# Automated Map Comparison using Non-invariant Fourier Descriptors

Submitted 5th April 2011; accepted 5th July 2011

*Robert Ouellette, Kotaro Hirasawa*

***Abstract:***
*In this paper, we use non-invariant Fourier descriptors to derive transformation variables which allow us to optimally localize and reorient robot-generated maps based on the map shapes in order to determine, in an automated way, the accuracy of the generated maps. Our method uses only 4 simple calculations for alignment, therefore is extremely fast and gives a very good optimization for data maps that contain consistent, high frequency noise. A drawback to this method is occlusions in the map which affect the low frequency Fourier descriptors and cause localization and orientation errors. Preprocessing and optimization can help minimize these drawbacks. This application can be easily adapted to other areas such as image comparison or fault detection.*

***Keywords****: Fourier descriptors, robot mapping, SLAM, map comparison, image comparison*

## 1. Introduction

Simultaneous localization and mapping (SLAM) provides an effective way of helping robots localize themselves within a known environment or during an ongoing mapping operation [1]-[3]. One way of determining how well generated maps (a.k.a. "data maps") reflect the area they map is to compare them to a model map by digitally superimposing the data map with the model map in such an orientation that the optimal amount of black pixels gets covered. In this paper, we present a new technique for automating this type of alignment.

Until now, map alignment has been done by manually manipulating the maps such that superimpositioning of the maps gives the best visual match. Validating maps this way is painstaking and time consuming [4] therefore it would be advantageous to automate the process. To do so, some problems must first be overcome. One problem is that simply maximizing the number of black pixels covered (in this paper, both data and model maps are assumed to be black and white images – the map lined in black against a white background) does not guarantee that the map has been oriented optimally. Another issue to consider is that when a map is automatically generated such as that done by mapping robots, no specific orientation or reference point gets defined which in turn presents a lack of a common reference point from which the software would be able to relate maps to one another. Instead, we propose that the shape of the map's boundary is enough to quantify the map's orientation, position, and scale, which provides us with a way to relate maps to one another and allows us to calculate the differences between them.

The technique we use to quantify map shapes is done in the frequency domain using Fourier Descriptors [5]. Fourier descriptors have been used extensively in shape-based image recognition [5]-[12]. They allow a simple method of image normalization [12] and, when doing so, if the Euclidean distance between the Fourier descriptors of the two image borders falls within a given value, the images are considered the same. In this research however, we pre-assume that the maps themselves are already the "same" regardless of the quality of the data map and, rather than normalize the images, we adapt the normalization techniques in order to determine the values of the transformation variables needed to reorient the images for superimposition. At that point we can go about measuring how *accurate* the data maps are – not how *similar* they are to the model map.

## 2. Fourier descriptors

The concept of Fourier descriptors were first introduced by Zahn and Roskies [5] in 1972 and are simply the result of taking the discrete Fourier transform (DFT) [13] of a closed boundary. Zahn and Roskies [5] used the Cartesian version of the DFT but the method we use is based on the complex interpretation as used by Granlund [6]. This is done by first expressing the points on the boundary as two parametric equations:

$$x(n) = x_n,\ n = 0,\ 1,\ 2,\ \ldots,N-1 \tag{2.1}$$

$$y(n) = y_n, n = 0,\ 1,\ 2,\ \ldots,N-1. \tag{2.2}$$

Now, if we consider the points to lie in the complex plane, we can combine the two parametric equations above into a single equation:

$$s(n) = x(n) + iy(n) \ \text{ for } n = 0,\ 1,\ 2,\ldots,\ N\text{-}1. \tag{2.3}$$

Using a parametric description in this way allows us to reduce the dimensionality of the boundary without loss of information. Using a complex interpretation also simplifies some of the math needed to solve for the transformation variables. Once the data is expressed in this form, we can use the complex version of the 1D DFT on (2.3):

$$a(u) = \sum_{n=0}^{N-1} s(n)e^{-i2\pi un/N} \ \text{ for } u = 0,\ 1,\ 2,\ldots,\ N\text{-}1, \tag{2.4}$$

