Impact Factor (SJIF): 6.092 **E-ISSN: 2583-1615**

International Journal of Research in Academic World

Received: 02/August/2024 **IJRAW:** 2024; 3(9):25-31 **Accepted:** 07/September/2024

Real-Time Processing of Localization for Piping Inspection Robot

[*1]Hirofumi Maeda

[*1]Professor, Information Science and Technology Department, National Institute of Technology (KOSEN), Yuge College, Yuge Shimoyuge, Kamijima-cho Ochi-gun, Ehime, Japan.

Abstract

The sewerage pipes laid in Japan are extensive, spanning approximately 470,000 km, with most of them constructed during the high economic growth period from around 1955 to around 1973. According to indicators from the Ministry of Land, Infrastructure, Transport, and Tourism, the service life of sewerage pipes is estimated at 50 years. This suggests that many sewer pipes across Japan, installed over 50 years ago, are becoming obsolete. Consequently, piping inspections using robots have commenced in Japan. Currently, stand-alone types that can be inspected by a single robot are garnering attention. Meanwhile, we have been conducting research and development with the aim of implementing a small, easily portable, stand-alone piping inspection robot. Furthermore, numerous stand-alone types have been employed to prevent falls by adjusting the tire shape or the distance between the axles. However, this hardware approach does not completely prevent falls. Therefore, we have opted for a software approach to explore measures to prevent falls by controlling driving, aiming to achieve the advanced localization required for this purpose. Localization ultimately needs to be implemented on an embedded computer or microcomputer installed in the robot itself, and for this purpose, localization must be processed in real-time. However, the previous method did not meet this requirement. Therefore, this paper describes the method and usefulness of real-time processing for localization using embedded PCs and microcomputers.

Keywords: Localization, real-time, numerical analysis, inspection robot, water pipe

Introduction

Sewer pipes laid in Japan are vast, spanning approximately 470,000 km, with most constructed during the high economic growth period from around 1955 to 1973. Indicators from the Ministry of Land, Infrastructure, Transport, and Tourism suggest that the service life of sewer pipes is 50 years. This indicates that many sewer pipes across Japan, installed for over 50 years, are becoming obsolete. In fact, problems such as sewage pipes collapsing and rainwater gushing out due to frequent heavy rainfall. While replacing old pipes is desirable to address these challenges, it remains impractical due to cost and human resource constraints. Therefore, each local governments conduct detailed inspections to identify problem areas and conduct partial repairs on site. Nonetheless, these tasks are labor-intensive and time-consuming.

Given this background, piping inspections using robots have been introduced in Japan [1]. Currently, mainstream pipe inspection robots are remotely controlled from within an operation vehicle called a transport vehicle installed on the ground. This design allows for equipping the transport vehicle with essential functions such as power supply and a control unit, making the self-propelled robot lightweight and highly maneuverable. Furthermore, real-time control and monitoring are facilitated as the operator can remotely operate the system from the transport vehicle. However, the installation of a transport vehicle poses challenges such as space requirements and traffic regulation at the work site, leading to increased costs. Consequently, stand-alone types, capable of inspection by a single robot, are gaining attention. Nevertheless, the development of stand-alone piping inspection robots in Japan has been limited. Additionally, many overseas robots are designed for pipe diameters of 200 mm or more, rendering them unsuitable for Japan where numerous pipes have diameters of 150 mm [2]-[4]. In light of this, our research and development efforts focus on implementing a small, easily portable, stand-alone piping inspection robot [5]-[9].

Therefore, achieving a stand-alone type pipe inspection robot requires ensuring the robot body's stability within the pipe. To address this, many stand-alone types have attempted to prevent falls by adjusting tire shape or axle distances. However, this hardware-centric approach does not offer complete fall prevention. Thus, we have opted for a software approach to mitigate falls by controlling driving, aiming to achieve the advanced localization necessary for this purpose [10]-[11].

So far, we have demonstrated the usefulness of the localization method using special three-dimensional position measuring instruments in environments where tire shape does not affect [12]-[21]. Currently, we are trying to build a highly accurate localization that considers the influence of tire shape, apart from this, the processing time for it is an issue. Localization ultimately needs to be implemented on an

***Corresponding Author:** Hirofumi Maeda

embedded computer or microcomputer installed in the robot itself, and for this purpose, localization must be processed in real-time. However, the previous method did not meet this requirement. Therefore, this paper describes the method and usefulness of real-time processing for localization using embedded PCs and microcomputers.

Robot Localization Method

Here, we describe the localization method that is subjected to real-time processing. The localization method is performed under the following two conditions.

i). 3 or More Tires Touch the Ground

The localization method assumes a robot consisting of four tires as shown in Figure 1, and three tires touch the inside of the pipe when stationary. However, if the robot is oriented directly along the direction of travel, all four tires will make contact. Note that the robot may vibrate based on the two diagonal tires that are in contact with the pipe when a shock is applied to the robot, and in this case, the robot will be in contact with the ground at two points momentarily. However, this phenomenon is not considered because it is rare and the robot immediately tilts toward the center of gravity and the vibration subsides.

