# Autonomous Robotic Arm for Object Sorting and Motion Compensation using Kalman Filter

Melanie Acelor Ndambani, Tianyang Fang and Jafar Saniie
*Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu/)*
*Department of Electrical and Computer Engineering*
*Illinois Institute of Technology, Chicago, Illinois, U.S.A.*

***Abstract* — Robotic arm has played a key role in the industry for more than a half-century. Although robotic arms have superior accuracy and efficiency, they are not always cost-effective, especially for small-scale applications. This paper proposes an autonomous robotic arm system that can identify, sort, and interact with different objects, using IoT devices. For precision robotic arm movement, we have developed a Kalman filter-based motion compensation system. The embedded computing system is used for real-time object recognition and motion compensation. Experimental results show that the proposed high precision autonomous robotic arm system can recognize objects with 95% accuracy, and locate and sort the moving objects with a mean error of 1.1mm.**

***Keywords— Robotic arm, Computer vision, Kalman filter, Electromagnets***

## I. INTRODUCTION

In modern days, robotic arms have seen increasing use in the industrial process for assembly, disassembly, and material handling. They replace workers for repetitive, dangerous, or exhaustive tasks, and are more precise and efficient than human workers in certain environments. Automated robotic arms only require minimum (if at all) control from humans while carrying out the designated tasks. On a production line, robotic arms are expected to interact with moving objects fast and precisely. To achieve this objective, we propose a robotic arm system that can sort objects and has motion compensation capability.

In the next section, we will overview the proposed robotic arm system and its mechanical details. Then in Section III, we will analyze the integrated control system and the algorithms used for the object recognition and motion compensation. Section IV evaluates the proposed design in success rate for object detection and trajectory prediction accuracy. Finally, Section V concludes the paper.

## II. ROBOTIC ARM SYSTEM DESIGN

The autonomous object recognition and sorting system, as shown in Fig. *1*, consists of a robotic arm, sorting buckets, and a conveyor. The workflow of this system is shown in Fig. *2*. An RGB camera is used as the input for the computer-vision-based object sorting and localization (see Fig. *2*). A Raspberry Pi hosts all the high-level functions, including the object sorting, localization, and the Kalman filter motion compensation. The object coordinates prediction is then passed down to the Arduino, where the object coordinates are used to determine the robotic arm motion sequence and the electromagnets switching sequence for picking the object. This action is achieved by controlling each motor on the robotic arm and the on-and-off timings of the electromagnets so that the robotic arm can reach the object accurately and finish picking up the object and dropping it into the matched sorting bucket properly.

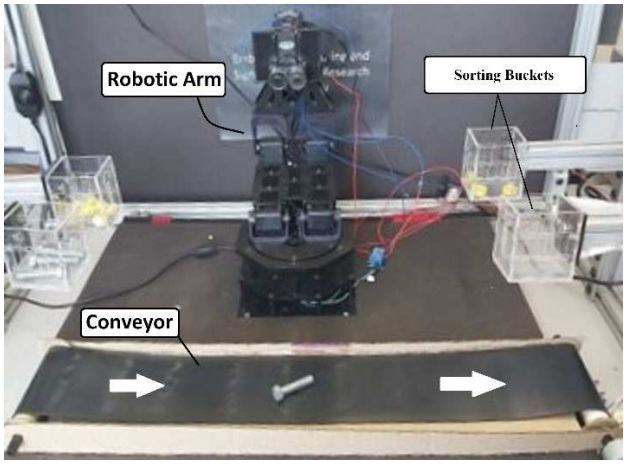


Fig. 1. The Robotic Arm System layout.

