoho, “Unconditional bases and bit-level compression,” Appl. Comput. Harmonic Anal., vol. 3, pp. 388–92, 1996.
[13]. A. Pinkus, n-Widths in Approximation Theory. New York: Springer-Verlag, 1985.