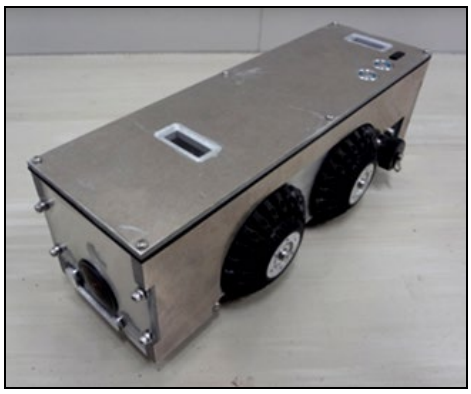

Fig 1: Target Piping Inspection Robot

ii). The Bottom of the Tire Touches the Inside of the Pipe

For all tires, the point where they touch the inside of the pipe is the area directly beneath them, perpendicular to the robot's top plate, and passing through the tire axle.

a) Acceleration of Gravity

The coordinate system is a right-handed orthogonal coordinate system. In addition, as shown in Figure 2, we establish a robot coordinate system with the center of the robot as the origin, the front of the robot as the positive direction of the x-axis, and the direction directly beneath the robot's top plate as the negative direction of the y-axis. Furthermore, the pipe is positioned along the x-axis so that its center intersects with the origin of the absolute coordinate system. The pipe coordinate system is then inclined at an angle of $^{\theta_s}$ [rad] with respect to the x-axis. Consequently, the piping coordinate system undergoes a rotation of $^{\theta_s}$ [rad] around the y-axis relative to the absolute coordinate system

Next, the acceleration sensor mounted on the robot is discussed. When the robot is stationary, the direction of gravity can be determined from each component of the acceleration sensor. Similarly, the roll and pitch, representing the robot's orientations in the absolute coordinate system, can be computed using each component of the sensor in equations (1) and (2). Note that each variable is defined as follows.

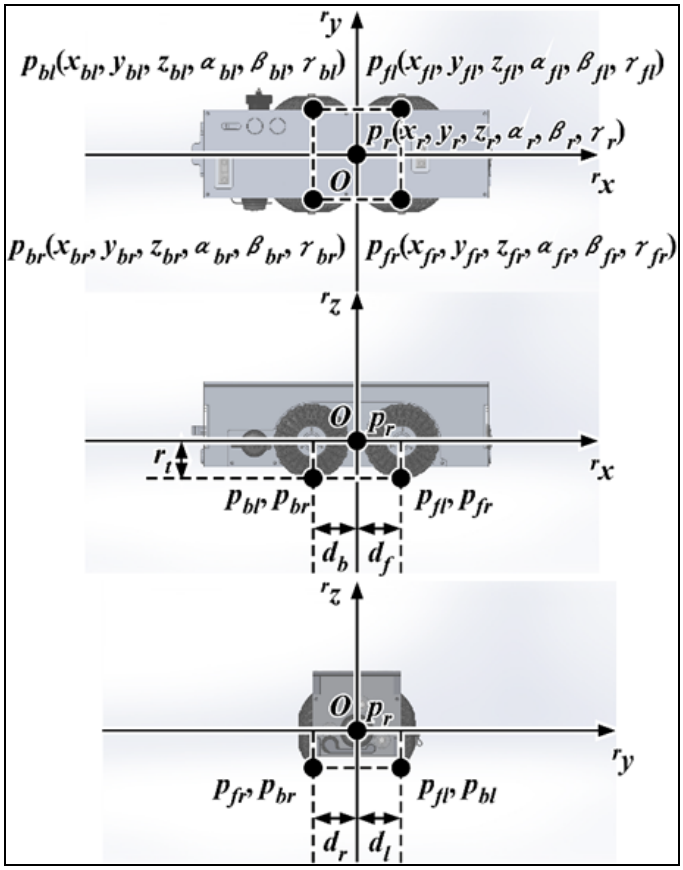

Fig 2: Robot Coordinate System and Variable Declaration

$^{o}\alpha_r$: Rotation angle around the *x*-axis of the robot (roll) (absolute coordinate system) (rad)

$^{o}\beta_r$: Rotation angle around the *y*-axis of the robot (pitch) (absolute coordinate system) (rad)

$^{r}a_x$: *x*-component of the accelerometer (robot coordinate system) (m/s^2)

$^{r}a_y$: *y*-component of the accelerometer (robot coordinate system) (m/s^2)