The complex coefficients a(u) are the Fourier descriptors of the boundary and are exactly the same as the Fourier coefficients produced by a DFT, with the

only stipulation being that the DFT be done on a closed boundary. The dual of (2.4) is:

$$s(n) = \frac{1}{N}\sum_{u=0}^{N-1} a(u)e^{i2\pi un/N} \text{ for } n = 0,\ 1,\ 2,...,\ N\text{-}1, \qquad (2.5)$$

which can be used to regenerate the original sequence of points.

## 3. FD-based map alignment

In image processing, a variety of transformations exist that transform an image from one form to another [14], [15]. Since image boundaries are part of the images themselves, transformations of an image cause the boundaries to be transformed the same way. Conversely, using the same transformation values that were used to transform a border, when applied to the image body, will cause the image to be transformed in the same way as well. In this paper we will be realigning the data map to match the model map by first finding the transformation values that allow us to align their borders. Once these transformation values are found, we can use the same transformation values that we used to align the borders to align the maps themselves.

Certain transformations on a boundary produce known relationships on their respective Fourier descriptors [14], [15]. Table I includes four of these: rotation (θ), scaling (α), translation (Δxy), and start point (n0). Spatially, the point from which boundary point trace begins is irrelevant; eventually the border will be completely recreated once the trace returns to that first point at the end of the trace cycle. However, in the frequency domain, since the sinusoids that make up the border are all dependent on the point that they are referenced from, starting from a different point will cause a phase shift in the sinusoids. Therefore, for boundary-related transformations, the first point that the trace begins from (i.e. the "start point") is critical to a correct border reconstruction, thus we also include it in this paper. Notably, non-uniform affine transformations (i.e. "shear") also have a chance of occurring during the mapping process but they would likely only occur due to severe problems with the hardware and/or software thus we have omitted them from this discussion.

Any transformation from one location/orientation to another can be minimized to a combination of one of each of the transformations listed in Table I. The amount of each transformation is dependent on the transformation sequence taken [reference analytic, image processing, robotics], but in this paper we set the sequence to be: rotation→scaling→start point→translation. Our reasoning for using this sequence is that it gives us an easy way to determine the respective transformation values for a transformed image. Combining each of the transformation equations in Table I in the above sequence we get,

$$\tilde{a}(u) = e^{i\theta}\alpha e^{-i2\pi un_0/N}a(u) + N\Delta_{xy}\delta(u). \qquad (3.1)$$

where $\tilde{a}(u)$ are the FDs of the transformed boundary.

By exploiting the relationships in Table I, we can algebraically manipulate the FDs of a map boundary and a transformed version of itself in such a way that we can find the values of the transformation variables of (3.1) using the equations summarized in (3.2)-(3.5).

$$\theta = i\ln\left(\frac{[a(p)]^2\tilde{a}(q)|\tilde{a}(p)|}{[\tilde{a}(p)]^2 a(q)|a(p)|}\right) \qquad (3.2)$$

$$\alpha = \frac{|\tilde{a}(u)|}{|a(u)|} \qquad (3.3)$$

$$n_0 = \frac{-iN}{2\pi(m-k)}\ln\left(\frac{\tilde{a}(k)a(m)}{\tilde{a}(m)a(k)}\right) \qquad (3.4)$$

$$\Delta_{xy} = \frac{1}{N}(\tilde{a}(0) - \alpha e^{i\theta}a(0)) \qquad (3.5)$$

where a indicates the FDs of the original border, the FDs of the transformed border, Δx=Re(Δxy), Δy=Im(Δxy), and k, m, p, q, and u are FD indices. Choosing which FDs to use in the equations can be done almost completely arbitrarily; however, derivation of the equations required that:

q=2p; p, q ≠ 0
α is real & positive
k ≠m; k, m ≠ 0.

