Conformal Net and Evolutionary Computing for Object Detection in Point Clouds

Monserrat Carlos, Carlos Villaseñor, Javier Gomez-Avila, and Nancy Arana-Daniel

University Center of Exact Sciences and Engineering, University of Guadalajara, 1421 Marcelino García Barragán, Guadalajara 44430, Jalisco, Mexico
raquel.carlos2487@alumnos.udg.mx

Abstract. In this paper, we introduce a method for detecting objects in point clouds by creating a hybrid approach that combines a Conformal Neural Network (CNet) with a bio-inspired algorithm called Germinal Center Optimization (GCO). This innovative approach addresses the challenge of detecting objects within 3D point clouds. Our hybrid model demonstrates competitive results, delivering a higher accuracy in object delineation along with a reduced runtime compared to other optimization algorithms such as Particle Swarm Optimization (PSO), Differential Evolution (DE), and Firefly Algorithm (FA). The improved efficiency of our model highlights its potential for applications that require fast and precise object detection in complex 3D environments.

Keywords: Object Detection, Neural Network, Evolutionary Computing

1 Introduction

In recent years, object detection has become a key topic in navigation, both for robotics and autonomous driving. Most object detection systems rely heavily on cameras; however, these present limitations, as they capture images in a 2D plane, which leads to a loss of information about depth and the spatial relationships between multiple objects. Point clouds provide a geometric 3D representation that is useful for different robotics operations like path planning or object avoidance.

The use of light detection and ranging (LiDAR) technology for object detection was not initially widespread, because of its capability but rather due to its high cost. However, as LiDAR has become more affordable, it has gained traction as a viable method for study and application, especially for more common equipment and robotic applications.

This paper is organized as fallow: First in Section **??**, we introduce the Conformal Net, used to estimate the probability of seeing each object. In Section 3, we present the Germinal Center Optimization (GCO) Algorithm. In Section 4, we present the proposed method for object detection. In Section 5, we present our results and comparison with other state-of-the-art algorithms. Finally in section 6 we offer a conclusion.

https://doi.org/10.61728/AE20254902

2 Conformal Net

In this section we explain in general terms the CNet architecture [1] .The kernel trick is a classic machine learning technique used to solve non-linear problems. It is a core concept in Support Vector Machines, Gaussian processes, and kernel machines [2,3]. A real-valued kernel function has the form $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ and it represents a proper inner product in high dimensional space $(m > n)$, as shown in Eq. (1), where $x_1, x_2 \in \mathbb{R}^n$ and $\phi : \mathbb{R}^n \to \mathbb{R}^m$.

$$K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle \tag{1}$$

The proposed method uses a Geometric Algebra conformal mapping as a kernel. Geometric Algebra (GA), also known as Clifford Algebra, is a powerful mathematical framework with numerous applications in physics, robotics, and computer science [4, 5, 7].

The architecture for neural Conformal Net show in Figure 1.

Let us define n as the number of points in a point cloud that represent one object. For experimentation, we set $n = 1024$. The input to the classification network is $(n, 3)$. We implement two layers for preprocessing the point cloud; these layers center and normalize the point cloud.

After that, we map the normalized points to $\mathbb{G}_{4,1}$ using the conformal kernel map. $\mathbb{G}_{4,1}$ has 32 subspaces, resulting in a $(n, 32)$ tensor. We extract the six subspaces from the conformal maps corresponding to $\{e_1, e_2, e_3, e_+, e_-, e_{123+-}\}$ (where e_{123+-} is the pseudoscalar element).