$^{r}a_z$: *z*-component of the accelerometer (robot coordinate system) (m/s^2)

$$^{o}\alpha_r = -\tan^{-1}\frac{^{r}a_y}{-^{r}a_z} \quad (1)$$

$$^{o}\beta_r = -\tan^{-1}\frac{-^{r}a_x}{-^{r}a_z} \quad (2)$$

b) Grounding of Tire and Pipe

Condition 1 indicates that the robot's tires and the interior of the pipe come into contact at three or more points. When viewing the pipe from the front, the circularity is maintained, leading to the derivation of equation (3). Also, $^{s}y_i$ and $^{s}z_i$ can be expressed by equation (4) from condition 2. The mounting position $[\,^{o}x_i \quad ^{o}y_i \quad ^{o}z_i]^T$ of each tire in equation (4) is derived by adding the displacement to the mounting position of each tire to the robot position $[\,^{o}x_r \quad ^{o}y_r \quad ^{o}z_r]^T$ in the absolute coordinate system. Note that the mounting position of each tire in the absolute coordinate system is determined by rotating the tire mounting position in the robot coordinate system by $^{o}\alpha_r$ [rad] around the *x*-axis, $^{o}\beta_r$ [rad] around the *y*-axis, and $^{o}\gamma_r$ [rad] around the *z*-axis. Furthermore, by substituting equation (4) into equation (3),

equations (5) and (6) are derived. The definitions of variables used in each equation are as follows.

$^{o}x_r$: x-componet of the robot position (absolute coordinate system) (m)

$^{o}y_r$: y-componet of the robot position (absolute coordinate system) (m)

$^{o}z_r$: z-componet of the robot position (absolute coordinate system) (m)

$^{o}\gamma_r$: Rotation angle around the z-axis of the robot (yaw) (absolute coordinate system) (rad)

i: Each wheel of the robot (fl, fr, br, bl)

fl: The robot left front wheel

fr: The robot right front wheel

bl: The robot left rear wheel

br: The robot right rear wheel

$^{o}x_i$: The robot tire mounting position (x-componet) (absolute coordinate system) (m)

$^{o}y_i$: The robot tire mounting position (y-componet) (absolute coordinate system) (m)

$^{o}z_i$: The robot tire mounting position (z-componet) (absolute coordinate system) (m)

$^{s}y_i$: The robot tire mounting position (y-componet) (pipe coordinate system) (m)

$^{s}z_i$: The robot tire mounting position (z-componet) (pipe coordinate system) (m)

$^{r}x_i$: The robot tire mounting position (x-componet) (robot coordinate system) (m)

$^{r}y_i$: The robot tire mounting position (y-componet) (robot coordinate system) (m)

$^{r}z_i$: The robot tire mounting position (z-componet) (robot coordinate system) (m)

$^{o}p_i$: Grounding position of the tires and the pipe (world coordinate system) (m)

r_i: The tires radius (m)

r_i: Inner radius of the pipe (m)

$$r_s^2 = {}^{s}y_i^2 + {}^{s}z_i^2 \tag{3}$$

$$
\begin{aligned}
{}^{o}p_i &= \begin{bmatrix} {}^{o}x_i \\ {}^{o}y_i \\ {}^{o}z_i \end{bmatrix} (i = fl, fr, bl, br) \\
&= \begin{bmatrix} \cos{}^{o}\gamma_r & -\sin{}^{o}\gamma_r & 0 \\ \sin{}^{o}\gamma_r & \cos{}^{o}\gamma_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&\quad \begin{bmatrix} \cos{}^{o}\beta_r & 0 & \sin{}^{o}\beta_r \\ 0 & 1 & 0 \\ -\sin{}^{o}\beta_r & 0 & \cos{}^{o}\beta_r \end{bmatrix} \\
&\quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos{}^{o}\alpha_r & -\sin{}^{o}\alpha_r \\ 0 & \sin{}^{o}\alpha_r & \cos{}^{o}\alpha_r \end{bmatrix} \begin{bmatrix} {}^{r}x_i \\ {}^{r}y_i \\ {}^{r}z_i \end{bmatrix} + \begin{bmatrix} {}^{o}x_r \\ {}^{o}y_r \\ {}^{o}z_r \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
&= \begin{bmatrix}
{}^{r}x_i \cos{}^{o}\beta_r \cos{}^{o}\gamma_r \\
+ {}^{r}y_i(\sin{}^{o}\alpha_r \sin{}^{o}\beta_r \cos{}^{o}\gamma_r \\
-\cos{}^{o}\alpha_r \sin{}^{o}\gamma_r) \\
+ {}^{r}z_i(\cos{}^{o}\alpha_r \sin{}^{o}\beta_r \cos{}^{o}\gamma_r \\
+\sin{}^{o}\alpha_r \sin{}^{o}\gamma_r) + {}^{o}x_r \\
\\
{}^{r}x_i \cos{}^{o}\beta_r \sin{}^{o}\gamma_r \\
+ {}^{r}y_i(\sin{}^{o}\alpha_r \sin{}^{o}\beta_r \sin{}^{o}\gamma_r \\
+\cos{}^{o}\alpha_r \cos{}^{o}\gamma_r) \\
+ {}^{r}z_i(\cos{}^{o}\alpha_r \sin{}^{o}\beta_r \sin{}^{o}\gamma_r \\
-\sin{}^{o}\alpha_r \cos{}^{o}\gamma_r) + {}^{o}y_r \\
\\
-{}^{r}x_i \sin{}^{o}\beta_r \\
+ {}^{r}y_i \sin{}^{o}\alpha_r \cos{}^{o}\beta_r \\
+ {}^{r}z_i \cos{}^{o}\alpha_r \cos{}^{o}\beta_r + {}^{o}z_r
\end{bmatrix}
\end{aligned} \tag{4}
$$

