Development of a Teaching Methodology of Complex Electromechatronic and Robotic Systems

Andrejs Stupāns
Institute of industrial Electronics and Electrical engineering
Riga Technical University
Riga, Latvia
andrejs.stupans@rtu.lv

Skaidrīte Kriviša
Institute of industrial Electronics and Electrical engineering
Riga Technical University
Riga, Latvia
skaidrite.krivisa@rtu.lv

Armands Šenfelds
Institute of industrial Electronics and Electrical engineering
Riga Technical University
Riga, Latvia
armands.senfelds@rtu.lv

Leonīds Ribickis
Institute of industrial Electronics and Electrical engineering
Riga Technical University
Riga, Latvia
leonids.ribickis@rtu.lv

2021 IEEE 19th International Power Electronics and Motion Control Conference (PEMC) | 978-1-7281-5660-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/PEMC48073.2021.9432500

***Abstract*—The paper describes a process of developing a driving simulator based on KUKA KR600 robot for educational purpose. The main idea is to introduce students to multiple topics – electromechatronics, robotics, inverse kinematics, motion cueing, washout filters. It is cross-disciplinary project. First, students generate car acceleration using Unity game engine's built-in physics. Then they design motion cueing algorithm in Matlab Simulink and transform the accelerations into robot motions. As a result, students check robot motion cues in virtual environment using MOBILE software. Last step is to test generated motions on a real robot manipulator.**

Keywords—filters, motion control, robot kinematics, manipulators, simulation, modeling

I. Introduction

This project is designed to introduce students to main topics of electromechatronics, control theory and robotics by implementing motion cueing algorithm to KUKA KR600 industrial robot manipulator. It is unique educational opportunity since multiple scientific fields are covered. It is cross-disciplinary project with few initial skill requirements. Students are required to have basic knowledge of dynamics and kinematics, application of rotation matrices, Euler/Bryant angles to calculate vehicle accelerations. Also understanding transfer functions and math operations in matrix form. After completing the project students will improve their knowledge in robotics, washout filters, inverse kinematics. Also, students will learn how to use specific software to complete the task. They will be introduced to MATLAB programming and creation of Simulink models, working with Unity game engine, programming in C++ and C#.

The laboratory stand consists of KUKA KR600 industrial robot manipulator with a passenger cabin at the end-effector, operator's computer running Windows, and KUKA KR C4 controller.

The project is designed for online education. All the steps can be completed using home computer, and all tests can be made in virtual environment. The only part that requires presence is final testing on the laboratory stand. This step is optional, so it is possible to teach students fully online.

This article will introduce and describe one of motion cueing algorithms. Motion cueing algorithms are widely used in driving simulators. Main purpose of the algorithm is to transform motions of real vehicle in form that simulator hardware could reproduce into accelerations felt by driver without exceeding motion limits [1], [2].

The main purpose of simulation is to represent motions of real and expensive vehicle on a robotic simulator platform so that the user has a proper sense of real vehicle motions. Working on a vehicle could cause costly hardware damage and health threats to user in case of mistakes. Simulators provide user safety from any kind of crash, damages and unexpected expenses. As a result, threats are eliminated, and user gets experience of possible consequences of his own actions.

There are many different types of driving simulators with their own specifications, performance and motion possibilities. Each simulator has its own advantages and disadvantages. Stewart platform [3] is cheap to build but it has little linear motion possibilities. Industrial robot-based manipulator has much more motion space, but it is more expensive in operation.

First the way of generating car accelerations using Unity game engine will be described in section II. Then accelerations should be transformed into simulator motion cues. The process of creating Simulink model and adjusting parameters of motion cueing algorithm is described in section III. Previously generated motion cues are tested in virtual environment in section IV. If virtual robot moves properly, no collisions detected, and no limits are exceeded, then motion cues can be tested on a real robot in section V. Future tasks and challenges, such as algorithm improvements and real-time simulation implementation is described in section VI.

