World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:15, No:12, 2021

Comparative Analysis of Classical and Parallel Inpainting Algorithms Based on Affine Combinations of Projections on Convex Sets

Irina Maria Artinescu, Costin Radu Boldea and Eduard-Ionut Matei

Abstract—The paper is a comparative study of two classical variants of parallel projection methods for solving the convex feasibility problem with their equivalents that involve variable weights in the construction of the solutions. We used a graphical representation of these methods for inpainting a convex area of an image in order to investigate their effectiveness in image reconstruction applications. We also presented a numerical analysis of the convergence of these four algorithms in terms of the average number of steps and execution time, in classical CPU and, alternativaly, in parallel GPU implementation.

Keywords—Convex Feasibility Problem, Convergence analysis, Inpainting, Parallel Projection Methods.

I. Introduction

ONE aim in all the classical problems in computational mathematics is to look for a solution that satisfies a given set of equations or inequations. In the past the projection methods were utilized to solve some systems of linear equations in Euclidean spaces [8] and were modified to be applied at systems of linear inequalities in [3], [16], [17]. The classical step in these first algorithms consists of projections onto some affine subspaces or half-spaces. Later, the method becomes more sophisticated [13], [14], [15], being adapted to resolve the overall convex feasibility problem of finding points inside the intersection of a family of closed convex sets in a given metric space. A detailed mathematical approach may be found in [4], [9].

The affine projection methods have numerous applications in watermarking, in data compression, in NMR imaging, neural networks or in image filtering - see [7] for other applications - but it was never employed in image recovery by inpainting until 2018, when an affine projection method SCAPF was proposed in [1] to recover small "scratches" in damaged images. In this paper we propose two variants of inpaintig algorithms based on parallel projection methods (PPM) which uses affine combinations with variable weights of the projections of an outer point on the sides of a convex polygon , in order to obtain a (finite) series of points ending inside it. The inpainting technique uses these finite series of points to transfer information from the outside to inside of the convex polygon, thus defining a new method of generating patterns for filling the small damaged regions of images. The method is proposed as an alternative to the classical inpainting techniques by copying external patterns [6],[5] or by "filling" the damaged region using textures synthesis [12].

Although the mathematical analysis of the convergence of assorted parallel projection methods used to solve the convex feasibility problem was extensively described in [4] and [10], the derived algorithms don't converge generally in finite time. Very few studies approached the direction of determining which algorithm converges in finite time almost everywhere within the Euclidean space. For instance, only in 2007 an algorithm in finite steps was proposed, supported alternative projections on two convex sets, to resolve the linear conic optimization problem [18]. Concerning the classical EMOPP algorithm [9], in 2018 was proved [2] that it does not converge in the finite number of steps in the case of a rectangular convex, for starting points belonging to large regions of the Euclidean plane.

Under these circumstances, a comparative study of the convergence of our variants of affine projection methods is required to verify their usefulness for graphical applications. For this purpose , we implemented the two algorithms proposed by us, together with the classic PPM and EMOPP algorithms, in the first stage in C ++, using them, comparatively, to inpainting a rectangular region of an image affected by information loss.

Nowadays, the need to process large amounts of data, such as image processing, as well as the mathematical complexity of the calculations required in graphics, have led to the development of processing methods using graphics processors (GPU). Starting from 2013, GPU based parallelization, implemented in CUDA or OpenCL, has been successfully used to accelerate the projection process in iterative reconstruction of images [19], [20]. In the case of projection-based inpainting algorithms, the complexity of the calculation is determined by the number of iterations required to transfer a pixel from outside the affected area inside it and by the number of pixels needed to "fill" the image. In the second part of this paper we described and analysed comparatively the parallel implementation in OpenCL on the GPU of the above algorithms.

II. The Convex Feasibility Problem (CFP) and the Parallel Projection Method (PPM)

The convex feasibility problem (CFP) was formulated in [9] as :

I.M. Artinescu is with the Department of Computer Science, West University of Timişoara, 4 V.Pârvan, Timişoara, Romania, e-mail: artinescu_irina@yahoo.com
C.R. Boldea (cboldea@inf.ucv.ro) and E.-I. Matei (eduard-matei@outlook.com) are with Department of Computer Science, University of Craiova, 13 A.I.Cuza, Craiova, Romania.