$$
\begin{aligned}
{}^{s}y_i = {} & ({}^{r}x_i \cos{}^{o}\beta_r + {}^{r}y_i \sin{}^{o}\alpha_r \sin{}^{o}\beta_r \\
& + {}^{r}z_i \cos{}^{o}\alpha_r \sin{}^{o}\beta_r) \sin{}^{o}\gamma_r \\
& + ({}^{r}y_i \cos{}^{o}\alpha_r - {}^{r}z_i \sin{}^{o}\alpha_r) \cos{}^{o}\gamma_r \\
& + {}^{o}y_r
\end{aligned} \tag{5}
$$

$$
\begin{aligned}
{}^{s}z_i = {} & ({}^{r}y_i \cos{}^{o}\alpha_r \sin\theta_s \\
& - {}^{r}z_i \sin{}^{o}\alpha_r \sin\theta_s) \sin{}^{o}\gamma_r \\
& + (-{}^{r}x_i \cos{}^{o}\beta_r \sin\theta_s \\
& - {}^{r}y_i \sin{}^{o}\alpha_r \sin{}^{o}\beta_r \sin\theta_s \\
& - {}^{r}z_i \cos{}^{o}\alpha_r \sin{}^{o}\beta_r \sin\theta_s) \cos{}^{o}\gamma_r \\
& + ({}^{r}x_i \sin{}^{o}\beta_r \cos\theta_s \\
& + {}^{r}y_i \sin{}^{o}\alpha_r \cos{}^{o}\beta_r \cos\theta_s \\
& + {}^{r}z_i \cos{}^{o}\alpha_r \cos{}^{o}\beta_r \cos\theta_s) \\
& + {}^{o}z_r \cos\theta_s
\end{aligned} \tag{6}
$$

c) Removal of x Component

$^{o}x_r$ in equation (4) corresponds to the distance the robot has traveled within the pipe, and has no direct relationship to preventing the robot from falling over. In addition, $^{o}x_r$ itself can be directly measured using odometry. Therefore, it is excluded by setting $^{o}x_r = 0$. As a result, the unknown variables in the localization are five components: $^{o}y_r$, $^{o}z_r$, $^{o}\alpha_r$, $^{o}\beta_r$, and $^{o}\gamma_r$. It is assumed that r_s and θ_s are known in advance.

d) Derivation of Unknown Variables

T As mentioned in III.C, the unknown variables found in the localization are the five components: $^{o}y_r$, $^{o}z_r$, $^{o}\alpha_r$, $^{o}\beta_r$, and $^{o}\gamma_r$. Among these, $^{o}\alpha_r$ and $^{o}\beta_r$ can be calculated from equations (1) and (2) using sensor values obtained from the acceleration sensor mounted on the robot. Subsequently, three or more equations are obtained by substituting the obtained $^{o}\alpha_r$ and $^{o}\beta_r$ into equation (3) based on condition 1. Furthermore, an equation consisting only of $^{o}\gamma_r$ is obtained using three equations and eliminating $^{o}y_r$ and $^{o}z_r$. However, in this formula, $^{o}\gamma_r$ is expressed by a trigonometric function. Therefore, the solution for $^{o}\gamma_r$ is solved by performing numerical analysis on $^{o}\gamma_r$ in the range of ±30 degrees. Finally, $^{o}y_r$ and $^{o}z_r$ are calculated using the value of $^{o}\gamma_r$.

Speeding Up Localization

To process localization in real-time, it is necessary to incorporate a new method for high-speed processing in addition to the conventional method. Therefore, we propose ways to speed up processing by revising processing procedures for numerical analysis and processing for problems specific to programming languages for implementation in embedded computers. We also propose array processing for implementation on microcomputers.

a) Handling Programming Language

Machine learning is essential for localization based on the influence of tire shape, which is currently being researched. Therefore, we selected Python widely used for machine learning, as the programming language used for localization.

Python is suitable for this research because it has a wealth of machine learning libraries used for AI and deep learning. On the other hand, since Python is an interpreted high-level programming language, it has the problem of slow processing speed. In addition, since there are many libraries available, there is a risk that the processing speed will slow down significantly if the wrong library is selected. The processing speed of Python can be solved by increasing the performance of embedded computers in recent years and by using CPython, which is written in the C language. Furthermore, since plans are underway to speed up Python, the processing speed is improving as the years roll on. Therefore, the important issue here is the selection of the library. The localization method uses many trigonometric functions, and the selection of the library used for the processing is important. Famous libraries with trigonometric functions in Python are math and SymPy. However, geometric calculations using SymPy require a significant amount of processing time. Therefore, it is necessary to select trigonometric functions of math. Other libraries do not affect the localization process and do not require special attention.

b) Improving Efficiency of Numerical Analysis Processing

In localization, the most time deriving unknown variables is the numerical analysis of ${}^{o}\gamma_r$. Figure 3 shows an example of the results of numerical analysis of ${}^{o}\gamma_r$ within a range of ± 30 degrees at 0.1 degrees intervals. When numerical analysis is performed, three convergence points occur, as seen in the figure. The first point is the convergence point to infinity that occurs when the denominator approaches 0 infinitely, which can be ignored. The remaining two points represent the actual state (real image) where the robot is installed at the bottom of the pipe and the impossible state (virtual image) where the robot is installed at the ceiling of the pipe, as shown in Figure 4. The real image and the virtual image can be discriminated by the estimated value of ${}^{o}z_r$. In the case of the virtual image, the robot exceeds a certain reference value and floats in the air.