While data maps and model maps refer to the same physical area, they are technically different, particularly from the software's point of view. In order to be able to relate the maps in software, we stipulate that data maps are copies of the model maps that have been transformed by some unknown amount and have been infused with noise. In the case that the data map were a perfect copy of the model map with the inclusion of random, high frequency noise, the FDs of the borders of both maps would be the same for all the FDs except the highest order FDs. It follows that we could determine the transformation values between the data and model map borders using only the lower-ordered FDs. This makes sense because,

*Table I. Spatial-Frequency Transformation Relationship*

| Type | Spatial Domain (boundary) | Frequency Domain (Fourier Descriptors) |
|---|---|---|
| Identity | $s_i(n)$ | $a_i(u)$ |
| Rotation | $s_r(u) = s(n)e^{i\theta}$ | $a_r(u) = a(u)e^{i\theta}$ |
| Scaling | $s_s(n) = \alpha s(n)$ | $a_s(u) = \alpha a(u)$ |
| Start point | $s_{sp}(n) = s(n - n_0)$ | $a_{sp}(u) = a(u)e^{-i2\pi un_0/N}$ |
| Translation | $s_t(n) = s(n) + \Delta_{xy}$ † | $a_t(u) = a(u) + N\Delta_{xy}\delta(u)$ ‡ |

*Transformation types: rotation (θ), scaling (α), change in the starting point ($n_0$), and translation ($\Delta_{xy}$).*

† $\Delta_{xy} = \Delta x + i\Delta y$ ‡ $\delta(u) = \begin{cases} 0, u \neq 0 \\ 1, u = 0 \end{cases}$

as indicated above, the lower-ordered FDs capture the location, orientation, and shape. Higher-ordered FDs capture things like noise. So, in this paper, rather than choosing the FD indices mostly arbitrarily, we select the first four FDs, FD(0)-FD(3) instead. Our reasoning for choosing is based in the geometric interpretation of the FDs. References [7] and [16] gives some insight into this reasoning.

Given that we can calculate the transformation values from the FDs of a transformed boundary, and given the relationships between transformations on FDs and on the boundaries themselves, we can transform the border back to its original location by:

$$\tilde{s}(n) = \frac{1}{\alpha}(s(n) - N\Delta_{xy})e^{i2\pi n n_0/N}e^{-i\theta}. \quad (3.6)$$

Since we are comparing the maps themselves and not the borders, we can ignore the start point part of the equation and use simply the inverse transform,

$$\tilde{s}(n) = \frac{1}{\alpha}(s(n) - N\Delta_{xy})e^{-i\theta}, \quad (3.7)$$

and use the non-complex version,

$$\tilde{x} = \mathrm{Re}[\tilde{s}(n)],\ \tilde{y} = \mathrm{Im}[\tilde{s}(n)], \quad (3.8)$$

in order to recreate the whole map.

## 4. Map comparison automation

In Fig. 1 we outline the steps taken during map comparison. The potential exists for there to be breaks in continuity in the model boundary as well as "islands" such as objects within the map so preprocessing the model and test maps entails making sure that the map outline has no breaks. We could automate the closing of gaps, however, considering that we only needed a few maps for validation, we found it was sufficient to fill in any gaps by hand. Islands are eliminated by using only the outermost boundary of the map and ignoring all others.