II. Trajectory Generation

First step is to generate car's accelerations from given trajectory. S-curve trajectory is shown in Fig. 1. The initial

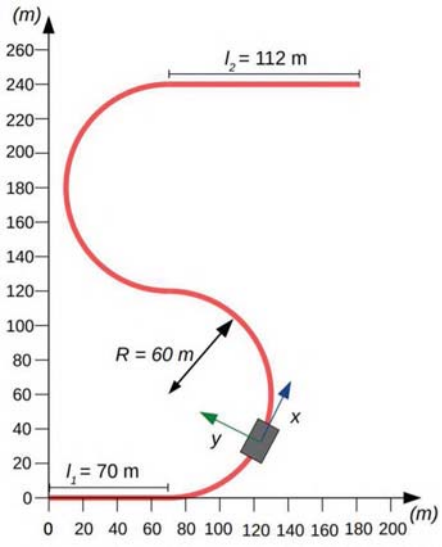


Fig. 1. Car's S-curve trajectory

parameters are curvature $R = 60\ m$, $l_1 = 70\ m$, $l_2 = 112\ m$ and car velocity $v = 14\ m/s$. Only acceleration by y axis is considered in this project. This way the only acceleration that affects a car is centripetal acceleration.

Unity game engine provides a good way to visualize given trajectory and has powerful programming environment. Creating a car as a game object and programming velocities to it results in a car following given trajectory. Then accelerations are calculated using built-in physics of Unity and exported in Excel file. The graph is shown in Fig. 2.

III. Motion Cueing

A. Classic Washout Filter

Motion cueing algorithm transforms given car motion cues into simulator motion cues. The transformation is required due to limited motion space of the robot. There are two types of motion cues. These are high frequency and low frequency motions [1]. High frequency rotational and translational motions are represented by robot's short rotations and translations. After these motions, the robot slowly returns to its initial position. This way the robot has optimal space for the next movement.

When the car enters a turn, driver experiences sustained acceleration. It cannot be reproduced the exact same way as high frequency accelerations, because of restricted workspace of the robot. To achieve driver's perception of accelerations, tilt coordination is used [2], [4]. Tilt of the cabin tricks driver's vestibular system. It uses part of gravitational acceleration to simulate sustained accelerations as shown in Fig. 3.

Implementing motion cueing algorithm using Classic Washout Filter (CWF) [1] is effective method to learn the field of driving simulators, because of the clear and predictive results the algorithm gives. To make results much more readable for students, the CWF is simplified, using only acceleration by y axis and ignoring angular velocities ω as input data [2]. The output data of the algorithm are translation S by y and z axis and rotation angle φ about x axis. The scheme of simplified CWF is shown in Fig. 4.

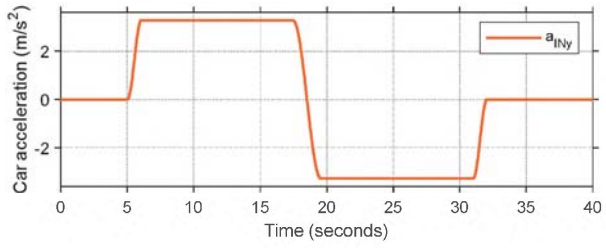


Fig. 2. Y axis component of car's acceleration

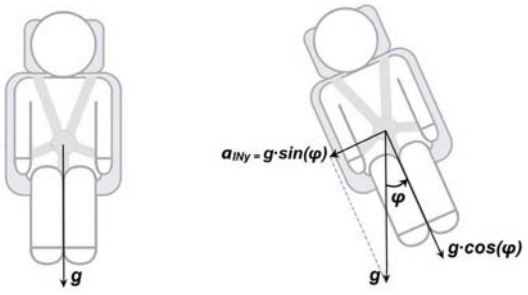


Fig. 3. Tilt coordination implemented to simulate car's lateral acceleration a_{INy}

