6D SLAM with GPGPU computation

Janusz Będkowski*, **Geert De Cubber****, **Andrzej Masłowski***

*Instytut Automatyki i Robotyki, Politechnika Warszawska, **Royal Military Academy, Brussels, Belgium

Abstract: The main goal was to improve a state of the art 6D SLAM algorithm with a new GPGPU-based implementation of data registration module. Data registration is based on ICP (Iterative Closest Point) algorithm that is fully implemented in the GPU with NVIDIA FERMI architecture. In our research we focus on mobile robot inspection intervention systems applicable in hazardous environments. The goal is to deliver a complete system capable of being used in real life. In this paper we demonstrate our achievements in the field of on line robot localization and mapping. We demonstrated an experiment in real large environment. We compared two strategies of data alignment – simple ICP and ICP using so called meta scan.

Keywords: 6D SLAM, parallel computation

1. Introduction

6D SLAM (Simultaneous Localization and Mapping) algorithm provides robot position (x, y, z, yaw, pitch, roll) for each time frame. In the same time this algorithm stores 3D clouds of points being observations for each robot position. Algorithm is composed of data registration module, loop closing module and map refinement module. Data registration module is responsible for data matching between two different robot observations giving as an output consistent clouds. It is possible to use several techniques for data registration for example point to point [1], point to plane [2] or hybrid approach [3]. Loop closing module finds locations in the map visited by robot at least twice. Map refinement module redistributes an loop-closing error over all scan in the loop. 6D SLAM is very demanding task because of large amount of data to be processed, therefore GPGPU computation brings a possibility to develop algorithm's improvements resulting on-line computation. We believe that in near future such algorithms implemented using dedicated GPGPU or even FPGA boards will drastically improve robotic applications.

2. Related Work

Alignment and merging of two 3D scans, which are obtained from different sensor coordinates, with respect to a reference coordinate system is called 3D registration [4] [5] [6]. Park [7] proposed a real-time approach for 3D registration using GPU, where the registration technique is based on the Iterative Projection Point (IPP) algorithm. The IPP technique is a combination of point-to-plane and point-to-projection registration schemes [8]. Processing time for this approach is about $60ms$ for aligning 2 3D data sets of 76800 points during 30 iterations of the IPP algorithm. Unfortunately, the IPP algorithm has a problem concerning the scalability of the implementation. Fast searching algorithms such as the k-d tree algorithm are usually used to improve the performance of the closest point search [9] [10]. GPU accelerated nearest neighbor search for 3D registration is proposed in work of Qiu [11].

A fast variant of the Iterative Closest Points (ICP) algorithm that registers the 3D scans in a common coordinate system and relocalizes the robot is shown in [12]. Consistent 3D maps are generated using closing loop detection and a global relaxation. The loop closing algorithm detects a loop by registering the last acquired 3D scan with earlier acquired scans, e.g. the first scan. If a registration is possible, the computed error is in a first step divided by the number of 3D scans in the loop and distributed over all scans. The authors reported that the computation time of about 1 min per scan is acceptable, but that further improvement is needed. An algorithm for efficient loop closing and consistent scan alignment that avoids iterative scan matching over all scans is proposed in [13]. Detecting loops in the path is done by using the Euclidean distance between the current and all previous poses (distance threshold of 5 meters), or using GPS data if available. A threshold of minimal number of intermediate scans (e.g. 20) is used to circumvent continuous loop closing within consecutive scans.

Another scan registration approach using 3D-NDT (Normal Distribution Transform) is shown in [14] and automatic appearance-based loop detection from 3D laser data using the Normal Distributions Transform is demonstrated in [15].

Vision has also been used successfully for localization of a mobile robot [16], [17], [18]. A comparison of loop closing techniques in monocular SLAM is show in [19]. Loop closure detection in SLAM by combining visual and spatial appearance was shown in [20]. This approach relies upon matching distinctive signatures of individual local scenes to prompt loop closure. Another advantage is the possibility to enhance robustness of loop closure detection by incorporating heterogeneous sensory observations. The laser scan is divided into smaller but sizeable segments and the complexity of a segment is encoded via entropy. SIFT [21] (Scale Invariant Feature Transform) descriptors are also used to match images. In [22] a methodology combining visual and spatial appearance is shown, whereas in [23], an approach based on multiple map intersection

detection based on visual features appearance is shown. The authors in [24] encode the similarity between all possible pairings of scenes in a similarity matrix and then pose the loop closing problem as the task of extracting statistically significant sequences of similar scenes from this matrix. The analysis (introspection) and decomposition (remediation) of the similarity matrix allows for the reliable detection of loops despite the presence of repetitive and visually ambiguous scenes. Relaxing loop-closing errors in 3D maps based on planar surface patches were shown in [25].

