TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 68, Issue II, June, 2025

A MACHINE LEARNING APPROACH IN PLANAR MECHANISM TRAJECTORY APPROXIMATION

Tiberiu Alexandru ANTAL

Abstract: *Determining the motion path of a point (or link) in a mechanism based on given input motion or designing a mechanism such that a specific point follows a desired trajectory (or motion profile) are key areas in kinematic design for planar and spatial mechanisms. The present work aims to find in the trajectories generated by mechanisms sequences of points that satisfy with a given approximation certain conditions in order to classify them into categories such as line (or arc) with the help of artificial intelligence.*
Key words: *artificial intelligence, mechanisms, trajectory, programing, Python.*

1. INTRODUCTION

In 1968 the book "The Art of Computer Programming, Volume 1: Fundamental Algorithms" written by Donald Knuth formalized the definition of algorithms, setting the precise conditions that an algorithm must satisfy: finiteness, definiteness, Input/Output, effectiveness. "Volume 1" also defined computer science as a mathematical discipline, introduced algorithm analysis and Big-O notation, provided a deep analysis of essential data structures and optimal allocation/manipulation of the resources. In 1976 the book "Algorithms + Data Structures = Programs" written by Niklaus Wirth covered a piece of the fundamental topics in computer programming related to the algorithms, algorithm evaluation metrics and data structures. Based on their researches an algorithm takes input, process it and produces output. In the classical sense, the notion of algorithm can be associated with Newtonian mechanics - Newton's second law ($\vec{F} = m\vec{a}$) - where we have a set of data that characterizes the initial state of the system (input data), a mathematical formula / expression (an algorithm or procedure to deal with the input) through which it evolves (and can form the output) from the initial state to other states. Nothing in the classical definition suggests that input and output can be coupled / associated to teach the algorithm, using different mathematical models, to predict each input and output for (input, output) data pairs that did not participate in the learning process. This algorithm will draw it's intelligence by identifying common habits in the (input, output) data pairs and by transforming the output to an input to perform prediction. The following research tries to use such an approach in the space of trajectories generated by mechanisms in order to identify trajectory sections that deviate with an imposed error from a known trajectory (in this case the line in the plan). If such a sequence is identified, then it extracts de characteristic features of the sequence making an association between the geometrical dimensions of the mechanism, the trajectory and the features discovered in the trajectory subset. Machine Learning (ML) is a about a set of algorithms that become aware of patterns in data and are able to make predictions on future or on new data (based on the experience accumulated from previous data).

2. AI AND TRAJECTORY RECOGNITION IN PLANAR MECHANISM

ML is a subfield of Artificial Intelligence (AI), as not all AI systems use ML (e.g., rule-based

Received: 27.04.25; **Similarities**: 29.05.25: **Reviewed**: 28.05./12.06.25: **Accepted**:12.06.25.

expert systems don't rely on learning from data). The recognition of trajectories generated by planar mechanisms is a wide research area in robotics, kinematics, and control systems. A brief and subject oriented current state of this field includes approaches like:

1. Mathematical and Geometric Approaches [1], [2], [3]:
   - Kinematic Equations: Trajectories are analyzed using closed-form kinematic equations that describe the motion of the mechanism [4], [5].
   - Differential Geometry: Curvature and torsion properties help distinguish different trajectories.
   - Fourier Descriptors: Used to represent complex trajectories in a frequency domain.

2. Computer Vision and Machine Learning [6], [7], [15], [16]:
   - Pattern Recognition: Machine learning algorithms, including neural networks and support vector machines (SVM), are used to classify and recognize trajectories.
   - Deep Learning: Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are applied to analyze time-series data of motion paths.
   - Feature Extraction: Keypoints and descriptors (SIFT, HOG) are used to identify distinct trajectory patterns.

3. Symbolic and Computational Methods [8], [9], [10], [13], [14]:
   - Graph-Based Representation: Trajectories are modeled as graphs for efficient matching and comparison.
   - Optimization Techniques: Genetic algorithms and particle swarm optimization refine trajectory recognition.