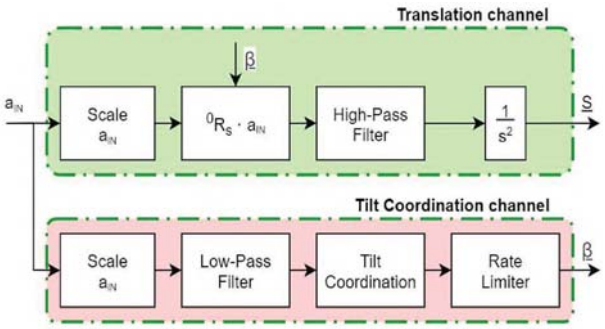


Fig. 4. Simplified scheme of Classic Washout Filter

B. Translation channel

Input acceleration a_{IN} (1) is a 3x1 vector with components of x, y and z axes. Due to algorithm simplifications a_{INx} and a_{INz} values are 0.

$$\underline{a}_{IN} = [a_{INx}\ ,\ a_{INy}\ ,\ a_{INz}]^T \qquad (1)$$

In translation channel input accelerations are scaled. It prevents translational movements from exceeding the robot's motion limits. Scaling gain parameter is chosen considering the properties of a specific simulator. If it was a Stewart platform, then translational movement would be limited to *20-30 cm* and optimal gain value is 0,5 [1], [3]. In this case KUKA robot has much more translational motion possibilities. To take advantage of larger motion space the scale factor can be set to greater value. This value is tuned empirically. In this project gain value is set to 2.

Then scaled accelerations are transformed from driver seat frame K_S to robot base frame K_0 with rotation matrix ${}^0R_S$ using (2), where $Rot(a,b)$ is a rotation about axis a with angle b. These angles are computed in tilt coordination channel. Reference frames are shown in Fig. 5 where Y_S axis points towards the driver's left side.

$${}^0R_s = Rot(Z,\psi) \cdot Rot(Y,\theta) \cdot Rot(X,\varphi) \qquad (2)$$

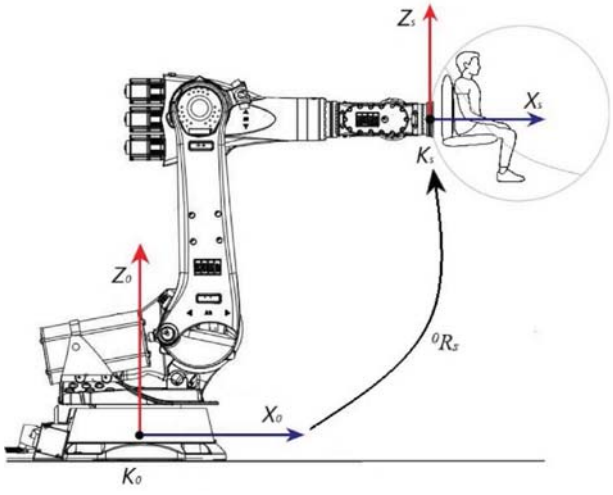


Fig. 5. Reference frames in KUKA KR600 simulator.

$$Rot(Z,\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

$$Rot(Y,\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad (2.2)$$

$$Rot(X,\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix} \quad (2.3)$$

Transformed accelerations then are filtered using 3rd order High-Pass Filter (3), where $\omega_{nHP/LP}$ is the natural frequency, $\zeta_{HP/LP}$ is the damping ratio, ω_b is break frequency and s is the Laplace operator. Optimal filter parameters are given in Table 1 considering recommendations from [1].

$$HP = \frac{s^2}{s^2+2\cdot\zeta_{HP}\cdot\omega_{nHP}\cdot s+\omega_{nHP}^2} \cdot \frac{s}{s+\omega_b} \quad (3)$$