Given m closed convex sets $C_1, C_2, \ldots, C_m \subseteq \mathcal{R}^n$, with nonempty intersection, $\cap C_i \neq \emptyset$, defined by $C_i = \{x \in \mathcal{R}^n \mid f_i(x) <= 0\}$, with $f_i : \mathcal{R}^n \to \mathcal{R}$ a convex function, the CFP is to find a point $x \in C = \bigcap_{i=1}^{m} C_i$.

This problem was approached using projection algorithms that propose, starting from an arbitrary point exterior to $C = \bigcap_{i=1}^{m} C_i$, to construct a sequence of points converging to a point inside C [4].

The *Parallel Projection Method (PPM)*, which is governed by the iteration ([11]):

$$(\forall n \in \mathcal{N}) Q_{n+1} = Q_n + \lambda_n (\sum_{i \in 1..n} w_i P_i(Q_n) - Q_n), \quad (1)$$

where $\epsilon < \lambda_n < 2-\epsilon$ are the *relaxation* parameters, $0 < \epsilon < 1$ and $\sum_{i \in 1..n} w_i = 1$. The fixed weight w_i and the relaxation parameters λ_n introduced a significative modification of efficiency of this algorithm. Later, some modifications of this method, involving different classes of control index sets $\{I_n\}$, that proposed a generalization of PPM under the name *Extrapolated Method of Parallel Projections (EMOPP)* were analyzed in [9]. The iteration of this method is similar to (1):

$$(\forall n \in \mathcal{N}) Q_{n+1} = Q_n + \lambda_n (\sum_{i \in I_n} w_i P_i(Q_n) - Q_n), \quad (2)$$

where the indices set $\{I_n\}$, called *control* sequence, are variable from one iteration to another. Many variants of the control sequences where studied in [9].

Although EMOPP is generally faster than PPM, it does not always converge in finite number of steps [2]. To correct this deficiency, an iterative algorithm called Scratch Covering by Affine Projection Filling (SCAPF), derived from PPM and used for inpaintig a simple scratch of an image was introduced in [1]. The SCAPF algorithm used at each iteration variable weights of the affine combination of projections for each iteration (1), weights that depend on the relative position of the current point relative to the convex target. In this paper we compared the convergence and time efficiency of two iterative variants of SCAPF implemented on CPU arhitecture with the original PPM and EMOPP algorithms, and we proposed a parallel implementation of these algorithms using the graphical processor (GPU).

III. The Proposed Variants of Parallel Projection Method

The algorithms proposed by us have a starting point based on Parallel Projection Method (1) with constant $\lambda_i \equiv 1$ and the Extrapolated Method of Parallel Projections (2) and are described below:

mPPM AND mEMOPP ALGORITHMS

Input A finite set $\{\mathcal{P}\} = \{P_1, P_2, \ldots, P_n\}$ of a convex polygon.

Output The image of the inpainted polygon with $NPmax$ exterior points and the descriptive analysis of the number of iterations for all points

1 Define $O \leftarrow \frac{1}{n} \sum_{i=1}^{n} P_i$; *(The center of the polygon)*

2 Define an outer region of the polygon as the starting points of the algorithm:

$$\mathcal{RG} \leftarrow \left\{Q \in \mathcal{R}^2 | d(Q,O) < 2 * r\right\},$$

where $r \leftarrow \max_j \{d(O, P_j)\}$.

3 **for**$j \leftarrow 1$ **to** $NPmax$

Generate a random point $Q_{j,0} \in \mathcal{RG} \cap Ext(\mathcal{P})$

4 $j \leftarrow 1$ and $k \leftarrow 0$

5 **repeat**

5.1 Determine the projections on the polygon sides:

$$M_{j,k,i} = pr(Q_{j,k}, P_i P_{i+1}),$$

by convention $(P_{n+1} = P_1)$ and $d_{i,k} = dist(Q_{j,k}, P_i P_{i+1})$

5.2 Compute the weights

$$w_{i,k} = \frac{1/(d_{i,k}+1)}{\sum_{i \in I} 1/(d_{i,k}+1)}$$

where $I = \{1, 2, \ldots, n\}$. [1]

5.3 The next point $Q_{j,k+1}$ is determined by

$$Q_{j,k+1} = Q_{j,k} + \lambda_k (\sum_{i \in I} w_i M_{j,k,i} - Q_{j,k})$$

until $(Q_{j,k+1} \in Int\mathcal{P})$ or $(k > NrMaxIterations)$

6 **if** $Q_{j,k+1} \in Int(\mathcal{P})$ **then** $color(Q_{j,k+1}) \leftarrow color(Q_{j,k})$

7 **repeat from Step 5 with** $j \leftarrow j+1$ **and** $k \leftarrow 0$**.**

The control sequence λ_i is chosen to verify $1 < \lambda_i < 2 - \epsilon$ where $0 < \epsilon < 1/2$.