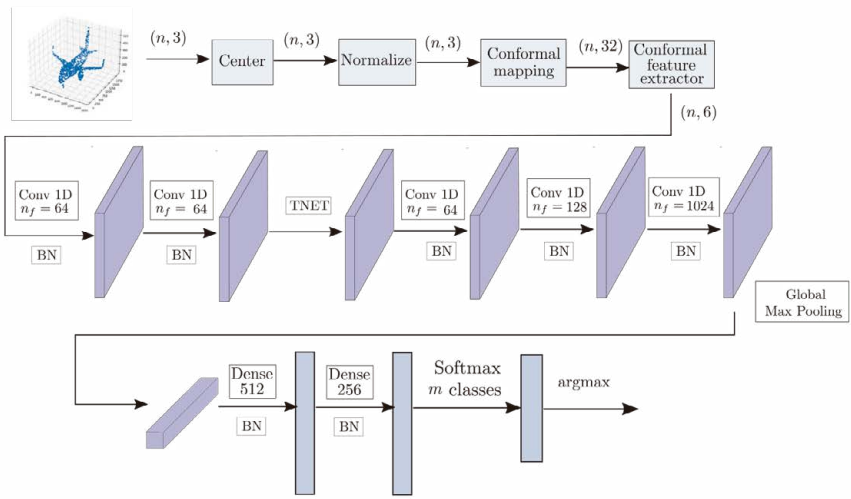

Fig. 1. Architecture CNet

3 Germical Center Optimization

In this section, we provide an overview of the GCO algorithm, which is bio-inspired by the immunology system.

Germinal Centers (GCs) are histologically distinguishable aggregates of B-cells and other immune cells. GCs are formed in the secondary lymphatic nodes when an infection persists for more than seven to ten days [6]. In GCs, the B-cells undergo a competitive process to generate Antibodies (Ab) with high affinity for a specific antigen (Ag). We present the GC process in Figure 2.

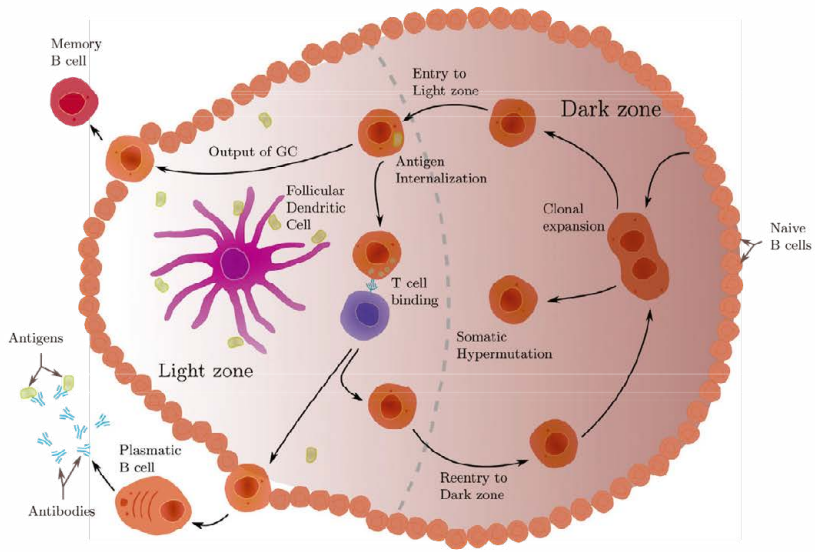

Fig. 2. Germinal Center Reaction

In [8], we presented an algorithm inspired by the GC process, and we have called it Germinal Center Optimization (GCO). We have found applications of GCO in automatic control [9–11], time series forecasting [12], and medical applications [13].

The GCs have two distinct areas: the Dark and Light zones. B-cells cycle through generations between these two zones. This process is mimicked in the Algorithm 1.

Algorithm 1: GCO algorithm

```
Initialize B-cells (B_i)
foreach k ∈ {1, ··· , Iterations} do
    Dark zone process (Algorithm 2)
    Light zone process (Algorithm ??)
end
```

Algorithm 2: Dark Zone process

```
foreach i ∈ {1, ··· , N} do
    if r_l ~ U[0, 100] < L_i of B_i then
        Add one to B_i cells counter
    else
        if Cells in B_i > 1 then
            Rest one to B_i cells counter
        end
    end
    Mutate B_i (Algorithm 3)
end
```

In the Dark Zone, the B-cells B_i proliferate through clonal expansion. We replicate this process in Algorithm 2, where B-cells with high life signal L_i can proliferate more while others will die.