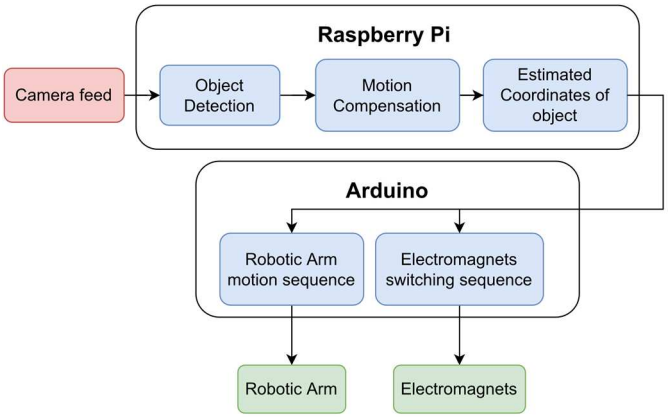


Fig. 2. Workflow of the Robotic Arm System

The robotic arm used in this research is a PhantomX Reactor Robotic Arm from Trossen Robotics. It has 8 motors controlled by pulse width modulation (PWM) with the robot controller ArbotiX-M microcontroller, programmable by the Arduino IDE software. Fig. *3* shows motor distribution on the robotic arm: Motor 1 controls the base rotation, which will move from 0° to

180°; Motors 2 and 3 control the shoulder; Motors 4 and 5 control the elbow; Motors 6 and 7 control the wrist vertically and horizontally; and Motor 8 controls the grip. Using the PWM, the position of each motor is defined by a number between 0 and 1024 which corresponds to the motor's degrees range. Regarding the pair of motors 2, 3, and 4, 5, one of the motors turns in the opposite direction of the other motor at the same speed to avoid breakdown. In this project, we use a function specifically created for the robotic arm, which optimally calculates the inverse kinematics of the arm and converts the object position into PWM values for all motors.


Fig. 3. Motor distribution on the robotic arm

This research used two electromagnets powered by 10V each to enable an efficient and quick grip and better precision. These electromagnets are controlled by the Arduino. These electromagnets can lift a maximun of 4 kg. We created and 3D printed a new fixture that was 6 cm wide and 5 cm long for mounting the electromagnets to the robotic arm. The two electromagnets are next to each other which creates a magnetic zone of 4 cm wide and 1 cm long. Fig. 4 shows a picture of the fixture for magnets. To move the objects in one direction at a certain speed, we prototyped a conveyor drive using two stepper motors powered by 5V each. The Arduino controls the speed of these motors.

## III. Object Sorting and Motion Compensation

The control system software, which hosts the object sorting and motion compensation algorithms, runs on a Raspberry Pi.

### *A. Object sorting algorithm*

First, we enter each point defining the border of a nut or a bolt. We also developed software for calibrating the camera which captures images of the objects. The captured images are processed using the following steps:

1) To extract the object shapes, the original image is filtered to a monotone image.
2) Each shape is then compared to predefined shapes.
3) Shapes that don't match are discarded.
4) Matched objects are further classified by RGB information.

Fig. 4. 3D-Printed Magnet Support with two electromagnets mounted.

In addition to the shape and color of the objects, the coordinates and orientation are also detected using computer vision. Fig. 5 shows the classes and coordinates of the detected objects. The Raspberry Pi then sends this information to the Arduino. The Arduino breaks down the target coordinates into a sequence of motions of the robotic arm and sends this sequence to the motors of the robotic arm, and controls the position of the electromagnets for picking up the object. With the object attached to the electromagnets, the Arduino uses the object class information to determine the sorting bucket for the object. After sending another sequence of motions that moves the object above the sorting bucket, the electromagnets are turned off to drop the object into the sorting bucket. The object identifications in terms of type, size, shape, and color can be expanded using classical image processing algorithms.


Fig. 5. Live camera feed with object detection. The object classes and coordinates are marked aside for respective objects.

### *B. Motion compensation using Kalman Filter*

The Kalman filter is named after Rudolf E. Kalman (May 19, 1930 – July 2, 2016) and was created in 1960. It is one of the most important and common estimation algorithms. The Kalman Filter produces estimates of variables based on uncertain measurements. As well, the Kalman Filter provides a

prediction of the future system state, based on the past measurements. Kalman filters combine the measurement and the prediction to find the optimal estimate of the object's position in the presence of process and measurement noise [4-10].