The modified version of EMOPP is obtained taking in consideration only the projections of $Q_{j,k}$ on the semi-plans that contain the polygon and don't contain the point $Q_{j,k}$. The weights are computed with the identical formula, within the condition $\sum w_i = 1$ for $i \in I_k$ where I_k is that the set of semi-plans taken into consideration at the k iteration.

Let's consider the set of indexes $I = \{1, 2, ..., n\}$ and the function $f : \mathbb{R}^2 \to \mathbb{R}^2$:

$$f(Q) = Q + \lambda_j (\sum_{i \in I} w_i \cdot pr(Q, P_i P_{i+1}) - Q) \quad (3)$$

where $d_i = dist(Q, pr(Q, P_i P_{i+1}))$, $(P_{n+1} = P_1)$ and $pr(\cdot, P_i P_{i+1})$ is the projection operator onto the $P_i P_{i+1}$ line and

$$w_i = \frac{1/(d_i+1)}{\sum_{j \in I} 1/(d_j+1)}.$$

In the case of mEMOPP, the set of indexes I will be replaced by the set I' of indexes i verifying that $P_i P_{i+1}$ separates Q from the rest of the polygon's $\mathcal{P}$ vertices $(P_{n+1} = P_1)$

The convergence of the algorithms is assured by the following theorem:

Theorem 1. Consider a sequence of points defined by $Q_{k+1} = f(Q_k)$ for $k \in \mathcal{N}$ with Q_0 arbitrary fixed outside the polygon $\mathcal{P}$. If the set of relaxation parameters verifies

[1] The distances are measured in pixels and the weights $\{w_{i,k}\}$ are inversely proportional to $\{d_{i,k}+1\}$ in order to eliminate the possible division by 0.

$1 < \lambda_i < 2-\epsilon$ with $0 < \epsilon < 1/2$, then for any point $C \in int(\mathcal{P})$:

1) we have

$$\|Q_{k+1}-C\|^2 <= \|Q_k-C\|^2 - \lambda_k(2-\lambda_k)\sum_{i\in I} w_{i,k}d_{i,k}^2 \quad (4)$$

for (mPPM) and

$$\|Q_{k+1}-C\|^2 <= \|Q_k-C\|^2 - \lambda_k(2-\lambda_k)\sum_{i\in I_k} w_{i,k}d_{i,k}^2 \quad (5)$$

for (mEMOPP), where I_k is the set of index i that P_iP_{i+1} separates Q_k from the rest of the polygon's $\mathcal{P}$ vertices,

2) for both algorithms:

$$\|Q_{k+1}-Q_k\|^2 <= \frac{1}{\epsilon}(\|Q_k-C\|^2-\|Q_{k+1}-C\|^2), \quad (6)$$

3) the sum: $\sum_k \|Q_{k+1}-Q_k\|^2 <= \frac{1}{\epsilon}dist(Q_0,\mathcal{P})$.

Then the sequence $\{Q_k\}$ converges in norm to some interior point of $\mathcal{P}$ for both mPPM and mEMOPP algorithms.

The demonstration of the theorem follows the step of similar theorem 2.16 and the corollary 3.16 from [4].

Proof

We note $M_{k,i} = pr(Q_k, P_iP_{i+1})$. The first point is obtained by the following relations (written here for mPPM algorithm):

$$\|Q_{k+1}-C\|^2 = \|Q_k - C + \lambda_k \cdot \sum_{i\in I} w_{i,k}\cdot M_{k,i} - Q_k\|^2 \quad (7)$$

$$= \|Q_k-C\|^2 + 2\lambda_k \left\langle Q_k - C, \sum_{i\in I} w_{i,k}\cdot M_{k,i} - Q_j \right\rangle$$
$$+\lambda_k^2 \|\sum_{i\in I} w_{i,k}\cdot M_{k,i} - Q_k\|^2$$