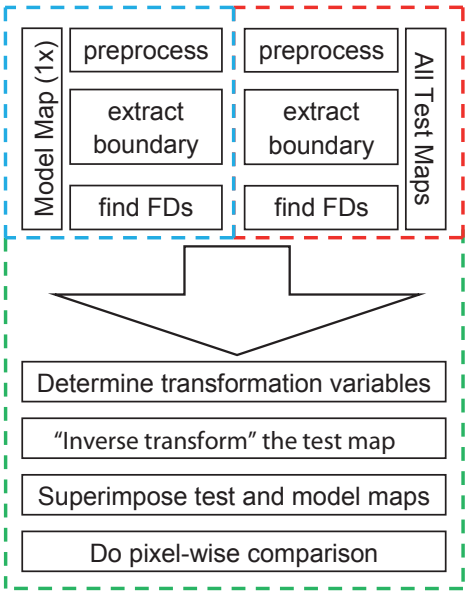


*Figure 1. Procedure used in map comparison automation*

We use the Moore boundary tracking algorithm [18] as described by [14] for finding the boundary of the model map and test maps. In practice, we should only have to do this with the model map since our sensor of choice, the laser scanner, by its very nature, gives us a discretized version of the boundary for the map. Below we summarize the Moore tracking algorithm.

1. Start in the upper left-hand corner of the map. Scan the map from left to right, top to bottom until you reach the first black pixel in the map. Denote the location of that pixel as the start point, b0.
2. Define c0 as the "west" neighbor of b0 (the pixel to the immediate left of b0). Starting at c0, examine the neighbors of b0 while moving in a clockwise direction. Record the position of the first non-zero pixel as b1 and register this location as the second point on the boundary. Let c1 be the point immediately preceding b1 in the sequence. Store b1and b0 for later reference. If no neighbor is found, the pixel is a singular "pixel island".
3. Let b = b1, c = c1.
4. Starting at c, examine the neighbors of b in a clockwise direction, like above until the first black pixel is found. Label the location for this pixel as b and the pixel just before it in the search sequence as c.
5. Register b as the next point on the boundary and use this as the next evaluation pixel.
6. Repeat steps 4 and 5 until b=b0 and the next boundary point found is b1

Unsupervised automation of border extraction does not guarantee that the first black pixel you find will be on the border you want to compare. Objects within or outside the map outline itself such as simple noise or a map legend could theoretically cause the software to find an unintended boundary. To minimize this potential problem, we found that finding all the borders in the map and choosing the border that contains the largest area to be that which worked best. This method may or may not solve all problems, but worked fine in our limited case.

Once the border is found, the remaining steps are mostly straightforward. Finding the Fourier descriptors of the maps, extracting the transformation variables, and performing the inverse transform on the test maps are as described in the previous sections. It is important to reiterate that we are not "inverse-transforming" the test map border, instead we are reorienting the whole data map using the extracted transform variables in order to compare all portions of the maps.

## 5. Results

Validation of our approach was done in two steps. In the first step, we transformed the boundary of the model map shown in Fig. 2 using known transformation values. Each type of transformation was applied individually as well as a rotation→ scaling→ start point→ translation transformation. The different transformations are superimposed with the model boundary in Fig. 3. Since we know by how much each of the transformed boundaries were transformed, when we extract the variables from

the transformed boundaries, the resultant extracted variables should be the same as the original transformation. The results, shown in Table II, verify that there is no error from the extracted variables using this "fixed" data method. We confirmed this visually by superimposing the inverse-transformed data boundary with the model boundary as shown in Fig. 4, showing that the boundaries match overall. Fig. 5 gives a zoom-in of the upper left-hand corner of Fig. 4, verifying that the boundaries match at the smallest scale as well.

The second step in validating our method involved applying our method to hand-generated, "raw" data. The data was acquired by hand-tracing a scanned image of the model map. Doing so ensured a reasonably well matching data map with enough regular noise to simulate a robot-produced closed-loop data map. During the scanning phase, the image was simultaneously given a random orientation and was scaled down to 75 percent (reference) of its original size. The hand-traced map and its pre-processed boundary are shown in Fig. 6 and the hand-traced boundary superimposed with the model map is shown in Fig. 7.