As can be seen in the previous classification, ML was found to be useful in analyzing mechanism motion paths. The general term of ML includes three types of paradigms:

1. **Supervised Learning (SL)**, the algorithm is trained on a labeled dataset, meaning each input comes with a corresponding correct output. The model learns to map inputs to outputs based on this labeled data. Characteristics of the model cam be summarized as:
   - Labeled Data - Each input has a known correct answer or output.
   - Direct Feedback - The model is trained with explicit right or wrong answers.
   - Predict Output - Learns to map inputs to expected outputs for future predictions.
2. **Unsupervised Learning (UL)** has no labeled data. Instead, the model identifies hidden structures and patterns in the data. Characteristics of the model cam be summarized as:
   - No Labels - The model is not told what the correct answer is.
   - No Direct Feedback - The model doesn't know if its output is correct or not.
   - Finds Patterns in Data - Groups, clusters, or detects anomalies in datasets.
3. **Reinforcement Learning (RL)**, an agent interacts with an environment and learns through trial and error to maximize rewards. Characteristics of the model cam be summarized as:
   - Decision Process - The agent makes a series of decisions over time.
   - Reward System - Receives rewards or penalties based on actions.
   - Learns Series of Actions - Develops strategies to maximize long-term rewards.

If the outputs are discrete the subcategory of SL is called **classification**, if the output is continuous the SL is called **regression** for predicting continuous outcomes. The following work is an attempt to approximate sections of motion paths based on SL regression analysis. Regression finds a mathematical function that best fits the given data. It tries to minimize the difference between actual values and predicted values using error metrics like Mean Squared Error (MSE). If the regression is linear the mathematical function is $Y = mX + n$ and

models a straight-line relationship between input (X) and output (Y) where m is the slope (relationship strength) and n the intercept (constant).

2.1 PYTHON and the linear regression

NumPy (Numerical Python) is a fundamental library for scientific computing in Python. It provides support for: multi-dimensional arrays (efficient storage & operations), mathematical functions (linear algebra, statistics, etc.), broadcasting (element-wise operations on arrays), and performance optimization (faster than standard Python lists). SciPy (Scientific Python) is an open-source library in Python that builds on NumPy and provides tools for scientific computing, including like: optimization (minimizing functions), interpolation (estimating missing values), signal processing (filtering data), linear algebra (solving equations), statistics and probability (distributions, hypothesis testing), integration and differentiation (solving integrals). `scipy.stats` is a module in SciPy that provides a wide range of statistical functions and probability distributions. In `scipy.stats` a simple way to perform linear regression is using the `linregress()` function. It calculates the best-fit line for two given sets of data points, which can be used to model the relationship between an independent variable x and a dependent variable y. The function signature is:

```
scipy.stats.linregress(x, y)
```

where the parameters are:

- `x`: an array-like, independent variable (input data);
- `y`: an array-like, dependent variable (output data).

Both x and y must have the same length and correspond to the data points we want to model. The function returns a tuple with 5 values:

- `slope`: The slope (m) of the regression line mm.
- `intercept`: The intercept (n) of the regression line.
- `r_value`: The correlation coefficient between x and y. Measures the strength and direction of the linear relationship between the variables (ranges from -1 to +1).
- `p_value`: The two-sided p-value for a hypothesis test where the null hypothesis is that the slope of the regression line is zero (no correlation).
- `std_err`: The standard error of the estimated slope. It provides a measure of how precise the slope estimate is.

A commented code to use the function follows:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress

X = np.arange(0, 10, 1)
y = np.array([])

# print the independent X array
print()
print("x=",X)

# create the Y dependent array using the
## the equation of a line whisch has
## the slope and intercept slightlty
## modifyed using randon numbers
for x in X:
    y1=np.append(y,2*np.random.uniform(0.6,
1.2)*x+np.random.uniform(0.1, 1.1))
    # NumPy arrays are immutable, meaning
    ## np.append() creates a new array
    ## instead of modifying the original one.
    y=y1
print()
print("y=",y)

# Compute linear regression
slope, intercept, r_value, p_value, std_err =
linregress(X, y)

# Print results
print()
print(f"Equation: y = {slope:.2f}X +
{intercept:.2f}")
print(f"R-squared: {r_value**2:.3f}") Measure of how
well the line fits the data
```

```
# Generate predicted y values
y_pred = slope * X + intercept

# Plot actual data points
plt.scatter(X, y, color='blue', label="Actual Data")

# Plot regression line
plt.plot(X, y_pred, color='red', label=f"Regression
Line (y = {slope:.2f}x + {intercept:.2f})")

# Labels and legend
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Linear Regression using scipy")
plt.legend()
plt.show()
```

The result of the code are:

```
x= [0 1 2 3 4 5 6 7 8 9]

y= [ 0.23949386  2.01693542  4.37974392  4.93306005
7.79004034  9.91579324
  8.71725701 17.41970643 12.7219643  18.72986918]

Equation: y = 1.94X + -0.04
R-squared: 0.902
```

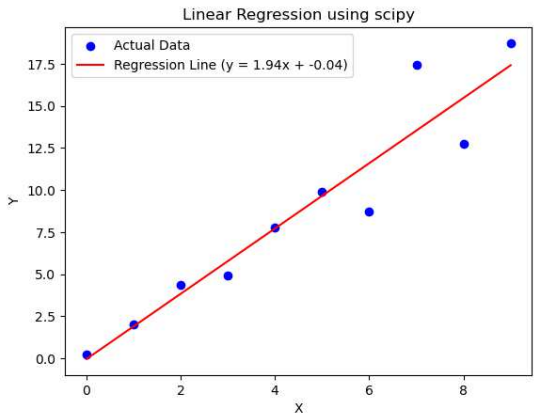

Fig. 1. - Linear regression results of the code for testing `scipy.stats.linregress(x, y)` function.

Using the `X = np.arange(0, 10, 1)` line of code an array of input values is generated starting from 1 to 10 with a step of 1. Then the `for` loop creates an array of dependent points with the slope and intercept slightly modified from a straight line as the blue dots in Figure 1 show. The red line is obtained by regression as well as the equation of the line. At the same time the correlation (`R-squared`) coefficient between x and y is computer having the following meaning:

- `R-squared =1`: the regression model perfectly explains the variance in the data, and the predicted values are exactly the same as the actual values.
- `R-squared =0`: the regression model explains none of the variance in the data. The model does no better than simply predicting the mean of the dependent variable for all points.
- `0< R-squared <1`: the regression model explains a portion of the variance. The closer `R-squared` is to 1, the better the model fits the data.

3. PYTHON IN MECHANISM SIMULATION

Python provides powerful tools for mechanical system simulation, whether it's kinematics, FEA, CFD, rigid body dynamics, or machine learning-based trajectory prediction [12]. It can be used for real-time applications, research, and engineering design. A four-bar linkage consists of four rigid links connected by rotating joints. ML can be used to approximate trajectories or trajectory sections generated by the four-bar linkage mechanism using the scikit-learn (simpler regression-based models). Python implements the OOP programming paradigm that models real-world entities using classes and objects. Python supports encapsulation, inheritance, and polymorphism, making it powerful for modular and reusable code. The OOP model of four-bar linkage mechanism based on the concept of vector [11] is presented as follows. In Figure 2 there are four vectors, each having a name and constructed from the start point to the endpoint. For example, the `v1` vector is named “a” starts at (0,0) and ends at (2-,0). The initial states for the mechanisms are presented and the simulation results are presented in Figure 2, Figure 3 and Figure 4. Vector `v4` has always the start point at the middle of the `v2` vector. By modifying one of the vectors a different four-bar mechanism is obtained that will produce different trajectory. The modifications are presented as a single vector

which changes the original configuration from set (1) to (2) and (3).

```
v1 = Vector2D("a", 0, 0, -2, 0)
v2 = Vector2D("b", -2, 0, 2, 3.5)
v3 = Vector2D("c", 4,0, 2, 3.5)
v4 = Vector2D("d", v2.mid_x, v2.mid_y, 5.5, 6.5)
```
(1)

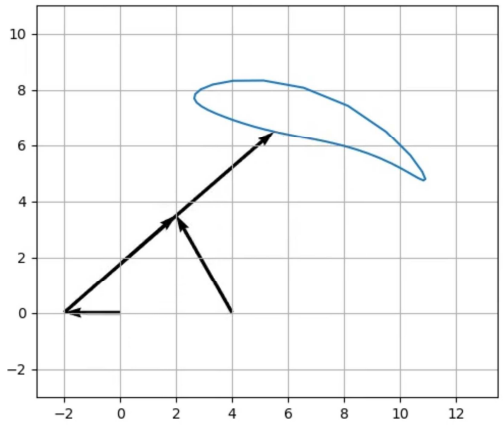

Fig. 2. - Four-bar linkage simulation for the set of (1) vectors.