But

$$\|\sum_{i\in I} w_{i,k}\cdot M_{k,i} - Q_k)\|^2 \le \sum_{i\in I} w_{i,k}\cdot\|M_{k,i}-Q_k\|^2 \quad (8)$$

and we also have (see [10])

$$\langle M_{k,i}-C, M_{k,i}-Q_k\rangle \le 0 \quad (9)$$

Then

$$\left\langle Q_k - C, \sum_{i\in I} w_{i,k}\cdot M_{k,i} - Q_k)\right\rangle \le$$
$$-\sum_{i\in I} w_{i,k}\cdot\|M_{k,i}-Q_k\|^2 = -\sum_{i\in I} w_{i,k}d_{i,k}^2 \quad (10)$$

The point (1) results from (7), (8) and (10). For the mEMOPP algorithm, the set of indices I is replaced by I_k.

The second point is obtained by

$$\begin{aligned}\|Q_{k+1}-Q_k\|^2 &= \lambda_k\|(\sum_{i\in I} w_{i_k}M_{k,i} - Q_k)\|^2 \\ &\le \lambda_k(\sum_{i\in I} w_{i,k}\|M_{k,i}-Q_k)\|^2 \quad (11)\\ &\le \lambda_k\frac{\|Q_k-C\|^2-\|Q_{k+1}-C\|^2}{\lambda_k(2-\lambda_k)} \\ &\le \frac{1}{\epsilon}(\|Q_k-C\|^2-\|Q_{k+1}-C\|^2)\end{aligned}$$

for any point C interior to the polygon $\mathcal{P}$ The third statement derived from

$$\sum_{k=0}^{n}(\|Q_k-C\|^2-\|Q_{k+1}-C\|^2) = \|Q_0-C\|^2-\|Q_n-C\|^2 \le \|Q_0-C\|^2 \quad (12)$$

for any $C \in int(\mathcal{P})$, then

$$\sum_{k=0}^{n}\|Q_{k+1}-Q_k\|^2 \le \frac{1}{\epsilon}dist(Q_0,\mathcal{P})^2 \quad (13)$$

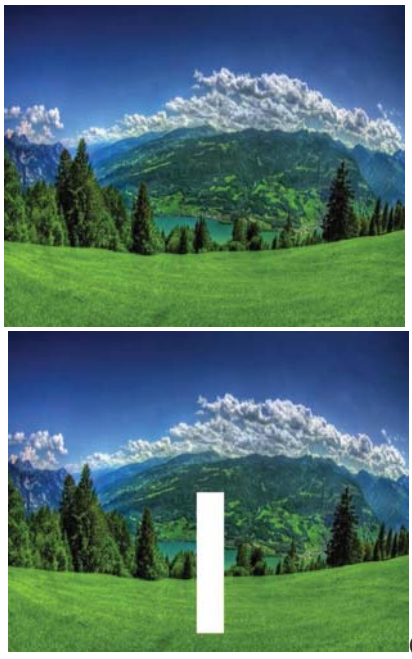
The convergence of Q_i is assured by the Corollary 3.3 from [4].

Note that the convergence depends on the initial distance from Q_0 to the convex polygon and the set $\{\lambda_k\}$. The convergence of the algorithms is not assured for $\lambda_k \ge 2$.

In order to give a more accurate evaluation of the dependence of the algorithms convergence on the relaxation parameters, we supposed $\lambda_k \to \Lambda$ and we proceeded to a series of numerical experiments, described in the next section.

IV. The Comparative Analysis of the Algorithms Implemented on CPU

We first implemented all four PPM, EMOPP, mPPM and mEMOPP algorithms in C++ using Visual Studio 2015 using a BGI emulator for graphic representations. In order to visually test the efficiency and the convergence of the inpainting process, we have chosen an image of 650x512 pixels that contains a rectangular region affected by the loss of information measuring 50x225 pixels (see Figure 1). The algorithms were tested using 10240 points outside the damaged region as starting points, and they where restricted for each starting



(a) (b)

Fig. 1. (a) The initial image, (b) The damaged images

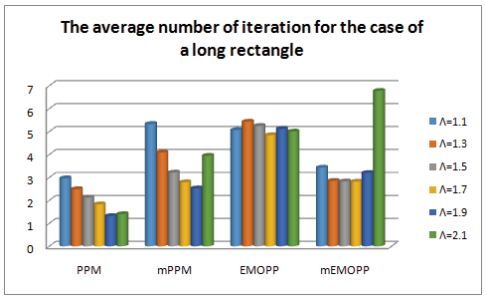


Fig. 2. The average number of iterations

point at maximum 45 iteration, eliminate the possible points where the algorithms do not converge in finite time inside the convex polygon.