Since the data map is a different size (smaller in this case) than the model map, Moore-tracing the boundary will give a different amount of data points, therefore it is necessary to resample either the model boundary or the data boundary so that the number of data points are the same. We chose to resample the model map since it was larger than the data map. Resampling the data map would have entailed splitting the distance between neighboring pixels which can be done easily mathematically, but is not so easy to imagine visually. We used a "quick and dirty" resampling method to resample the border as given in (7.1):

$$\tilde{s}[n] = s[\tilde{n}] + (n\alpha - \tilde{n})(s[\tilde{n}+1] - s[\tilde{n}]) \qquad (5.1)$$

where

$$\alpha = \frac{size(\tilde{s}(n))}{size(s(n))},\ \tilde{n} = \text{floor}(\alpha n),$$

and is after being resampled.

This gives a regular sampling period with only the distance from the last point to the first point not necessarily matching the sampling distance. This method was sufficient for our purposes.

Following the algorithm outlined in Fig. 1, we did an inverse transformation using the transformation variables extracted from the hand-traced test data. The results are shown in Fig. 8 which shows a very good

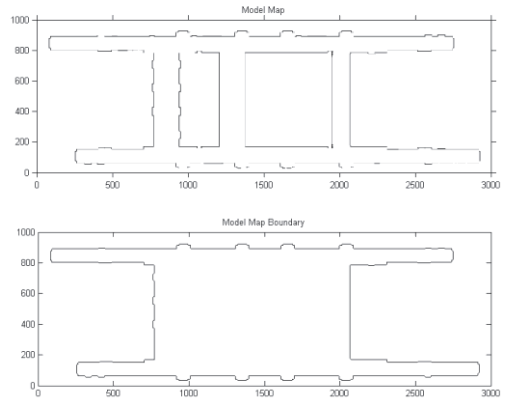


*Figure 2. Model map (top) and its boundary (bottom)*

match between the hand-generated data boundary and the model map boundary. There is, of course, some deviation as shown in the close up in Fig. 9, but this is simply due to the original freehand-generated error. This type of error will be present in any real-world environment.

Even though we could clearly see that the map borders match, our goal was to compare the actual maps. Therefore, we applied the transformation variables extracted from the border of the hand-traced map to the actual data map and superimposed the data map on top of the model map as shown in Fig. 10. As expected, we see that, like the border comparison above, the data map (red pixels) and the model map (black pixels) match well with errors only due to tracing as shown in Fig. 11. All image transformations and variable extractions were done using Matlab. The Fourier descriptors themselves were generated using the DIPUM 1.1.3 package for Matlab.

## 6. Discussion

As shown in the results above, this method has the potential to work very well. The maps we used were convenient in that the data maps were, in essence, exactly the same map as the model map with differences only in regular, high frequency noise which has little effect on the transformation variables themselves. In reality, a robot mapping a building has to deal with open or closed doors, obstacles, etc, which, in turn, if taken by themselves, would affect the low frequency

*Table II. Individual and complete transformation validation error results*

| Transformation Type | Transformation Amount | Error after Inverse Transform | | | |
|---|---|---|---|---|---|
| | | α | $n_0$ | θ | $\Delta_{xy}$ |
| Scaling | α =2.2 $n_0$ = 00 θ = 2.2 Δxy = 0 | 0.00+0.00i | 0.00+0.00i | 0.00+0.00i | 0.00+0.00i |
| Shifted Start Point | α =0 $n_0$ = 4575 θ = 0 Δxy = 0 | 0.00+0.00i | 0.00-0.00i | 0.00+0.00i | 0.00+0.00i |
| Rotation | α =0 $n_0$ = 00 θ = 1.14 Δxy = 0 | 0.00+0.00i | 0.00+0.00i | 0.00+0.00i | 0.00+0.00i |
| Translation | α =0 $n_0$ = 0 θ = 0 Δxy = 440+560i | 0.00+0.00i | 0.00+0.00i | 0.00+0.00i | 0.00+0.00i |
| Rot.-Scaling-S.P.-Translation | α =2.2 $n_0$ = 4575 θ = 1.14 Δxy = 440+560i | 0.00+0.00i | 0.00-0.00i | 0.00+0.00i | 0.00+0.00i |