According to the Kalman model, we can estimate a state by taking into account both the current measurement and the previously estimated state [5]. To configure a Kalman Filter we need the following parameters:

- Motion Model
- Initial Location
- Motion Noise
- Measurement Noise

Since the objects move in 2 dimensions, we have Newton's motion equations here:

$$x = x_0 + \dot{x}_0 \Delta t + \frac{1}{2}\ddot{x}\Delta t^2 \quad (1)$$

$$y = y_0 + \dot{y}_0 \Delta t + \frac{1}{2}\ddot{y}\Delta t^2 \quad (2)$$

Where $x$ is the target's position along the x-axis; $y$ is the target's position along the y axis; $x_0$ is the target's initial position along the x-axis; $y_0$ is the target's initial position along the y axis; $\dot{x}_0$ is the target's initial velocity along the x-axis; $\dot{y}_0$ is the target's initial velocity along the y axis; $\ddot{x}$ is the target's acceleration along the x-axis; $\ddot{y}$ is the target's acceleration along the y axis; $\Delta t$ is the time between 2 position updates. Since we consider constant velocity and motion on x, we can simplify the model using:

$$x_{n+1} = x_n + \dot{x}_n \Delta t \quad (3)$$

$$\dot{x}_{n+1} = \dot{x}_n \quad (4)$$

then we have the prediction equations:

$$\hat{x}_{n+1} = \hat{x}_n + \hat{\dot{x}}_n \Delta t \quad (5)$$

$$\hat{\dot{x}}_{n+1} = \hat{\dot{x}}_n \quad (6)$$

With the motion model established, the algorithm can be summarized as the following three steps:

**Prediction:** First, estimate the initial position $x_0$, the speed and $\Delta t$ and we have:

$$\hat{x}_1 = \hat{x}_0 + \hat{\dot{x}}_0 \Delta t \quad (7)$$

$$\hat{\dot{x}}_1 = \hat{\dot{x}}_0 \quad (8)$$

**Measurement:** Next, measure the object's real position $z_n$. This is done with the camera.

**Update:** In this step, the prediction model is updated with the updated state equations:

$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n,n-1} + \alpha\left(z_n - \hat{x}_{n,n-1}\right) \quad (10)$$

$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n,n-1} + \beta * \frac{z_n - \hat{x}_{n,n-1}}{\Delta t} \quad (11)$$

where α is the measurement precision and β indicates the precision level of the camera detection. In our situation, both alpha and beta are high (0.9) since the measurement done by the camera are precise. These three steps are repeated until the robotic arm reaches the target.

## IV. Experiment Results

For the experiment, two types of objects, bolts, and nuts, with two different colors, yellow and silver, are positioned on the conveyor. The conveyor is set to run in two levels of speed, 1.3cm/s, and 2.6cm/s. Fig. *6* and Fig. *7* show the results of the motion compensation, where if the robotic arm successfully grabs the target object, it is counted as a success, otherwise, a failure. In the end, we achieved a 100% accuracy for bolts and nuts at 1.3cm/s and a 90% accuracy at 2.6cm/s. The miss sorting was due to the object not being grabbed at an optimal angle. To better quantify how accurate the motion compensation is, the predicted and real trajectories are compared in Fig. *8*. As the figure shows, the predicted trajectory matches the true target trajectory accurately, with a mean error of 1.1 mm.