We tested the dependence of the convergence of algorithms on the relaxation parameter Λ and the capacity of each algorithm to "fill" the polygonal shape, property that assures that the solutions of the convex feasibility problem, depending on the staring points, cover uniformly the interior of the convex set.

The implemented algorithms produced a list of iterated positions $(Q_{j,k})$ and a short descriptive statistic of number of iterations executed for each starting points. The results for the four algorithms are presented in the next table, Table I, for the relaxation parameters $\lambda_i = \Lambda = 1.1$.

Because the application of the weight factor depends on the position of the points, the number of iterations increases significantly for the case of mPPM algorithm, but not for the mEMOPP algorithm. The convergence of mEMOPP is comparable with the case when the weights are constants.

The same experiments were repeated for each value of $\Lambda \in \{1.1 + 0.2 \cdot i | i = 1, 5\}$. (Note that the convergence of the PPM and EMOPP method are not assured for $\Lambda >= 2$ [9].) The dependence of the number of iterations on the value of Λ is represented in the Figure 2.

In order to test the " efficiency " of the algorithms to "cover" the entire surface of the convex target, we systematised the spatial distribution of the solutions obtained for all the considered algorithms, only for the case of the relaxation parameter $\Lambda = 1.1$, $\Lambda = 1.7$ and $\Lambda = 2.0$ in Table II. To better observe the distribution of the inpaited points we limited the number of starting point to 1000, only for this experiment.

We observed that for small values of Λ, the solutions of the mPPM, EMOPP and mEMOPP are concentrated near the borders of the initial convex polygon. For increased Λ, the filled region becomes larger in the case of these three algorithms. Only for EMOPP and mEMOPP the entire surface of the convex set will be covered, for $\Lambda >= 1.9$, more efficiently in the case of mEMOPP. The classical PPM algorithm produces an inverted copy of the neighborhood of the polygon, centred in the middle of it.

Note that the inpaintig algorithms described in this paper can be used efficiently only for covering small damaged region of an image. The testing area chosen here was relatively large (225x50 pixels).

Finally we analyzed the dependence of the execution time on the relaxation parameter Λ. The results are centralized in the Table III. One remarks that, if the execution time does not depend significantly on the value of relaxation parameter Λ for the case of PPM and mPPM algorithms, the EMOPP and mEMOPP algorithms are faster for $\Lambda = 1.7, 1.9$. The increased values of execution time for $\Lambda = 2.0$ are the result of multiplication of the starting points for which the algorithms do not converge in finite steps, in the case of mEMOPP.

V. Parallelization of the Above Algorithms

For the algorithms presented in the Section III, we propose a parallelization method that involves the transfer of the main iterations from Step 3 to 5 from the CPU execution to GPU, using an OpenCL language implementation. The specificities of OpenCL permit the transfer only of float parameters and simple vectors as pointers, but the iteration from Step 3 can be executed autonomously for as many points as the number of the Graphic Card kernels. The main loop of Step 5 uses at each iteration only the coordinates of the precedent point and the (fixed) vector of the polygon edges P_i, $i = 1..N$. The execution of this Step was distributed between the different kernels of GPU, strongly reducing the execution time for the huge number of points (tested for 10240 points) involved by the algorithms.

The simplified schema of the parallelization method is presented in Figure 3. The main program was implemented in C Sharp. For the damaged image, the user can select a polygonal area that covers the scratch, in our cases the same rectangular area tested in the precedent section, and the execution begins with the random generation of starting points in C Sharp, using a *BuldingPoins()* procedure .

The processed image was temporary stocked in the shared CPU/GPU memory using the methods *(Mem)CL.CreateBuffer* and *CL.EnqueueWriteBuffer()*, and the coordinates of each starting point were transferred to the same shared memory in the beginning of the execution. The coordinate of each one of them becomes input parameters for the iteration procedure (Step 5), entirely implemented in OpenCL using *kernel* functions. The OpenCL procedure modifies the temporary image in the shared memory. At the end, the main program recuperates the inpainted image by the bias of *CL.EnqueueReadBuffer()* method.