TABLE I. WASHOUT FILTER PARAMETERS

Filter parameter	Value
ω_{nHP}	4
ω_{nLP}	8
ζ_{HP}	1
ζ_{LP}	1
ω_b	1

Filtered accelerations are integrated two times to get output translation vector $\underline{S}$ with x, y and z components (4).

$$\underline{S} = [x,y,z]^T \quad (4)$$

C. Tilt Coordination channel

Initially input accelerations are scaled. Then using (5) sustained accelerations are filtered in Low-Pass Filter block.

$$LP = \frac{\omega_{nLP}^2}{s^2+2\cdot\zeta_{LP}\cdot\omega_{nLP}\cdot s+\omega_{nLP}^2} \quad (5)$$

Then filtered accelerations are transformed into tilt angles $\underline{\beta_T}$ using (6).

$$\underline{\beta_T} = \begin{bmatrix} 0 & \frac{1}{g} & 0 \\ -\frac{1}{g} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \quad (6)$$

The driver must not sense this tilt as rotation. To keep tilt below human perception threshold, angles are limited to 3 *deg/s* in Rate Limiter block. Output of Rate Limiter block are Bryant angles $\underline{\beta}$ (7).

$$\underline{\beta} = [\varphi,\theta,\psi]^T \quad (7)$$

D. Software

Matlab Simulink software was used to create a model of the simplified Classic Washout Filter. Step size of Simulink model is set to *0.01 s*. Then output data are exported into a text file, for example *.csv* file, with 7 values in each line, separated by space symbol. First value is time, next is translation by x, y, z axis, then rotation angles φ, θ, ψ). Result is shown in Fig.6.

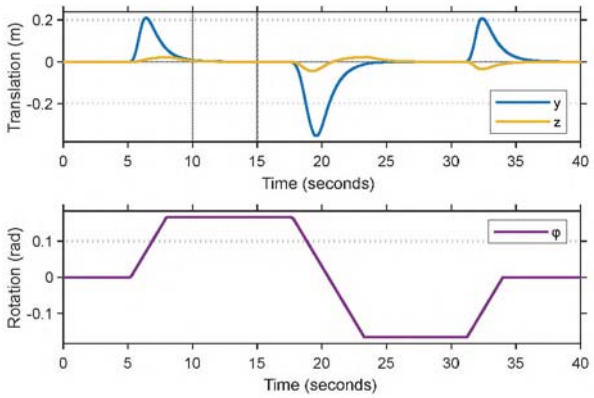


Fig. 6. CWF output translations y, z and rotation angle φ

IV. VIRTUAL TEST

Before applying received data to an actual robot, it is safer to test it in virtual environment. MOBILE is C++ object-oriented library for the modeling of mechatronic systems [5], [6]. In this case it is used to solve inverse kinematics for specific industrial robot manipulator model. Kinematics studies motion of rigid bodies without applying forces to them. Robot manipulator is open chain kinematic system. It means that all the joints have independent relative motions as in example Fig. 7.

Direct kinematics is the process of calculating the position on the robot's end-effector using values of joint parameters. But inverse kinematics calculates all the joint parameters from the given end-effector position. The problem is that one position of an end-effector can result in various possible and impossible robot configurations. There are multiple algorithms for solving inverse kinematics more effectively, avoiding singularity points.

MOBILE library with precise kinematic model of the KUKA KR600 robot makes it possible to solve inverse kinematics. Input for the simulation is the text file previously calculated in Simulink. Each line in the text file defines robot's end-effector positions and angles for each time step. MOBILE computes all joint parameters and visualizes robot motions in 3D environment as Fig. 8 shows. Then MOBILE checks for possible collisions based on robot's precise 3D model. If robot exceeds maximum accelerations, defined for passenger's safety, the problem can be noticed and solved timely.