Until now, in the numerical analysis of ${}^{o}\gamma_r$, we have performed all calculations within a range of ±30 degrees at 0.1 degrees intervals, and then searched and selected an appropriate solution. Therefore, it was necessary to perform 301 calculations for each localization. However, ${}^{o}\gamma_r$ often takes a value close to 0 degrees. Furthermore, when straight-ahead control is implemented, the value converges to 0 degrees so that the value will remain near 0 degrees. In other words, almost all the numerical analysis has been a waste of calculations. Previously, the numerical analysis was performed from -30 to 30 degrees in 0.1-degrees steps as shown in Figure 5. Change this so that numerical calculations are performed from 0 degrees to a singular point that becomes a real or virtual image in the negative and positive directions. This can significantly reduce the amount of calculation. As a result, localization becomes faster and more efficient overall.

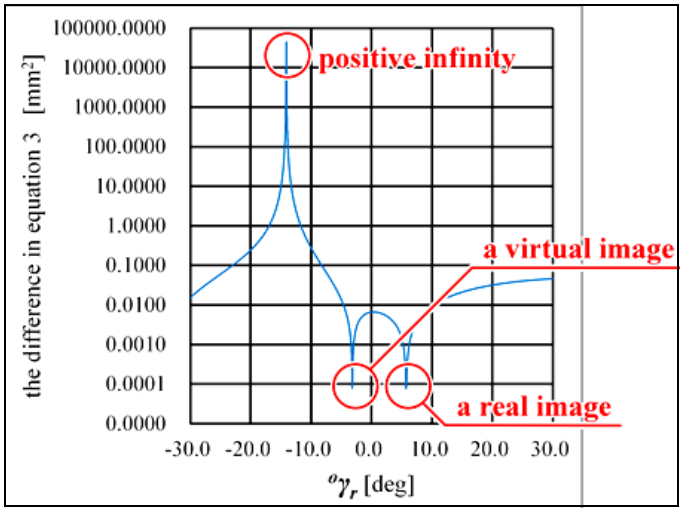

Fig 3: Three convergence points

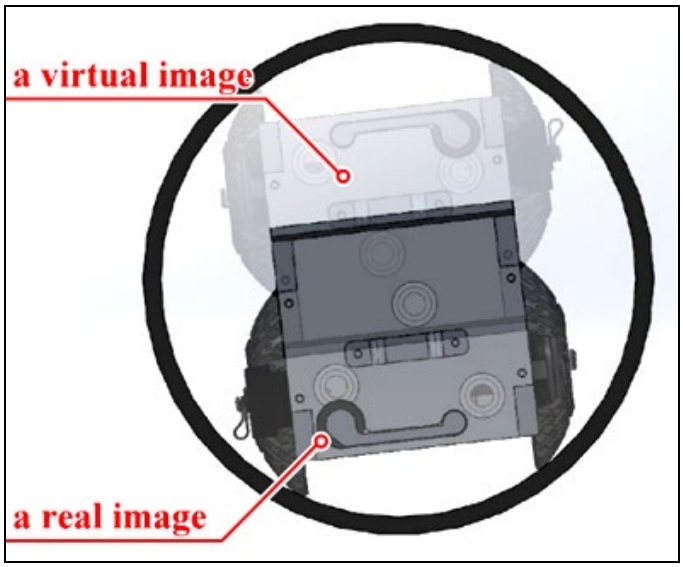

Fig 4: Real image and virtual image

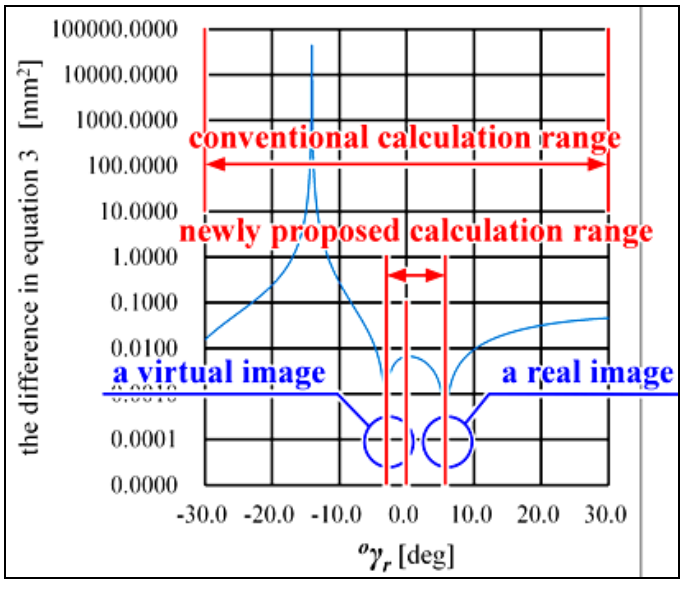

Fig 5: Calculation range

c) Omit Estimation by Array Processing

Although the amount of calculation is significantly reduced by increasing the efficiency of the numerical analysis process shown in b), the load may be too large to implement on a microcomputer. Also, since the amount of calculation varies depending on the estimated value of ${}^{o}\gamma_T$, this may be undesirable if the processing speed and feedback cycle should be kept constant.

Therefore, we pay attention that ${}^{o}\gamma_T$ is uniquely determined by two variables, ${}^{o}\alpha_T$ and ${}^{o}\beta_T$. In a program, two inputs and one output can be expressed using an array. Therefore, calculate ${}^{o}\gamma_T$ by combining ${}^{o}\alpha_T$ and ${}^{o}\beta_T$ in advance, and store these in a two-dimensional array. This makes it possible to instantly extract estimation results without performing localization. However, this method has two problems. First, since a two-dimensional array is used, ${}^{o}\alpha_T$, ${}^{o}\beta_T$, and ${}^{o}\gamma_T$ take discrete values, which causes errors in the estimation results. This can be solved by interpolating the values between each variable. Another problem is that the two-dimensional array data must be stored in the microcomputer in advance, and depending on the machine's specifications, it may run out of memory. This can be solved by adding external memory or reading data from an external file.

Verification of Speeding Up

A general desktop computer was used for operational verification to confirm that the proposed method is useful. Table 1 shows the PC specifications of the personal computer used this time and the embedded computer Raspberry Pi 5/8GB.