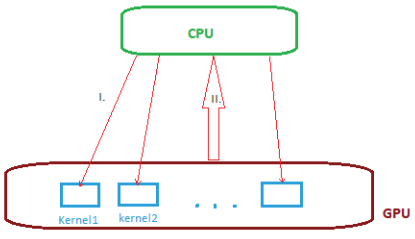


Fig. 3. The simplified parallelization schema (I.) The transfer of random generated starting points, (II.) The transfer of inpainted pixels coordinates

TABLE I
THE DISTRIBUTION OF THE STARTING POINTS SO THAT THE ALGORITHMS STOP IN k STEPS.

Number of iterations	Starting points(%) for PPM	Starting points(%) for mPPM	Starting points(%) for EMOPP	Starting points(%) for mEMOPP
1	13.35	6.32	48.07	48.78
2	20.28	10.38	2.62	2.18
3	46.22	13.47	1.33	7.47
4	16.24	13.30	1.64	12.51
5	0	11.65	1.50	10.89
6	0	12.07	1.44	3.91
7	0	9.21	1.27	0.93
8	0	6.78	1.15	0.15
9	0	4.74	0.93	0.09
>=10	3.91	12.08	40.05	13.08

To comparatively test the GPU implementation of the four algorithms, we used a GeForce Super RTX 2070 graphic card with 2650 cores and a 266-bit memory bus. The comparative execution times, expressed in milliseconds, for different values of the relaxation parameter Λ are presented in Table IV. Comparing also with the execution times for the classical CPU implementation (Table III), the parallel implementation was up to 20 times faster. The obtained inpainted images were very similar to the final image obtained with the C++ implementation.

VI. CONCLUSIONS

The comparative analysis of the graphical implementations of the PPM, EMOPP and their modified versions proposed in Section III allows to identify the way that the variable weight and the relaxation factor Λ influence the number of iterations calculated until reaching the solution of the convex feasibility problem. The results of our experiments are in concordance with the analysis of the effectiveness of projection methods for convex feasibility problems presented in [7] for the PPM and EMOPP.

The graphical comparison allows also to verify the spatial distribution of the solutions, depending on the shape of the convex polygon and the relaxation factor. Even if the methods are not efficient in the recovery of a damaged image by inpainting, in the case of $\lambda \simeq 1.7, 2.1$ the polygon is uniformly covered by the solution points, for EMOPP and mEMOPP.

We choose this type of implementation in order to verify if the analyzed methods for solving the convex feasibility problem can be used to recover a damaged image by inpainting points from the exterior in the affected area. In conclusion, the EMOPP and mEMOPP method can be better used for this purpose, but only for small regions affected and a good choice of relaxation parameter Λ; moreover, the image of the covered area does not reproduce the neighborhood of the convex polygon. All the results of this paper are obtained by numerical experiments, then more theoretical results must be obtained in order to support the presented analysis of the convergence of the four algorithms.

As a final conclusion, the parallelization of the algorithms using a mixed CPU/GPU implementation produces a spectacular incremetation of the execution speed. Parallelization is recommended, although the results are not always spectacular from speedup point of view.

REFERENCES

[1] I.M. Artinescu, L.O. Mafteiu-Scai: *A Scratch Covering Algorithm using Affine Projection Method.* Mathematics and Computer Sciences, *12*(2), 235–246, 2018.

[2] I.M. Artinescu: *A comparative analysis of the convergence regions for different parallel affine projection algorithms.* Stud. Univ. Babes-Bolyai Math., *63*(3), 401–411, 2018.

[3] S. Agmon: *The relaxation method for linear inequalities.* Canadian Journal of Mathematics, *6*, pp. 382–392, 1954.

[4] H.H. Bauschke, J.M. Borwein: *On Projection Algorithms for Solving Convex Feasibility Problems.* (SIAM Review) Society for Industrial and Applied Mathematics Review, *38*(3), 367–426, 1996.

[5] M. Bertalmio, G. Sapiro , V. Caselles, C. Ballester: *Image inpainting. ACM Comput. Graph. (SIGGRAPH 2002).* In SIGGRAPH: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, New Orleans, ACM Press/Addison-Wesley, New York, pp. 417–424,2000.

[6] M. Bertalmio, L. Vese, G. Sapiro, S. Osher: *Simultaneous structure and texture image inpainting.* IEEE Transactions on Image Processing, *12*, pp. 882–889, 2003.