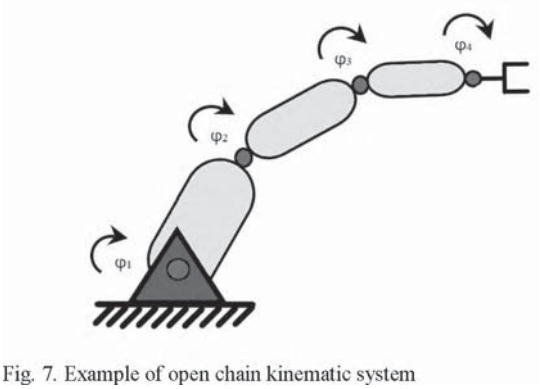
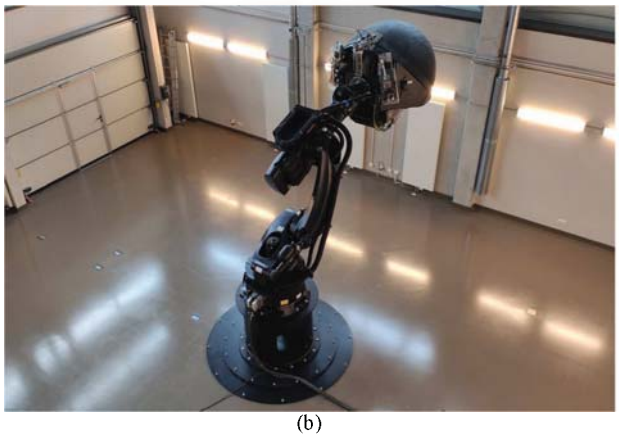


Fig. 7. Example of open chain kinematic system

(a)



(b)

Fig. 8. (a) Example of MOBILE virtual environment with robot 3D model, (b) KUKA KR600 robot

V. Real Robot

When motions cues were examined virtually, checked for collisions, and all safety requirements were fulfilled, final step is to test motion cues on a real robot. Students upload their generated *.csv* files to operator's computer, configure MOBILE to control the real robot and examine its motion. Control of KUKA KR600 robot manipulator using MOBILE is handled by KUKA Robot Sensor Interface (RSI) [7]. Robot has maximum acceleration limit, because of passenger safety precautions. If robot accidentally exceeds these limits, brakes are triggered.

Robot has 6 degrees of freedom controlled by 6 AC servomotors. Each drive has its own motion range and maximum rotation speed defined in manual. If these parameters are exceeded during operation, brakes are triggered. The robot also has maximum payload of *600 kg*, so the weight of the passenger does not affect robot's operation.

VI. Future Tasks and Challenges

A. Washout Filter improvements

For now, this project is based on simplified motion trajectory without acceleration and braking. Both turns of S-curve are identical to each other and angular velocity values are intentionally equated to zero. In situation close to reality it would be impossible to get such accurate trajectory by manually steering a vehicle. Consequently, one of the future tasks would be to consider acceleration and braking along the x axis, also as angular velocity changes. This way simulated motion cues will have more relevance to reality.

Classic Washout filter algorithm is computationally simple and has relatively transparent design [1], [8], so it is good for introducing students to the topic. But it is not the only one. There are other algorithms with improved performance [9], [10], [11]. The adaptive motion cueing algorithm can adjust filter parameters during operation. It generates more realistic motion cues compared to CWF [8], [12].

CWF algorithm works in Cartesian coordinate system. It is suited more for Stewart platform simulators because of the platform's design. For industrial robot manipulator it is more effective to use cylindrical coordinate system. This way it can represent lateral accelerations with circular motion of frame K_S around Z_0 axis. Hence, robot's motion limits are extended [2], [13].

B. Real-time simulation

One of the future tasks would be to transform corresponding real motions into simulator movements in real-time, without beforehand prepared data. It can be done using Unity game engine to receive data from a control device (joystick, keyboard, steering wheel etc.). Then using built-in physics to simulate car movements and calculate its accelerations. After that, send accelerations via UDP to MATLAB with installed Simulink Desktop Real-Time Toolbox. There accelerations are transformed into simulator motion cues in real-time. Then MATLAB sends calculated data to robot controller via UDP.