```
v3 = Vector2D("c", 4,-2, 2, 3.5)
```
(2)

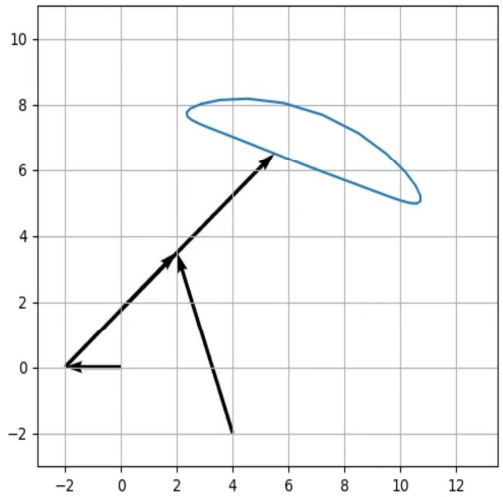

Fig. 3. - Four-bar linkage simulation for the set (2) of vectors.

```
v4 = Vector2D("d", v2.mid_x, v2.mid_y, 2, 6)
```
(3)

The grid of graphical representations allows the evaluation of magnitude, orientation and sense of each vector in the model. If the `v1` vector rotates around the start point the trajectory generated by the endpoint of the `v4` vector will result in a close curve drawn in blue during the simulation on each figure.

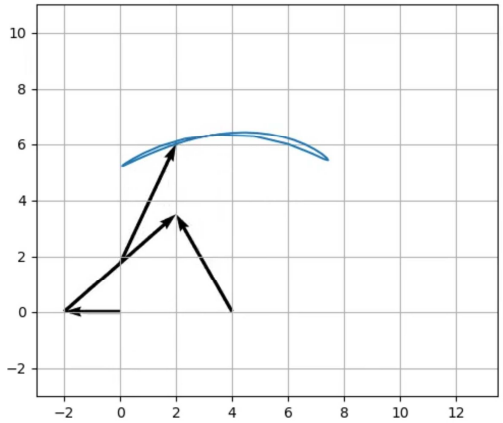

Fig. 4. - Four-bar linkage simulation for the set (3) of vectors.

The simulations from the previous Figure 2 to Figure 4 generated by the code from [11] are processed with the following code and the results are given in Figure 5 to Figure 7. This sequence of code is filtering consecutive points that will participate in the regression set of values by selecting only those points that differ by slope from the previous ones by a given `eps` value.

```
import matplotlib.pyplot as plt1
from scipy import stats

xf, yf = [], []
eps = 0.122

for i in range(0,len(xdata)-1):
  d = delta(xdata[i],ydata[i], xdata[i+1],
ydata[i+1])
  for j in range(i+1, len(xdata)-1):
    d1 = delta(xdata[j],ydata[j], xdata[j+1],
ydata[j+1])
    if abs(d-d1) <= eps:
     xf.append(xdata[j])
      yf.append(ydata[j])
    else:
      break

m, n, r, p, std_err = stats.linregress(xdata, ydata)
mf, nf, rf, pf, std_errf = stats.linregress(xf, yf)

mymodel = list(map(myfunc, xdata))
mymodelf = list(map(myfuncf, xf))
```

```
fig, (ax1, ax2) = plt1.subplots(2,1,
figsize=(50,1.05*40/2), dpi=100)
ax1.scatter(xdata, ydata, color='red')
ax1.scatter(xf, yf, color='green')
ax1.plot(xdata, mymodel, color='red')
ax1.plot(xf, mymodelf,color='green')
ax2.scatter(xf,yf,color='green')
ax2.plot(xf, mymodelf,color='green')
plt1.show()