3. SLAM 6D

6D SLAM is composed of following components:

– data registration;

– loop closing;

– map refinement;

The result is consistent 3D clouds of points representing an environment. In following subsections each component will be discussed.

3.1. Point to point ICP

The key concept of the standard ICP algorithm can be summarized in two steps [26]:

1) compute correspondences between the two scans (Nearest Neighbor Search).

2) compute a transformation which minimizes distance between corresponding points.

Iteratively repeating these two steps should result in convergence to the desired transformation. Range images (scans) are defined as model set M where

$$(|M| = N_m) \tag{1}$$

and data set D where

$$(|D| = N_d) \tag{2}$$

The alignment of these two data sets is solved by minimization following cost function:

$$E = (\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{ij} \left\| \mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t}) \right\|^2 \tag{3}$$

w_{ij} is assigned 1 if the i^{th} point of M correspond to the j^{th} point in D. Otherwise w_{ij}=0. $\mathbf{R}$ is the rotation matrix, $\mathbf{t}$ is the translation matrix, $\mathbf{m}_i$ correspond to points from the model set M, $\mathbf{d}_j$ correspond to points from the data set D.

The main idea of using the GPU is to decompose the 3D space into a regular grid of 256 x 256 x 256 buckets. Because we are violating the assumption of full overlap, we are forced to add a maximum matching threshold d_{max} related to the dimension of single bucket. This threshold accounts for the fact that some points will not have any correspondence in the second scan. In most implementations of ICP, the choice of d_{max} represents a trade off between convergence and accuracy. A low value results in bad convergence, a large value causes incorrect correspondences to pull the final alignment away from the correct value. In our implementation the choice of d_{max} is done by a normalization point cloud, which has XYZ coordinates from the interval $< -1, 1 >$. We improved the state of the art algorithm described in [9] by replacing the complex $k-d$ tree data structure to improve the performance of the closest point search. We where focused on a soultion that does not need to download any data structures from CPU instead of 3D data points, therefore we obtained an algorithm fully implemented on GPGPU. All derivations of investigated registration method can be found in [27]. Classic ICP is listed as algorithm 1.

Algorithm 1 Classic ICP

INPUT: Two point clouds A = $\{a_i\}$, B= $\{b_i\}$, an initial transformation T_0
OUTPUT: The correct transformation T, which aligns A and B
$T \leftarrow T_0$
for $iter \leftarrow 0$ to $maxIterations$ **do**
  for $i \leftarrow 0$ to N **do**
    $m_i \leftarrow$ FindClosestPointInA$(T \cdot b_i)$
    if $\|m_i - T \cdot b_i\| \leqslant d_{max}$ **then**
      $w_i \leftarrow 1$
    else
      $w_i \leftarrow 0$
    end if
  end for
  $T \leftarrow \underset{T}{argmin} \left\{ \sum_i w_i \|T \cdot b_i - m_i\|^2 \right\}$
end for

3.2. GPGPU implementation of classic ICP

NVIDIA GPGPUs are fully programmable multi core chips built around an array of processors working in parallel. Details about the GPU architecture can be found in [28] and useful additional programming issues are published in [29]. The GPU is composed of an array of SM multiprocessors, where each of them can launch up to 1024 co-resident concurrent threads.

It should be noticed that available graphics units are in the range from 1 SM up to 30 SMs in high end products. Each single SM contains 8 scalar processors (SP) each with 1024 32-bit registers, the total of 64 KB of register space is available for each SM. Each SM is also equipped with a 16 KB on-chip memory that is characterized by low access latency and high bandwidth.