Some B-cells mutate with a process called somatic hyper-mutation. We imitate this with the mutation Algorithm 3. Notice that the mutation algorithm is close to the differential evolution mutation.

4 Object detection with CNet and GCO

In this section explain who was working hybridization which the CNet and GCO for Object detection.

Object detection is the problem of classifying and finding the bounding box of an object; typically, this is a problem formulated for computer vision [14]. In recent years, LiDAR (Light Detection and Ranging) technology has become more affordable and the challenges that its use represents have attracted the attention of the scientific community. We can find new algorithms for object detection in point clouds [15] and others based in neural networks like Localization Network (LocNet) [16]. However, many of these algorithms have high requirements for parallel computing.

We present a hybrid algorithm that uses the CNet and the GCO algorithm to find objects at a point cloud. The B-cells in GCO will represent the parameter of a bounding box, as we show in Eq. (2).

$$B_i = [x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max}]^T \tag{2}$$

Algorithm 3: Mutation algorithm

```
Calculate distribution C
Calculate r1, r2, r3 ~ C
foreach j ∈ {1, ··· , d} do
    if r ~ U[0, 1] < CR then
        M(j) ← B_r1(j) + F * (B_r2(j) − B_r3(j))
    else
        M(j) ← B_i(j)
    end
end
Calculate f(M)
if f(M) < f(B_i) then
    B_i ← M
    Add 10 to L of B_i
    if f(M) < best then
        best ← f(M)
        best_index ← i
    end
end
```

We create random environments consisting of k objects taken from the ModelNet40 dataset and apply random rigid transformations to each object. For example, Figure 3 shows a randomly generated environment with 15 objects.

We generated 300 of these environments, 250 for retraining the CNet and 50 for testing. In Figure 4, we present the training schedule for the CNet.

First, we take a randomly generated environment with $1024o \times 3$ points and generate a random bounding box cut of the point cloud. We will get a $m \times 3$ point cloud. If $m < 400$, we dismiss the sample; if $400 < m < 1024$, we sample with replacement until the size reaches 1024. For $m > 1024$, we sample uniformly to get 1024. We apply this method because the CNet's input is a tensor of 1024×3.

As the point cloud is labeled, we can calculate the proportion of points belonging to each object, giving us a probability distribution. We retrain the CNet to approach this vector using the categorical cross entropy as cost function.

Once the CNet is trained, it can provide the probability distribution of the classes in the cut. The probability allows the algorithm to search in the space to find a target object. In Figure 5, we show the hybrid system for object detection.

We work with a random, unlabeled environment. The particles of the evolutionary algorithm (it could be GCO or another EC algorithm) have information about the bounding boxes used to cut the original point cloud. If the cut is valid, let $v = 1$, and if it is not, let $v = 0$. We resample to fill the point cloud and feed the CNet. The CNet estimates a probability distribution of the objects in the cut.

To guide the particles towards the target, we have developed a custom objective function. Consider $p \sim U[0, 1]^{40}$ as the probability vector output by the