VII. Conclusion

The teaching methodology presented in the paper is well suited for online education. Using virtual environment to test the robot motion is essential, because it prevents any hardware damage due to student mistakes. If all the virtual tests were verified to be safe, there will be no threat for damaging the laboratory stand. The project can be integrated as practical work in a variety of courses, because it covers many topics – electromechatronics, robotics, motion cueing and washout filters.

References

[1] M. A. Nahon and L. D. Reid, "Simulator motion-drive algorithms - A designer's perspective," Journal of Guidance, Control, and Dynamics, vol. 13, no. 2, pp. 356–362, Mar. 1990, doi: 10.2514/3.20557.

[2] D. A. Pham, "A study on state-of-the-art motion cueing algorithms applied to planar motion with pure lateral acceleration — comparison , auto-tuning and subjective evaluation on a KUKA Robocoaster Serial Ride simulator," 2017.

[3] N. Murgovski, "Vehicle modelling and washout filter tuning for the Chalmers vehicle simulator," 2007.

[4] K. Stahl, K. D. Leimbach, A. Meroth, and R. Zollner, "A washout and a tilt coordination algorithm for a hexapod platform," in IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, 2015, vol. 2015-Octob, pp. 1196–1201, doi: 10.1109/ITSC.2015.197.

[5] A. Kecskemethy, MOBILE Version 1.3. User's guide. 1999.

[6] E. Auer, A. Kecskeméthy, M. Tändl, and H. Traczinski, "Interval algorithms in modeling of multibody systems," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 2991, pp. 132–159, 2004, doi: 10.1007/978-3-540-24738-8_8.

[7] A. Senfelds, "Analysis of motion modelling approaches for industrial robot applications," 2020, pp. 1–4, doi: 10.1109/aieee48629.2019.8977112.

[8] G. S. Natal et al., "Implementation analysis of a washout filter on a robotic flight simulator - a case study," Journal of Aerospace Technology and Management, vol. 11, Jan. 2019, doi: 10.5028/jatm.v11.978.

[9] Y.-H. Chang, C.-S. Liao, and W.-H. Chieng, "Optimal motion cueing for 5-DOF motion simulations via a 3-DOF motion simulator," 2008, doi: 10.1016/j.conengprac.2008.05.016.

[10] Z. Fang and A. Kemeny, "Explicit MPC motion cueing algorithm for real-time driving simulator," in Conference Proceedings - 2012 IEEE

7th International Power Electronics and Motion Control Conference - ECCE Asia, IPEMC 2012, 2012, vol. 2, pp. 874-878, doi: 10.1109/IPEMC.2012.6258965.

[11] A. Mohammadi, H. Asadi, S. Mohamed, K. Nelson, and S. Nahavandi, "Future reference prediction in model predictive control based driving simulators," 2016.

[12] H. Asadi, A. Mohammadi, S. Mohamed, and S. Nahavandi, "Adaptive translational cueing motion algorithm using fuzzy based tilt coordination," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8836, Springer Verlag, 2014, pp. 474-482.

[13] P. Robuffo Giordano, C. Masone, J. Tesch, M. Breidt, L. Pollini, and H. H. Bülthoff, "A novel framework for closed-loop robotic motion simulation - Part II: Motion cueing design and experimental validation," in Proceedings - IEEE International Conference on Robotics and Automation, 2010, pp. 3896-3903, doi: 10.1109/ROBOT.2010.5509945.

This is a post-print of a paper published in Proceedings of the 2021 IEEE 19th International Power Electronics and Motion Control Conference (PEMC) and is subject to IEEE copyright.
https://doi.org/10.1109/PEMC48073.2021.9432500
Electronic ISBN: 978-1-7281-5660-6.
USB ISBN: 978-1-7281-5659-0.
Print on Demand (PoD) ISBN: 978-1-7281-5661-3.