It is important to realize that all thread management (creation, scheduling, synchronization) is performed in hardware (SM), and overhead is extremely low. The SM multiprocessors work in SIMT scheme (Single Instruction, Multiple Thread), where threads are executed in groups of 32 called *warps*. The CUDA programming model defines the *host* and the *device*. The *Host* executes CPU sequential procedures, whereas the *device* executes parallel programs – *kernels*. A *kernel* works acoording to a SPMD scheme (Single Program, Multiple Data). CUDA gives an advantage of using massively parallel computation for se-veral applications.

The ICP algorithm using CUDA parallel programming is listed as algorithm 2.

Algorithm 2 ICP - parallel computing approach

INPUT: Two point clouds $M = \{m_i\}$, $D = \{d_i\}$, an initial transformation T_0
OUTPUT: The correct transformation T, which aligns M and D
$M_{device} \leftarrow M$
$D_{device} \leftarrow D$
$T_{device} \leftarrow T_0$
for $iter \leftarrow 0$ to $maxIterations$ **do**
  for $i \leftarrow 0$ to N {in parallel} **do**
    $m_i \leftarrow$ FindClosestPointInM $(T_{device} \cdot d_i)$ {using regular grid decomposition}
    if $foundClosestPointInNeighboringBuckets$ **then**
      $w_i \leftarrow 1$
    else
      $w_i \leftarrow 0$
    end if
  end for
  $T_{device} \leftarrow \underset{T_{device}}{argmin} \left\{ \sum_i w_i \left\| T \cdot d_i - m_i \right\|^2 \right\}$ {calculation T←R,t with SVD}
end for
$M \leftarrow M_{device}$
$D \leftarrow D_{device}$
$T \leftarrow T_{device}$

3.3. Loop closing

Loop closing occurs when robot is visiting the same place a second time. It is very important to realize that the robot is not able to perform exactly the same path, therefore a displacement between loop closing robot positions occurs. In our opinion, this displacement should by minimized to increase the efficiency of the loop closing method. This can be done using the strategy of robot motion, especially when it traverses narrow paths. Even when the loop closing method is guarantying heading and displacement invariance, the compromise has to be taken into the consideration. For this reason we define loop closing as a location of the robot that it visited twice with the similar region of observation. The main concept of classing loop closing approach is to find robot locations neighboring the current robot position in the path-graph obtained by ICP odometry correction. The decision concerning loop closing is performed based on the best matching using also ICP method. Because the classic method needs to perform ICP for all robot locations in the neighborhood, the processing time is increasing with its radius. The loop closing transformation matrix is found from best matching.

3.4. Map refinement

Map refinement is done by distributing the loop closing error over all nodes in the loop. Each node stores an information concerning local 3D map – robot observation and robot position (x, y, z, yaw, pitch, roll). When loop closing occurs we obtain the consistent map.

4. Experiments

In the experiments commercially available robot PIONEER 3AT was used (fig. 1). Robot is equipped with 3D laser measurement system 3DLSN based on rotated SICK LMS 200. The robot was acquiring observations in a stop-scan fashion with a one meter step. Each scan contains 361 (horizontal) x 498 (vertical) data points. An environment where robot acquired a set of 3D scans is shown on figure 2; a map composed of several aligned clouds of points is shown in fig. 3. Figures 4 and 5 show an important comparison between two different strategies of data registration. Green line corresponds to robot trajectory acquired by odometry. Red line corresponds to robot trajectory corrected using iteratively ICP algorithm. Blue line corresponds to robot trajectory corrected using ICP algorithm aligning iteratively next scan with previously build so called meta scan. Meta scan stores few scans from previous robot locations, therefore an odometry error is corrected better than using only iteratively ICP. Fig. 6 shows the result of loop closing and map refinement modules.

Fig. 1. Robot PIONEER 3AT equipped with 3D laser measurement system
Rys. 1. Robot PIONEER 3AT z laserowym systemem pomiarowym 3D

5. Conclusion

The robot was acquiring observations in a stop-scan fashion with a one meter step. The goal was to align iteratively all scans, therefore the odometry error was decreased. We set as a benchmark for obtaining a satisfying result of odometry correction, performed with different strategies of data registration, that the loop closing procedure can be performed. In our approach, all strategies can be used as a component of a 6D SLAM processing pipeline, but we recommend using meta scan strategy. The main observation is that both strategies did correction of robot path derived from odometry with gyroscopic correction system.