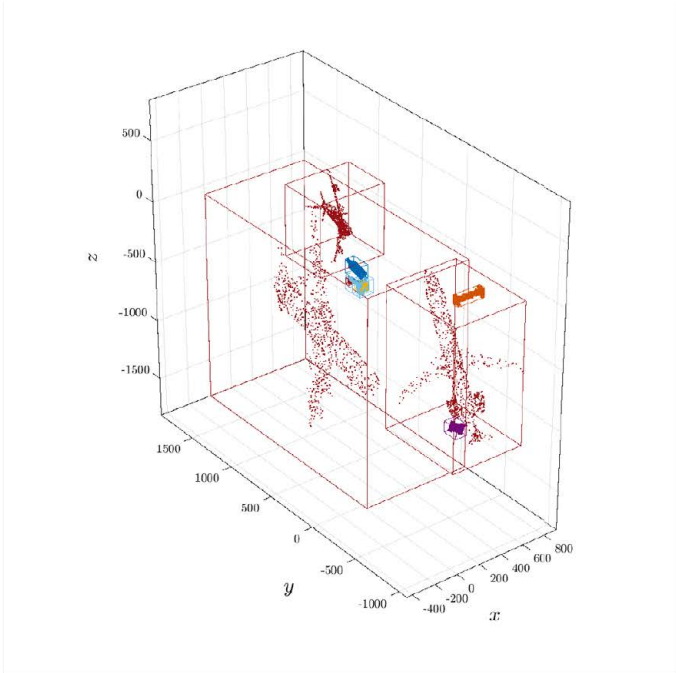

Fig. 3. Example of a randomly generated environment

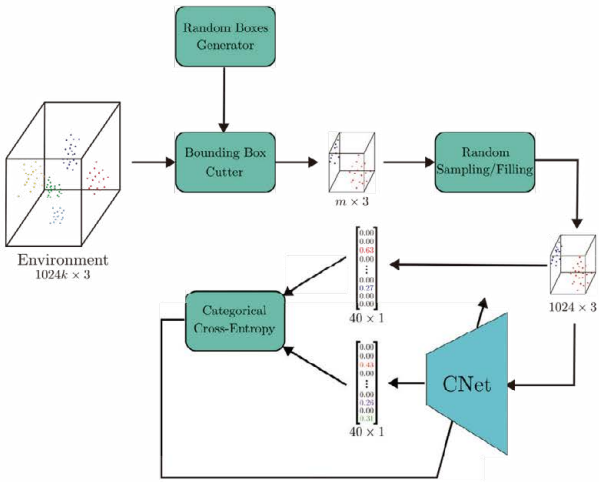

Fig. 4. Training schedule for the CNet for object detection

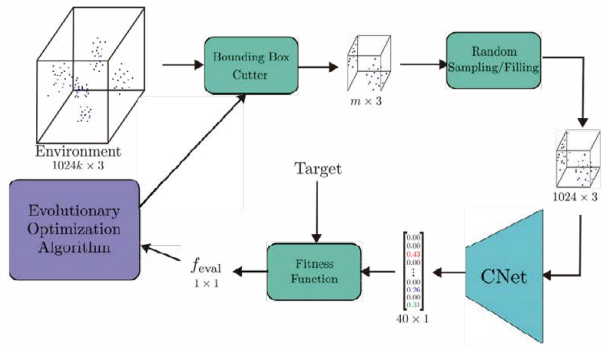

Fig. 5. Evolutionary-CNet detection Scheme.

CNet, where $sum(p) = 1$. Let M be a signed mask that depends on the target object t, and it is defined by

$$M(t)_i = \begin{cases} -1 \; if & i = t \\ 1 \; if \; \text{otherwise} \end{cases} \tag{3}$$

where $i \in \{1, 2, \ldots, 40\}$. To mitigate growing bounding boxes, let us define the range of the environment:

$$E = [ex_{min}, ey_{min}, ez_{min}, ex_{max}, ey_{max}, ez_{max}]^T \tag{4}$$

We can calculate the volumetric ratio between a bounding box and the environment with

$$V = \frac{(x_{max} - x_{min})(y_{max} - y_{min})(z_{max} - z_{min})}{(ex_{max} - ex_{min})(ey_{max} - ey_{min})(ez_{max} - ez_{min})} \tag{5}$$

Then, we can define the fitness function for this problem by

$$f(t, p, v) = \begin{cases} \left[M^T p + 1 + \eta V\right]^2 \; if \; v = 1 \\ \infty \qquad\qquad if \; v = 0 \end{cases} \tag{6}$$

Notice that $V \in [0, 1]$ and $M^T p$ as a minimum of -1 (when p is one in the target position); we compensate this by adding one. Consequently, the minimum of f is 0. We have set $\eta = 0.1$ heuristically.

5 Results and Comparisons

We have compared the GCO algorithm with three classical algorithms: Differential Evolution (DE), Particle Swarm Optimization (PSO), and Firefly Algorithm (FA). We have tested these algorithms with five randomly selected cases and conducted 30 tests for each algorithm.