print(f'Initial line equation is: {m}x+ {n} with std
err: {std_err}')
print(f'Adjusted line equation is: {mf}x+ {nf} with
std err: {std_errf} >> ')
```

The red set of points contain all the points that are used in the simulation; these overlap the trajectories presented in Figure 2 to Figure 5. The green set of points is filtered from the red set by applying the `eps` value to the selection. As seen in Figure 5 to Figure 8 this set narrows the original set the points that fit to the slope condition. The red line is the regression based on all the points, while the green line fits only the filtered points. For each result the code also prints the standard errors and line equations.

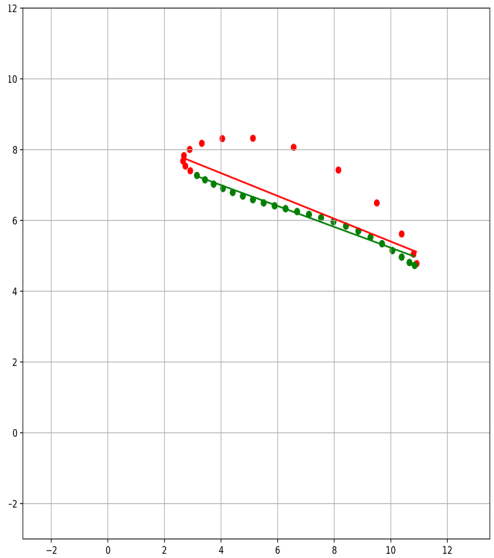

Fig. 5. - Four-bar linkage linear regression for the set (1) without (red) and with filter (green for `eps = 0.122`).

```
eps=0.122
Initial line equation is: -0.3209208605956093x+
8.608079316908118 with std err: 0.03219330490967546
Adjusted line equation is: -0.2950935509526894x+
8.177922296144654 with std err: 0.004522828261481185
```

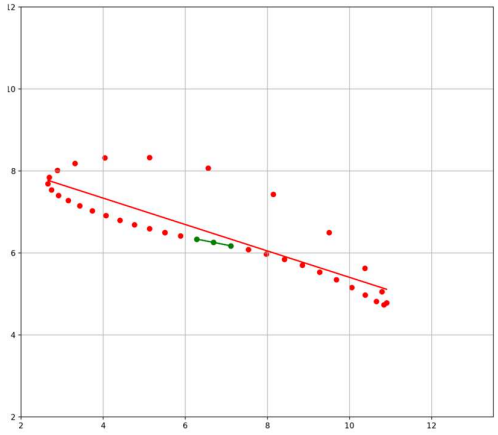

Fig. 6. - Four-bar linkage linear regression for the set (1) without (red) and with filter (green for `eps = 0.05`) .

```
eps = 0.005
Initial line equation is: -0.32210198726971784x+
8.624795574824729 with std err: 0.022679965457873954
Adjusted line equation is: -0.19708300569626602x+
7.5738795427731045 with std err:
0.0004822294740303662
```

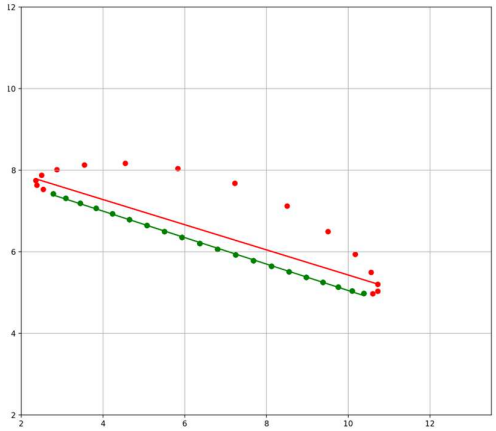

Fig. 7. - Four-bar linkage linear regression for the set (2) without (red) and with filtering (green for `eps = 0.122`).

```
eps=0.122
Initial line equation is: -0.30829236532141124x+
8.51320479109249 with std err: 0.021770586604831168
Adjusted line equation is: -0.3243627143488145x+
8.293453800439082 with std err:
0.00044578900272856784
```

In linear regression, the standard error is an important metric that provides information about the accuracy of the estimated regression coefficients (slope and intercept). In `scipy.linregress()`, the standard error (SE) is calculated for the slope of the regression line, and it gives an idea of how much the estimated slope can vary due to the random noise in the data. It reflects how precise the estimate of the

slope is in relation to the data, and it depends on the spread of the data and the number of data points. The SE of the slope is better if the value is smaller as this means the slope is more precisely estimated.

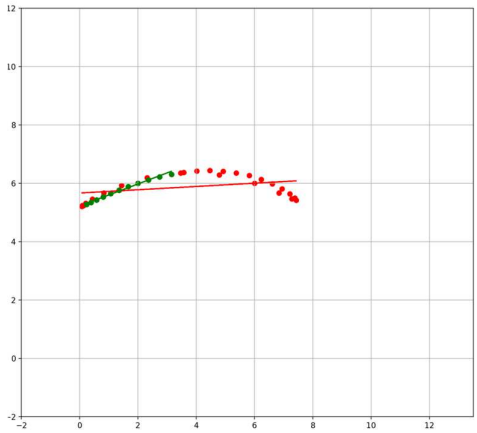

Fig. 8. - Four-bar linkage linear regression for the set (3) without (red) and with filtering (green for `eps = 0.122`).