However, we want to emphasize the fact that the accuracy of ICP strongly depends on various of factors. The accuracy of ICP point to point depends more on the amount of points in the bucket (it can be tuned by normalization of the point cloud) rather than number of iterations. The average amount of points in buckets should be more than 10 and less than 100 to obtain accurate alignment with on-line

Fig. 2. An environment during an eperiment (Royal Military Academy, Brussels, Belgium). Robot's path is shown as red line
Rys. 2. Środowisko robota podczas eksperymentu (Royal Military Academy, Bruksela, Belgia). Trajektoria robota została zaznaczona linią koloru czerwonego

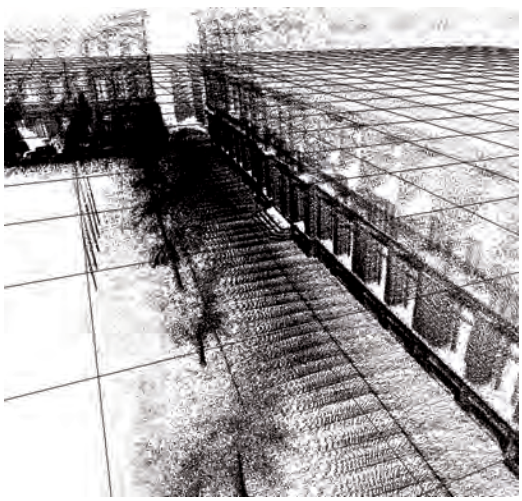

Fig. 3. Map composed of several aligned clouds of points
Rys. 3. Mapa składająca się z kilku dopasowanych chmur punktów

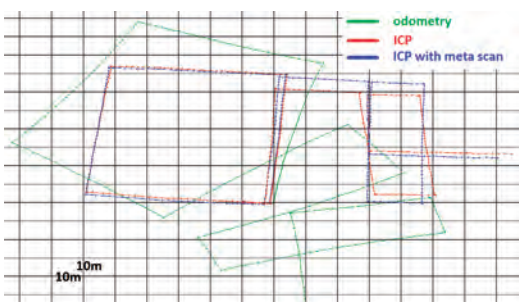

Fig. 4. Comparison between two different strategies of data registration
Rys. 4. Porównanie dwóch różnych strategii rejestracji danych

computation (300 ms for 30 iterations). We observed that 30 iterations guarantee satisfying result.

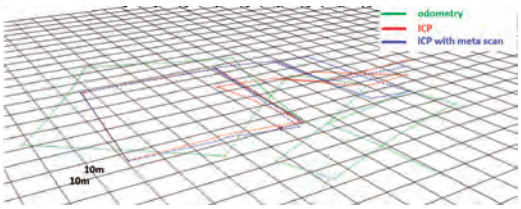

Fig. 5. Comparison between two different strategies of data registration (perspective view)
Rys. 5. Porównanie dwóch różnych strategii rejestracji danych (widok perspektywistyczny)

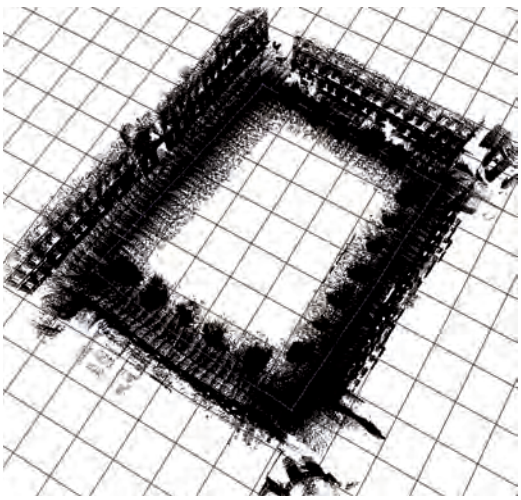

Fig. 6. A result of loop closing and map refinement modules
Rys. 6. Rezultat modułu zamykania pętli oraz modułu poprawy mapy.

The contribution of this paper is the improvement of a state of the art 6D SLAM approach by using parallel computation. Two strategies of data registration were compared. Empirical evaluation performed on a large data set showed the on-line capability of parallel computation.