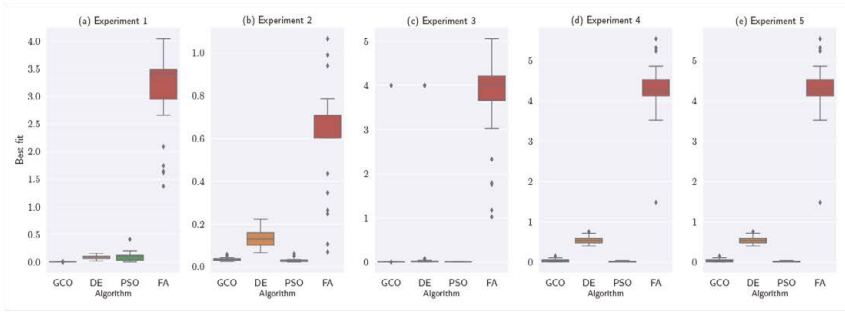

Fig. 6. Results of the detection over the five experiments

Table 2. Time results in **seconds** of the different Evolutionary Algorithms. For Each experiment, we show the mean and standard deviation: mean ± std. **Bold values indicate the best result.**

Experiment	GCO	DE	PSO	FA
1	**0.9565 ± 0.0268**	8.1013 ± 0.2709	1.3396 ± 0.0640	40.0578 ± 0.8207
2	**0.9704 ± 0.0162**	7.7933 ± 0.2146	1.8114 ± 0.0458	39.9203 ± 1.0637
3	**0.8864 ± 0.0381**	7.2119 ± 0.4800	1.2662 ± 0.0426	38.1189 ± 0.5941
4	**0.9950 ± 0.0178**	7.9625 ± 0.1634	1.4072 ± 0.0600	38.1494 ± 0.4109
5	**0.9107 ± 0.0596**	8.0411 ± 0.4220	1.3784 ± 0.2459	36.1997 ± 0.4255

For each algorithm, we used 20 particles and 100 iterations. For GCO and DE, we use the F=1.25 and CR=0.7 parameters. DE is set in its variant DE/rand/1. For PSO, we used an inertial weight of W=0.8 and c1=c2=2. Finally, we use $alpha = 1.0$, $beta = 1$, and $gamma = 0.01$ for the FA. All test software was programmed in Python 3 with an RTX3060 GPU for the neural network programmed in Tensorflow/Keras.

Table 1 shows the results of the experiments as the and standard deviation of the best value in the fitness function. Notice that GCO wins one of the experiments, and PSO wins the other 4. However, the results are very close. DE is close to GCO and PSO, but FA yielded the lowest performance.

Table 1. Results of the different Evolutionary Algorithms. For Each experiment, we show the and standard deviation of best fit: ± std. **Bold values indicate the best result.**

Experiment	GCO	DE	PSO	FA
1	**0.0005 ± 0.0017**	0.0870 ± 0.0305	0.0953 ± 0.0822	3.1449 ± 0.7241
2	0.0358 ± 0.0080	0.1356 ± 0.0423	**0.0307 ± 0.0090**	0.6031 ± 0.2230
3	0.1333 ± 0.7180	0.2823 ± 0.9937	**0.0000 ± 0.0000**	3.6550 ± 0.9925
4	0.0365 ± 0.0361	0.5409 ± 0.0850	**0.0127 ± 0.0123**	4.2689 ± 0.6677
5	0.4173 ± 1.1948	0.0887 ± 0.0848	**0.1095 ± 0.5827**	2.8874 ± 0.2008

In Figure 6, we present the boxplots of the experimental results. Notice that FA has the worst results with high variability. GCO and PSO have similar performances, very close to 0, with small variability, and DE is always slightly behind.

On the other hand, we also present the table that summarizes the fastest running times in seconds for the same experiments. Notice that GCO always achieves the best time, followed by PSO. In third place, we have DE and FA as the last one. These differences in time depend highly on the number of invalid bounding boxes that each algorithm computes.

For visual comparison, in Figure 7, we have selected the first run of each experiment and drawn the target object in the point cloud. We have drawn the best bounding box found by every algorithm in different colors.