Next, Table 2 shows the estimation results obtained through high-speed processing. This estimation was performed by installing the robot inside a pipe with a nominal diameter of 200 mm. ${}^{o}\gamma_T$ was varied in five steps from negative to positive, and two types of tires were used, corresponding to pipe diameters of 150 mm and 200 mm. Note that the processing time t is the average value of the 10 measured processing times. The sampling period of the inertial measurement unit used for localization is 20 msec. Therefore, the calculation speed is less than 0.5 msec according to the estimation results in Table 2, so the requirements are fully met and the speeding up has been successful. Furthermore, when implemented on an embedded computer, the comparison of specifications in Table 1 shows that calculations can be performed on the order of milliseconds. Therefore, this method is also useful for embedded computers. Next, Figures 6 and 7 show graphs of the data in Table 2. From the graph, as the absolute value of ${}^{o}\gamma_T$ increases from 0 degrees as the origin, the processing time also increases and varies. This is because the number of numerical calculations increases as the absolute value of ${}^{o}\gamma_T$ increases.

Table 3 shows the results estimated by array processing using the same data. As before, the processing time t is the average value of 10 measurements. Comparing Table 2 and Table 3, the processing time using array processing is more than 16 times faster than high-speed processing for efficiency. Also, the variation in processing time has been eliminated. However, as shown in Table 4, an error of up to 0.3 degrees occurs for ${}^{o}\gamma_T$. This can be handled by interpolating the values as mentioned earlier. From the above, localization using array processing is also sufficiently useful.

Table 1: PC Specifications

Model Name	LEVEL-C066-LC127-SAX	Raspberry Pi 5/8GB
Product Model Number	ILeDEs-C066-L127-SASXB	SC1112
CPU	Intel Core i7-12700 Processor (4.90GHz/12 core)	2.4GHz quad-core Cortex-A76 (ARMv8, 64bit, L2: 512KB, L3: 2MB)
GPU	GeForce RTX 3060 Ti 8GB GDDR6	VideoCore VII
Memory	32GB (16GB×2) DDR4-3200	8GB LPDDR4X-4267 SDRAM, 2133MHz
OS	Windows 11 Home (DSP)	Raspberry Pi OS

Table 2: Estimated Data in High-speed Processing

Type of Tire	Estimated Data in High-speed Processing					
	y [mm]	z [mm]	α [degree]	β [degree]	γ [degree]	t [msec]
150	-2.443	-38.934	-19.127	4.016	-17.100	0.422
150	1.376	-40.950	-7.739	3.714	-9.200	0.343
150	-7.505	-41.424	-16.072	1.196	-5.700	0.188
150	6.965	-42.114	14.679	1.191	5.200	0.180
150	5.035	-40.143	18.417	2.490	11.700	0.322
200	-2.417	-34.886	-16.380	2.695	-11.000	0.306
200	-0.316	-35.586	-9.595	2.624	-7.500	0.259
200	-5.238	-36.479	-14.817	1.307	-5.700	0.202
200	1.150	-35.275	12.260	2.712	8.800	0.284
200	1.006	-34.571	15.478	3.349	12.300	0.352