```
eps=0.122
Initial line equation is: 0.054099974604102236x+
5.68297218438789 with std err: 0.023729057267452593
Adjusted line equation is: 0.3783954932633479x+
5.219579265874449 with std err: 0.006760759745804158
```

4. CONCLUSIONS

As visible in all the presented results (figures and numbers) the linear regression on filter data using the slope criterion will produce good results. For each of the figures showing the algorithm results (Figure 5 to Figure 8) the green line obtained from filtered data is closer to the green set of points and as a numerical quality indicator the "Adjusted line equation" is having a better (smaller) standard error value that the value for the "Initial line equation". With the obtained results a ML classifier system can be trained having the input data (features or the independent variables) organized in a structured table to store 4 vectors (or 8 numbers) and the labels as multi-class responses as {"unknown", "linear", "arc", "circle", "quadratic"}.

5. REFERENCES

[1] Ting, HZ (Ting, Han Zhong); Zaman, MHM (Zaman, Mohd Hairi Mohd); Ibrahim, MF (Ibrahim, Mohd Faisal); Moubark, AM (Moubark, Asraf Mohamed). *Kinematic Analysis for Trajectory Planning of Open-Source 4-DoF Robot Arm*. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, Volume: 12, Issue: 6, Pages: 768-775, 2021, ISSN: 2158-107X. WOS: 000686178900090.

[2] Han Kyul Joo, Tatsuya Yazaki, Masahito Takezawa, Takashi Maekawa. *Differential geometry properties of lines of curvature of parametric surfaces and their visualization*, Graphical Models,Volume 76, Issue 4, 2014, Pages 224-238, ISSN 1524-0703, https://doi.org/10.1016/j.gmod.2014.05.001.

[3] Mark S. Nixon, Alberto S. Aguado, 7 - Object description, Editor(s): Mark S. Nixon, Alberto S. Aguado, *Feature Extraction and Image Processing for Computer Vision (Fourth Edition)*, Academic Press, 2020, Pages 339-398, ISBN 9780128149768, https://doi.org/10.1016/B978-0-12-814976-8.00007-5.

[4] Vaida, C.; Pisla, D.; Schadlbauer, J.; Husty, M.; Plitea, N. Kinematic analysis of an innovative medical parallel robot using study parameters. In *New Trends in Medical and Service Robots. Mechanisms and Machine Science*; Wenger, P., Chevallereau, C., Pisla, D., Bleuler, H., Rodić, A., Eds.; Springer: Cham, Switzerland, 2016; Volume 39.

[5] Plitea, N.; Hesselbach, J.; Vaida, C.; Raatz, A.; Pisla, D.; Budde, C.; Vlad, L.; Burisch, A.; Senner, R. Innovative development of surgical parallel robots. Acta Electron. Mediamira Sci. Cluj Napoca 2007, 4, 201–206.

[6] Asrar G. Alharthi, Salha M. Alzahrani, *Do it the transformer way: A comprehensive review of brain and vision transformers for autism spectrum disorder diagnosis and classification*, Computers in Biology and Medicine, Volume 167, 2023, 107667, ISSN 0010-4825, https://doi.org/10.1016/j.compbiomed.2023.107667.

[7] Rus, G.; Andras, I.; Vaida, C.; Crisan, N.; Gherman, B.; Radu, C.; Tucan, P.; Iakab, S.; Hajjar, N.A.; Pisla, D. *Artificial Intelligence-*

Based Hazard Detection in Robotic-Assisted Single-Incision Oncologic Surgery. Cancers 2023, 15, 3387. https://doi.org/10.3390/cancers15133387

[8] A. Antal, A. Antal, *The use of genetic algorithms for the design of mechatronic transmissions with improved operating conditions*, 3rd International Conference on Human System Interaction, Rzeszow, Poland, 2010, pp. 63-66, doi: 10.1109/HSI.2010.5514588.