Note that we have not drawn the FA bounding box in some cases because it was very far from the others. DE tends to make smaller bounding boxes (except for experiment 3). Qualitative GCO makes better bounding boxes than PSO (except in experiment 5), but they are very close in results.

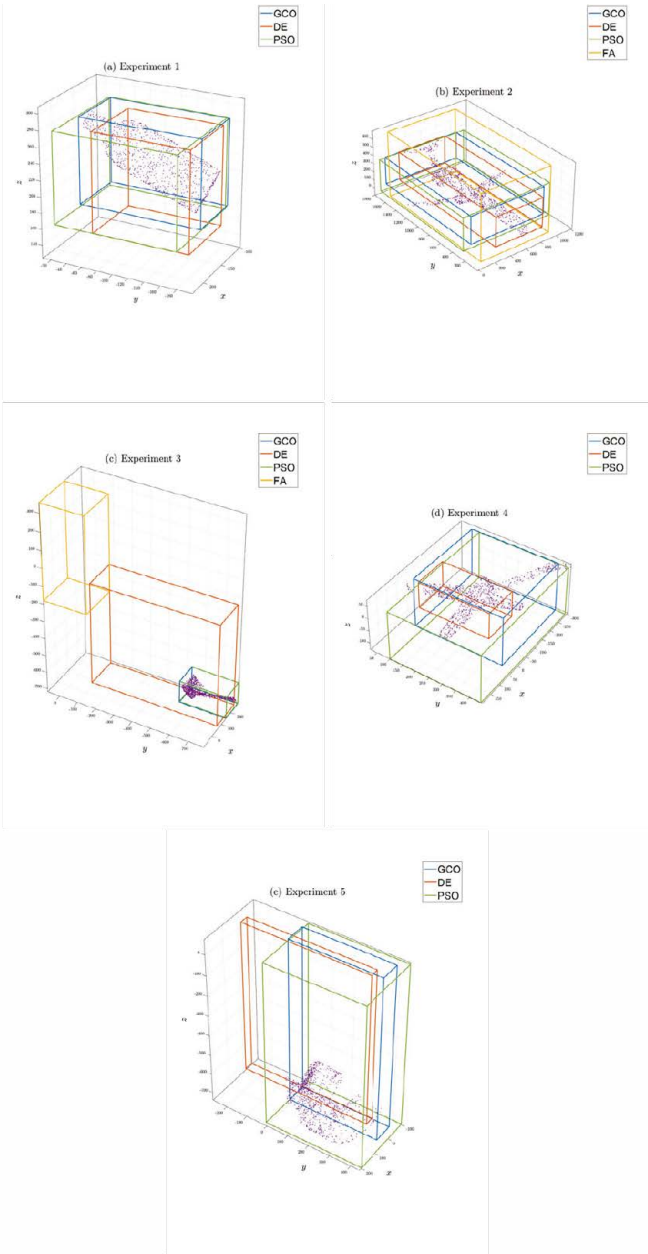

Fig. 7. Comparison of the best solution of the Evolutionary Algorithms for the target object in each experiment.Color scheme: GCO (blue), DE (Orange), PSO (Green), FA (Yellow)

6 Conclusions

We presented a hybrid intelligent system that combines the proposed CNet with the GCO algorithm for object detection in the point cloud. We calculated statistics results against classical state-of-the-art EC algorithms. GCO showed good performance in test, achieving very close minimums with respect to the best results obtained by PSO, but with considerable faster running times due to GCO estimates less invalid bounding boxes than PSO, DE and FA.