[7] Y. Censor, W. Chen, P. Combettes, R. David, G. Herman: *On the effectiveness of projection methods for convex feasibility problems with linear inequality constraints.* In Computational Optimization and Applications, Kluwer Academic Publishers Norwell, USA, pp. 1065–1088, 2012

[8] G. Cimmino: *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari.* La Ricerca Scientifica, *1*, pp. 326–333, 1938.

[9] P.L. Combettes: *The Convex Feasibility Problem in Image Recovery.* Advances in Imaging and Electron Physics, *95*, pp. 155–270,1996.

[10] P.L. Combettes: *Hilbertian Convex Feasibility Problem: Convergence of Projection Methods.* Applied Mathematics and Optimization, *35*, pp. 311–330, 1997a.

[11] P.L. Combettes:*Convex set theoretic image recovery by extrapolated iterations of parallel subgradient projections.* IEEE Transactions on Image Processing , *6*(4), pp. 493–506, 1997b.

[12] A. Criminis, P. Perez, K. Toyama: *Region Filling and Object Removal by Exemplar-Based Image Inpainting.* Transactions on Image Processing, *13*, pp. 1200–1212, 2004.

[13] P. Gilbert: *Iterative methods for the three-dimensional reconstruction of an object from projections.* Journal of Theoretical Biology, *36*, pp. 105–117, 1972.

[14] R. Gordon, R. Bender, G.T. Herman : *Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography.* Journal of Theoretical Biology, *29*, pp. 471–481,1970.

[15] L.G. Gubin, B.T. Polyak, E.V. Raik: *The method of projections for finding the common point of convex sets.* USSR Computational Mathematics and Mathematical Physics, *7*(6), pp. 1–24, 1967.

[16] Y.I. Merzlyakov: *On a relaxation method of solving systems of linear inequalities.* USSR Computational Mathematics and Mathematical Physics, *2*, pp. 504–510, 1963.

[17] T.S. Motzkin, I.J. Schoenberg: *The relaxation method for linear inequalities.* Canadian Journal of Mathematics, *6*, pp. 393–404, 1954.

[18] M. Ait Rami , U. Helmke, J.B. Moore: *A finite steps algorithm for solving convex feasibility problems.* Journal of Global Optimizations, *38*, pp. 143–160, 2007.

[19] Zhao X.; Hu J.; Yao T. GPU based iterative cone-beam CT reconstruction using empty space skipping technique. *Journal of X-Ray Science and Technology* **2013**, pp. 53-69.

TABLE II
THE FILLED REGIONS DEFINED BY THE SOLUTION OF THE CONVEX FEASIBILITY PROBLEM, FOR DIFFERENT ALGORITHMS AND DIFFERENT VALUES OF Λ

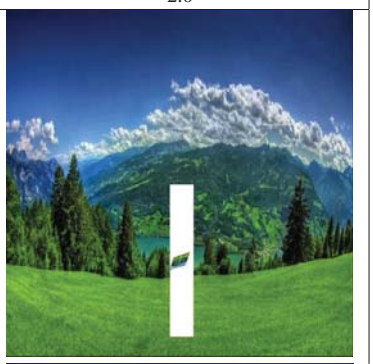
Λ	1.1	1.7	2.0
PPM			
mPPM			
EMOPP			
mEMOPP			

TABLE III
THE COMPARATIVE EXECUTION TIMES FOR DIFFERENT VALUES OF Λ IN THE CASE OF GPU IMPLEMENTATION.

Λ	PPM	mPPM	EMOPP	mEMOPP
1.1	1069	1041	907	694
1.3	1034	1031	786	754
1.5	1026	1001	851	498
1.7	1018	993	757	540
1.9	997	981	656	529
2.0	1065	991	844	749

TABLE IV
THE COMPARATIVE EXECUTION TIMES FOR DIFFERENT VALUES OF Λ IN THE CASE OF GPU IMPLEMENTATION.

Λ	PPM	mPPM	EMOPP	mEMOPP
1.1	48	64	57	67
1.3	46	63	51	58
1.5	49	59	47	55
1.7	51	53	46	56
1.9	53	56	48	64
2.1	67	61	126	71

[20] T. Zheng, X. Cheng, D. Xiaoyun: *A Parallel Depth-Aided Exemplar-Based Inpainting for Real-Time View Synthesis on GPU*. IEEE 10th International Conference on High Performance Computing and Communications and IEEE International Conference on Embedded and Ubiquitous Computing, pp. 1739–1744,2013.