[9] ANTAL, T. A., *Planar mechanism synthesis using genetic algorithms*, Acta Technica Napocensis, Series: Applied Mathematics and Mechanics. No. 42, Vol. II, 1999, p.55-60, U. T. PRESS - The Technical University form Cluj-Napoca, Romania, ISSN 1221-5872.

[10] Tucan, P.; Vaida, C.; Horvath, D.; Caprariu, A.; Burz, A.; Gherman, B.; Iakab, S.; Pisla, D. *Design and Experimental Setup of a Robotic Medical Instrument for Brachytherapy in Non-Resectable Liver Tumors*. *Cancers* **2022**, *14*, 5841. https://doi.org/10.3390/cancers14235841

[11] ANTAL, T. A., *Python in the planar four-bar linkage mechanism simulation, Acta Technica Napocensis*, ACTA TECHNICA NAPOCENSIS - Series: APPLIED MATHEMATICS, MECHANICS, and ENGINEERING, [S.l.], v. 68, n. 1, 2025. ISSN 2393–2988.

[12] Vaida, C.; Rus, G.; Tucan, P.; Machado, J.; Pisla, A.; Zima, I.; Birlescu, I.; Pisla, D. *Enhancing Robotic-Assisted Lower Limb Rehabilitation Using Augmented Reality and Serious Gaming*. Appl. Sci. 2024, 14, 12029. https://doi.org/10.3390/app142412029

[13] Morariu-Gligor, R.M., Șerdean, Florina M., Moholea, Iuliana F., *Computer programming in C language with applications in mechanical engineering*, vol. I, Editura Tehnica-Info, Chișinău, 2023, pag. 223.

[14] Șerdean, Florina M., Morariu-Gligor, R.M., Moholea, Iuliana F., *Programming in Matlab with applications in mechanical engineering*, vol. I, Editura Tehnica-Info, Chișinău, 2023, pag. 227.

[15] Deteşan, O.A., *The Path Planning of RTTRR Small-Sized Industrial Robot in a Process of Microprocessor Packing,* Acta Technica Napocensis, Series: Applied Mathematics, Mechanics and Engineering, vol. 60, issue. IV, p. 551-556, Cluj-Napoca, 2017, ISSN 1221-5872

[16] Deteşan, O.A., *The Path Planning of Industrial Robots Using Polynomial Interpolation, with Applications to Fanuc LR-Mate 100iB*, Acta Technica Napocensis, Series: Applied Mathematics and Mechanics, no. 56, vol. IV, p. 665-670, Cluj-Napoca, 2013, ISSN 1221-5872.

6. ACKNOWLEDGEMENT

The completion of this work would not have been possible without the support of TERAPIA S.A who, in 2024, sponsored the “TERAPIA Laboratory” within the Department of Mechanical Systems Engineering with the AI hardware equipment necessary for the research. I would also thank the leadership of the Technical University from Cluj-Napoca to encourage and support us in attending various forms of AI education. And from a teacher to a teacher, special thanks to Andrei Luchici, instructor at ILBAH STUDIO, for his steady guidance, perseverance, and constant encouragements in teaching the "Python Machine Learning & AI" course.

O abordare a învățării automate în aproximarea traiectoriei mecanismelor plane

Determinarea traiectoriei de mișcare a unui punct (element cinematic) dintr-un mecanism sau proiectarea acestuia astfel încât un anumit punct să urmeze o traiectorie dorită (sau un tip de mișcare) sunt domenii cheie în proiectarea cinematică pentru mecanismele plane și spațiale. Lucrarea de față își propune să găsească în traiectoriile generate de mecanisme secvențe de puncte care satisfac cu o aproximare dată anumite condiții pentru a le clasifica în categorii precum linia (sau arcul) cu ajutorul inteligenței artificiale.

ANTAL Tiberiu Alexandru, Professor, dr. eng., Technical University of Cluj-Napoca, Department of Mechanical System Engineering, antaljr@mail.utcluj.ro, 0264-401667, B-dul Muncii, Nr. 103-105, Cluj-Napoca, ROMANIA.