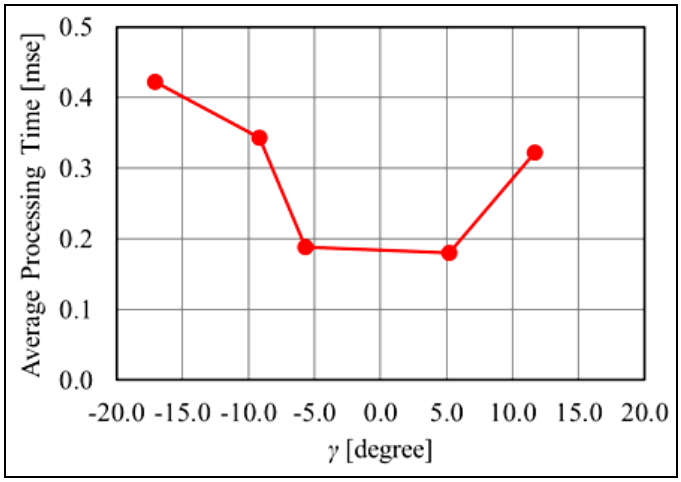

Fig 6: Processing Time with Tires for Pipe Diameter 150 mm

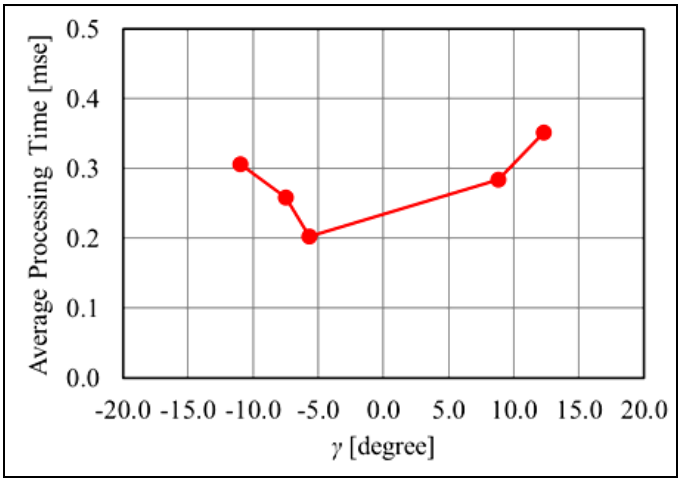

Fig 7: Processing Time with Tires for Pipe Diameter 200 mm

Table 3: Estimated Data in Array Processing

Type of tire	Estimated Data in Array Processing					
	y [mm]	*z* [mm]	*α* [degree]	*β* [degree]	*γ* [degree]	*t* [msec]
150	-2.275	-38.844	-19.127	4.016	-17.200	0.011
150	1.575	-40.738	-7.739	3.714	-9.300	0.011
150	-7.507	-41.910	-16.072	1.196	-5.700	0.011
150	6.878	-41.711	14.679	1.191	5.200	0.011
150	5.036	-40.128	18.417	2.490	11.800	0.011
200	-2.363	-34.907	-16.380	2.695	-10.900	0.011
200	-0.099	-35.454	-9.595	2.624	-7.600	0.012
200	-4.817	-35.695	-14.817	1.307	-5.900	0.012
200	1.049	-35.594	12.260	2.712	9.100	0.011
200	0.890	-34.277	15.478	3.349	12.400	0.011

Table 4: Estimated Data Error

Type of Tire	Estimated Data Error				
	y [mm]	*z* [mm]	*α* [degree]	*β* [degree]	*γ* [degree]
150	-0.168	-0.090	0.000	0.000	0.100
150	-0.198	-0.212	0.000	0.000	0.100
150	0.001	0.486	0.000	0.000	0.000
150	0.087	-0.403	0.000	0.000	0.000
150	-0.002	-0.015	0.000	0.000	-0.100
200	-0.054	0.022	0.000	0.000	-0.100
200	-0.217	-0.132	0.000	0.000	0.100
200	-0.421	-0.784	0.000	0.000	0.200
200	0.100	0.320	0.000	0.000	-0.300
200	0.116	-0.294	0.000	0.000	-0.100

Conclusion

In this paper, we proposed a method to speed up the real-time localization processing for embedded computers and microcomputers. How to handle programming languages, improve the efficiency of numerical analysis processing, and omit estimation through array processing were described. Furthermore, verification using an actual machine showed its usefulness. In the future, we would like to build a highly accurate localization method with the influence of tire shape and implement it with a robot. In addition, two issues that must be solved in the process are presented.

i). **Development of Measuring Instrument for Machine Learning:** Machine learning is essential in building a localization method considering the influence of tire shape. In addition, it is necessary to perform machine learning to install it on a robot, and a large amount of measurement data is required for this learning. Although the currently used measuring instrument can accurately measure the robot's position and orientation, it takes time to make one measurement. Therefore, it is necessary to develop measuring instruments that can acquire large amounts of measurement data for machine learning in a short time.

ii). **Measuring Instrument Calibration:** The accuracy of the measurement data used for machine learning greatly affects the learning results. Therefore, it is necessary to calibrate the measuring instrument to improve its accuracy. In addition, installation errors between the

measuring instrument and a test field can also cause errors in the measured data, so it is necessary to calibrate it.