Acknowledgment

This work is performed during postdoctoral scholarship (CAS/19/POKL 15.05.2011-15.11.2011) in Royal Military Academy, Unmanned Ground Vehicle Center, Brussels, Belgium, funded by Center for Advanced Studies, Warsaw University of Technology (project: "Priorytet IV Programu Operacyjnego Kapitał Ludzki z Europejskiego Funduszu Społecznego").

Bibliography

1. Bedkowski J. (2011): *Data registration module - a component of semantic simulation engine*, [in:] *Proceedings of 5th European Conference on Mobile Robots ECMR 2011*, Orebro, Sweden, 133–138.
2. Bedkowski J., Maslowski A. (2011): *GPGPU implementation of On-Line point to plane 3D data registration*, [in:] *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics* Volume 2, 17-19 July 2011, Bandung, Indonesia, 931–936.
3. Bedkowski J. (2011): *Parallel implementation of hybridICP data registration*, "Elektronika" (8), 114–118.

4. Huber D., Hebert M. (2003): *Fully automatic registration of multiple 3D data sets*, "Image and Vision Computing" 21(1), 637–650.
5. Fitzgibbon A. W. (2001): *Robust registration of 2D and 3D point sets*, [in:] *British Machine Vision Conference*, 411–420.
6. Magnusson M., Duckett T. (2005): *A Comparison of 3D Registration Algorithms for Autonomous Underground Mining Vehicles*, [in:] *Proc. ECMR*, 86–91.
7. Park S.-Y., Choi S.-I., Kim J., Chae J. (2010): *Real-time 3D registration using GPU*, "Machine Vision and Applications", 10.1007/s00138-010-0282-z, 1–14.
8. Park S.-Y., Subbarao M. (2003): *An accurate and fast point-to-plane registration technique*, "Pattern Recogn. Lett." 24, 2967–2976.
9. Nuchter A., Lingemann K., Hertzberg J. (2007): *Cached k-d tree search for ICP algorithms*, [in:] *Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling*, IEEE Computer Society, Washington, DC, USA, 419–426.
10. Rusinkiewicz S., Levoy M. (2001): *Efficient Variants of the ICP Algorithm*, [in:] *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling (3DIM)*.
11. Qiu D., May S., Nuchter A. (2009): *GPU-Accelerated Nearest Neighbor Search for 3D Registration*, [in:] *Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems*, ICVS '09, Springer-Verlag Berlin/Heidelberg, 194–203.
12. Surmann H., Nuchter A., Lingemann K., Hertzberg J. (2004): *6D SLAM - Preliminary report on closing the loop in six dimensions*, [in:] *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles* (IAV'04).
13. Sprickerhof J., Nuchter A., Lingemann K., Hertzberg J. (2009): *An Explicit Loop Closing Technique for 6D SLAM*, [in:] *4th European Conference on Mobile Robots*, September 23-25, 2009, Mlini/Dubrovnik, Croatia, 229-234.
14. Magnusson M., Duckett T., Lilienthal A. J. (2007): *3D Scan Registration for Autonomous Mining Vehicles*, "Journal of Field Robotics" 24(10), 803–827.
15. Magnusson M., Andreasson H., Nüchter A., Lilienthal A. J. (2009): *Automatic Appearance-Based Loop Detection from 3D Laser Data Using the Normal Distributions Transform*, "Journal of Field Robotics" 26(11–12), 892–914.
16. Newman P., Cole D., Ho K. L. (2006): *Outdoor SLAM using Visual Appearance and Laser Ranging*, [in:] *Proceedings of the IEEE International Conference on Robotics and Automation* (ICRA).
17. Newman P., Ho K. L. (2005): *SLAM – Loop Closing with Visually Salient Features*, [in:] *IEEE International Conference on Robotics and Automation*, 18-22 April.
18. Cummins M., Newman P. (2008): *FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance*, "The International Journal of Robotics Research" 27(6), 647–665.
19. Williams B., Cummins M., Neira J., Newman P., Reid I., Tardos J. (2009): *A comparison of loop closing techniques in monocular SLAM*, [in:] *Robotics and Autonomous Systems*.
20. Ho K. L., Newman P. M. (2006): *Loop closure detection in SLAM by combining visual and spatial appearance*, [in:] *Robotics and Autonomous Systems*, 740–749.
21. Lowe D. G. (2004): *Distinctive Image Features from Scale Invariant Keypoints*, "Int. J. Comput. Vision" 60, 91–110.
22. Leong K., Newman P. (2005): *Combining Visual and Spatial Appearance for Loop Closure Detection in SLAM*, [in:] *2nd European Conference on Mobile Robots* (ECMR).
23. Ho K. L., Newman P. (2005): *Multiple Map Intersection Detection using Visual Appearance*, [in:] *3rd International Conference on Computational Intelligence*, "Robotics and Autonomous Systems", Singapore.
24. Ho K. L., Newman P. (2007): *Detecting Loop Closure with Scene Sequences*, "Int. J. Comput. Vision" 74, 261–286.
25. Kaustubh Pathak N. V.Max Pfingsthorn, Birk A. (2009): *Relaxing Loop-Closing Errors in 3D Maps Based on Planar Surface Patches*, [in:] *14th International Conference on Advanced Robotics* (ICAR), IEEE Press.
26. Segal A., Haehnel D., Thrun S. (2009): *Generalized-ICP*, [in:] *Proceedings of Robotics: Science and Systems*, Seattle, USA.
27. Nüchter A., Hertzberg J. (2008): *Towards semantic maps for mobile robots*, "Robot. Auton. Syst." 56(11), 915–926.
28. (2010a): *NVIDIA CUDA C Programming Guide 3.2*, http://www.nvidia.com/cuda.
29. (2010b): *CUDA C Best Practices Guide 3.2*, http://www.nvidia.com/cuda.