References

1. Romero, Fernando., Villaseñor, Carlos , Lopez-Franco, Carlos , Gomez-Avila, Javier , Arana-Daniel, Nancy. (2023). Geometric Convolutional Neural Network for Point Cloud Object Classification. doi:10.1109/ROPEC58757.2023.10409322.
2. Zoppis, Italo and Mauri, Giancarlo and Dondi, Riccardo and others.Encyclopedia of Bioinformatics and Computational Biology. Volume 1,2019; pp. 511-518.
3. Pisner, Derek A and Schnyer, David M. Machine learning,2020; pp. 101-121.
4. Hitzer E, Lavor C, Hildenbrand D. Current Survey of Clifford Geometric Algebra Applications. Math. Meth. Appl. Sci. 47 (2024), pp. 1331–1361, doi:10.1002/mma.8316.
5. Arana-Daniel, N., Villaseñor, C., López-Franco, C., Alanís, A.Y., Valencia-Murillo, R. (2018). Structure from motion using bio-inspired intelligence algorithm and conformal geometric algebra. Intelligent Automation Soft Computing, 24(3), 461-467. doi:10.1080/10798587.2017.1299356.
6. Victora GD, Nussenzweig MC. Germinal Centers. Annu Rev Immunol. 2022 Apr 26;40:413-442. doi:10.1146/annurev-immunol-120419-022408.
7. R. Wang, K. Wang, W. Cao and X. Wang, "Geometric Algebra in Signal and Image Processing: A Survey," in IEEE Access, vol. 7, pp. 156315-156325, 2019, doi:10.1109/ACCESS.2019.2948615.
8. Carlos Villaseñor and Nancy Arana-Daniel and Alma Y. Alanis and Carlos López-Franco and Esteban A. Hernandez-Vargas. Germinal Center Optimization Algorithm 2018. International Journal of Computational Intelligence Systems vol.12, pp.13-27 doi:10.2991/ijcis.2018.25905179.
9. Jorge D. Rios, Carlos Villaseñor, Alma Y. Alanis, Nancy Arana-Daniel, Carlos Lopez-Franco, Germinal Center Optimization Applied to Recurrent High Order Neural Network Observer, IFAC-PapersOnLine, vol 51, Issue 13, 2018, pp 332-337,doi:10.1016/j.ifacol.2018.07.300.
10. Villaseñor, C.; Rios, J.D.; Arana-Daniel, N.; Alanis, A.Y.; Lopez-Franco, C.; Hernandez-Vargas, E.A. Germinal Center Optimization Applied to Neural Inverse Optimal Control for an All-Terrain Tracked Robot. Appl. Sci. 2018, 8, 31. doi:10.3390/app8010031.
11. Carlos Villaseñor, Jorge D. Rios, Nancy Arana-Daniel, Carlos Lopez-Franco, Javier Gomez-Avila, Optimized control and neural observers with germinal center optimization: A review, Annual Reviews in Control,vol. 48,2019, pp. 273-280, ISSN 1367-5788, doi:10.1016/j.arcontrol.2019.07.001.
12. Carlos Villaseñor; Hyperellipsoidal Neural Network Trained With Extended Kalman Filter for Forecasting of Time Series, Artificial Neural Networks for Engineering Applications, 2019,pp. 9-19. doi:10.1016/B978-0-12-818247-5.00011-3.

13. Villaseñor, C., Arana-Daniel, N., Alanis, A.Y., Lopez-Franco, C., Valencia-Murillo, R. (2020). Tracking of Non-rigid Motion in 3D Medical Imaging with Ellipsoidal Mapping and Germinal Center Optimization. In: Castillo, O., Melin, P. (eds) Hybrid Intelligent Systems in Control, Pattern Recognition and Medicine. Studies in Computational Intelligence, vol 827. Springer, Cham. doi:10.1007/978-3-030-34135-0_17.
14. Arkin, E., Yadikar, N., Xu, X. et al. A survey: object detection methods from CNN to transformer. Multimed Tools Appl 82, 21353–21383 (2023). doi:10.1007/s11042-022-13801-3.
15. K. Aoki, K. Koide, S. Oishi, M. Yokozuka, A. Banno and J. Meguro, "3D-BBS: Global Localization for 3D Point Cloud Scan Matching Using Branch-and-Bound Algorithm," 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024, pp. 1796-1802, doi:10.1109/ICRA57147.2024.10610810.
16. H. Yin, L. Tang, X. Ding, Y. Wang and R. Xiong, "LocNet: Global Localization in 3D Point Clouds for Mobile Vehicles," 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 2018, pp. 728-733, doi:10.1109/IVS.2018.8500682.