Acknowledgement
This work was supported by Grant-in-Aid for Scientific Research(C) Grant Number 23K04316.

References

1. Japan institute of wastewater engineering technology, Development foundation survey of sewerage facilities management robot, Sewer new technology Annual report of the Institute, 1992, 43-52.
2. Rome E, Hertzberg J, Kirchner F, Licht U and Christaller T. towards Autonomous Sewer Robots: the MAKRO Project, Urban Water. 1999; 1:57-70.
3. Streich H and Adria O. Software approach for the autonomous inspection robot MAKRO, in Proceedings of the 2004 IEEE International Conference Robotics and Automation, 2004, 3411-3416.
4. Birkenhofer C, Regenstein K, Zöllner JM and Dillmann R. Architecture of multi-segmented inspection Robot KAIRO-II, DOI: 10.1007/978-1-84628-974-3_35, In book: Robot Motion and Control, 2007, 381-389.
5. Ayaka N, Kazutomo F, Toshikazu S, Mikio G and Hirofumi M. Prototype design for a piping inspection robot, 43rd Graduation Research Presentation Lecture of Student Members of the JSME, 2013, 716.
6. Kazutomo F, Yoshiki I and Hirofumi M. Modularization for a piping inspection robot, 2013 Symposium on System Integration, 2013, 1297-1300.
7. Kazutomo F, Toshikazu S, Mikio G, Yoshiki I and Hirofumi M. Miniaturization of the piping inspection robot by modularization, 44th Graduation Research Presentation Lecture of Student Members of the JSME, 2014, 613.
8. Hirofumi M, Takuya K, Kazutomo F, Yoshiki I, Toshikazu S and Mikio G. Research and development about a piping inspection robot-Report1: Prototype design for a miniaturization-, Bulletin of National Institute of Technology, Yuge College. 2014; 36:79-82.
9. Hirofumi M, Yoshiki I. Toshikazu, S. and Mikio, G., Research and development about a piping inspection robot-Report2: Prototype design for maintenance improvement-, Bulletin of National Institute of Technology, Yuge College. 2015; 37:75-79.
10. Hirofumi M, Ryota K. Development of a small autonomous pipe inspection robot (Modularization of hardware using the technique of wooden mosaic work), Transactions of the Japan Society of Mechanical Engineers. 2016; 82(839):1-16.
11. Hirofumi M. Automatic Compensation of the Positional Error Utilizing Localization Method in Pipe, *International Journal of Recent Technology and Engineering (IJRTE)*. 2021; 9(6):151-157.
12. Yuki Y, Yoshiki I, Hirofumi M. Self-localization Measurement of the Piping Inspection Robot by an ARtoolkit, Transactions of the Japan Society of Mechanical Engineers. 2016; 165(1):502.
13. Ayano, T., Hirofumi, M., Accuracy Improvement of the Measuring Instrument for the Piping Inspection Robot, The Japan Society of Mechanical Engineers Chugoku-Shikoku Branch, the 47th Conference on the Graduation Thesis for Undergraduate Students, 2017, 921.
14. Ayano T, Hirofumi M. Tilt Adjustment to Measuring Instrument for Piping Inspection Robot, Transactions of the Japan Society of Mechanical Engineers. 2018; 185(1):1304.
15. Keita I. Hirofumi M., Hardware Design for the Contact Type Measuring Instrument, The Japan Society of Mechanical Engineers Chugoku-Shikoku Branch, the 52nd Conference on the Graduation Thesis for Undergraduate Students, 2022, 11b4.
16. Ibuki T. Hirofumi M., Data Reception of the Contact Type Measuring Instrument using BLE, Transactions of the Japan Society of Mechanical Engineers. 2022; 225(1):09a1.
17. Kohei S. Hirofumi M., Posture Measurement of a Robot using the Contact Type Measuring Instrument, Transactions of the Japan Society of Mechanical Engineers. 2022; 225(1):09a2.
18. Hirofumi M. A Contact Type Three Dimensional Position Measuring Instrument for Verification of a Piping Inspection Robot, *International Journal of Recent Technology and Engineering (IJRTE)*. 2022; 10(6):65-72.
19. Hirofumi M., Translational Calibration for Contact Type Three-Dimensional Position-Measuring Instruments, *International Journal of Recent Technology and Engineering (IJRTE)*. 2023; 11(5):9-16.
20. Hirofumi M. Calibration Considering the Direction of Rotation for Contact Type Three-Dimensional Position-Measuring Instruments, *International Journal of Recent Technology and Engineering (IJRTE)*. 2023; 12(3):10-19.
21. Hirofumi M. Impact of Tire Shape on Localization Accuracy in Piping Inspection Robots, *International Journal of Recent Technology and Engineering (IJRTE)*. 2024; 12(6):35-42.