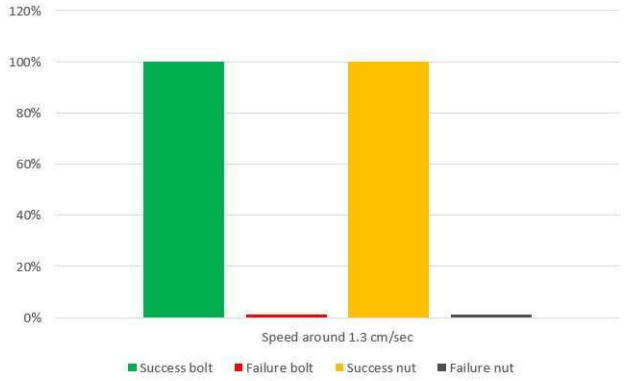


Fig. 6. The success rate of grabbing objects moving at 1.3cm/s

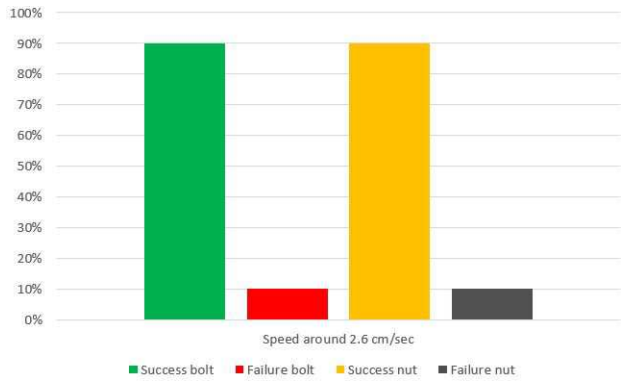


Fig. 7. The success rate of grabbing objects moving at 2.6 cm/s

## V. Conclusion

In this research, we designed an Automatic Robotic Arm system that uses computer vision to achieve object sorting capability and utilizes Kalman Filter to provide motion compensation for the robotic arm. In the experimental setup where objects are moving on a rotating conveyor, the designed system successfully grabbed and sorted 95% of objects, hence we consider the proposed design a success. For future works, we consider replacing the conventional computer vision algorithms with neural networks for better recognition accuracy.

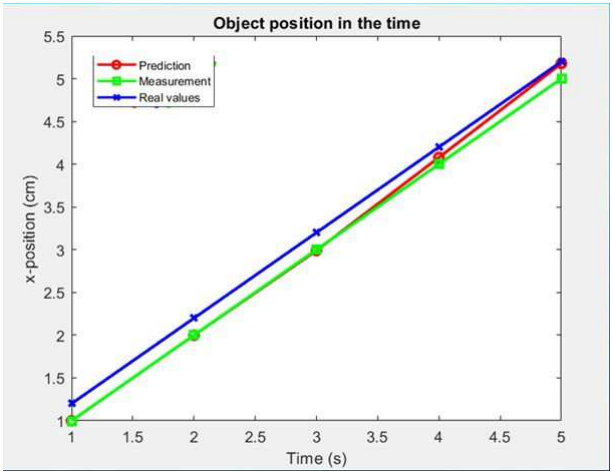


Fig. 8. Trajectory prediction using Kalman Filter. The prediction successfully converged to the true location of the object in five seconds.

## REFERENCES


[1] A. R. F. Quiros, A. C. Abad and E. P. Dadios, "Object locator and collector robotic arm using artificial neural networks," 2015 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2015, pp. 1-6, doi: 10.1109/HNICEM.2015.7393209.

[2] R. Szabó and A. Gontean, "Robotic arm control with stereo vision made in LabWindows/CVI," 2015 38th International Conference on Telecommunications and Signal Processing (TSP), 2015, pp. 1-5, doi: 10.1109/TSP.2015.7296382.

[3] Y. Jia, G. Yang and J. Saniie, "Real-time color-based sorting robotic arm system," 2017 IEEE International Conference on Electro Information Technology (EIT), 2017, pp. 354-358, doi: 10.1109/EIT.2017.8053385.

[4] R. Szabó and A. Gontean, "Robotic arm control with stereo vision made in LabWindows/CVI," 2015 38th International Conference on Telecommunications and Signal Processing (TSP), 2015, pp. 1-5, doi: 10.1109/TSP.2015.7296382.

[5] J. Maridor, M. Markovic and Y. Perriard, "Kalman filter to measure position and speed of a linear actuator," 2011 IEEE International Electric Machines & Drives Conference (IEMDC), 2011, pp. 330-335, doi: 10.1109/IEMDC.2011.5994869.

[6] A. E. Nordsjo, "A constrained extended Kalman filter for target tracking," Proceedings of the 2004 IEEE Radar Conference (IEEE Cat. No.04CH37509), 2004, pp. 123-127, doi: 10.1109/NRC.2004.1316407.

[7] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373), 2000, pp. 153-158, doi: 10.1109/ASSPCC.2000.882463.

[8] F. Gustafsson et al., "Particle filters for positioning, navigation, and tracking," in IEEE Transactions on Signal Processing, vol. 50, no. 2, pp. 425-437, Feb. 2002, doi: 10.1109/78.978396.

[9] S. J. Julier, J. K. Uhlmann and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," Proceedings of 1995 American Control Conference - ACC'95, 1995, pp. 1628-1632 vol.3, doi: 10.1109/ACC.1995.529783.

[10] P. J. Hargrave, "A tutorial introduction to Kalman filtering," IEE Colloquium on Kalman Filters: Introduction, Applications and Future Developments, 1989, pp. 1/1-1/6