6D SLAM wykorzystujacy obliczenia GPGPU

Streszczenie: Głównym celem jest artykułu jest usprawnienie algorytmu 6D SLAM za pomocą implementacji modułu rejestracji danych wykorzystującą obliczenia równoległe. Moduł rejestracji danych jest oparty o algorytm ICP (ang. *Iterative Closest Point*), który został w pełni zaimplementowany w architekturze GPU NVIDIA FERMI. W naszych badaniach koncentrujemy się na mobilnych systemach robotycznych inspekcyjno-interwencyjnych dedykowanych do pracy w niebezpiecznym środowisku. Celem jest opracowanie kompletnego systemu, który może być wykorzystany w realnej aplikacji. W tym artykule przedstawiamy nasze rezultaty w zakresie lokalizacji i budowy mapy w trybie on-line. Przedstawiamy eksperyment w rzeczywistym, rozległym środowisku. Zostały porównane dwie strategie dopasowywania danych, klasyczna oraz wykorzystującą tzw. meta scan.

Słowa kluczowe: 6D SLAM, obliczenia równoległe

Janusz Będkowski, PhD

PhD in Automation and Robotics, Assistant Professor at Institute of Automatic Control and Robotics – Warsaw University of Technology; adjunct at Industrial Research Institute for Automation and Measurements PIAP as well as Institute of Mathematical Machines.

The scope of research: inspection and intervention robot systems, semantic mapping, virtual training with AR techniques.

e-mail: januszbedkowski@gmail.com

Geert De Cubber, PhD

Geert De Cubber received his PhD in engineering from the Vrije Universiteit Brussels (VUB), Belgium and the Royal Military Academy (RMA), Belgium. He is currently head of the research activities of the Unmanned Vehicle Centre of the Belgian Royal Military Academy. His main research interest goes out to computer vision algorithms which can be used by intelligent mobile robots.

e-mail: geert.de.cubber@rma.ac.be

Andrzej Masłowski, PhD, Full Prof.

Full Professor in Automation and Robotics in Institute of Automatic Control and Robotics – Warsaw University of Technology as well as Air Force Academy in Dęblin. The scope of research: intelligent mobile systems, multirobot systems for RISE applications, e-Training for advanced mobile systems. Member of IFIP, IFAC, IMACS, IMEKO TC-17 Measurement and Control in Robotics. Since 2006 Representative of Poland in International Advanced Robotics Programme.

e-mail: a.maslowski@mchtr.pw.edu.pl