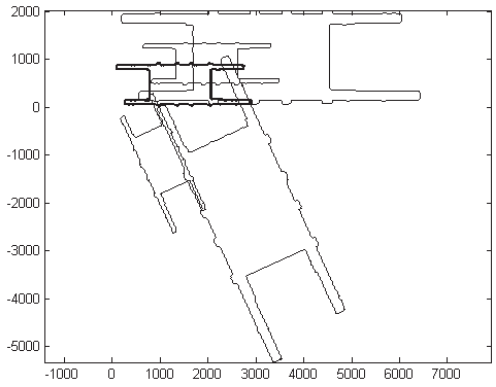

*Figure 3. Original boundary (in bold) and the boundary after undergoing various transformations*

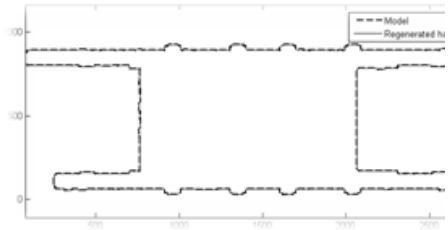


*Figure 4. Model map boundary (bold, hyphenated) and the data boundary after being inverse-transformed (solid line)*

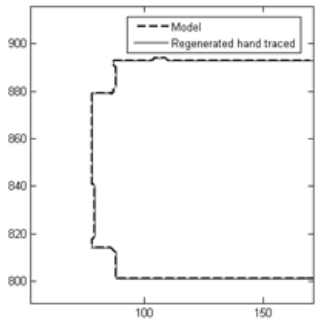


*Figure 5. The top-left corner of Figure 4 zoomed to show that there is no discernible variation from the original boundary and the inversed-transformed boundary*

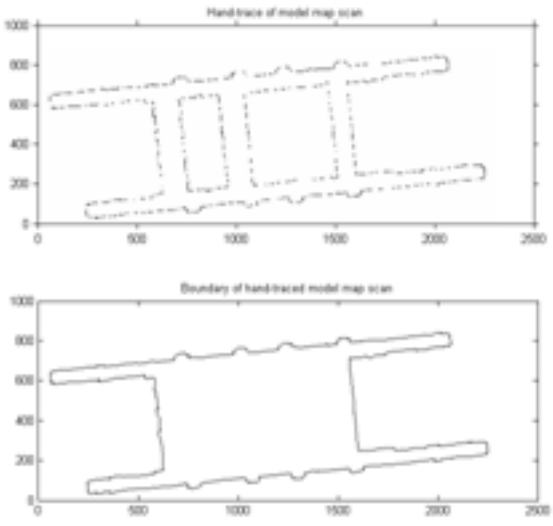

*Figure 6. Hand-tracing of model map (above) and its border (below)*

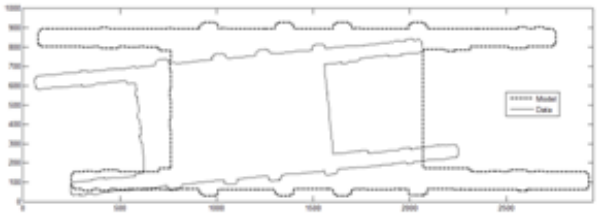


*Figure 7. Boundary of hand-traced map in its raw state superimposed over the model map boundary*

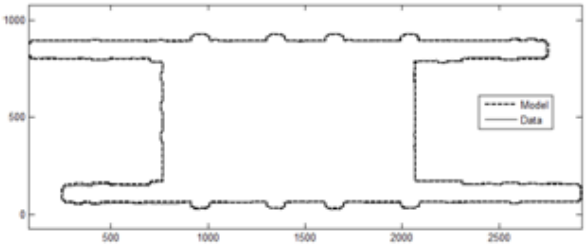


*Figure 8. The data map boundary (solid) after undergoing an inverse transformation and superimposed against the model map boundary (bold, hyphenated)*

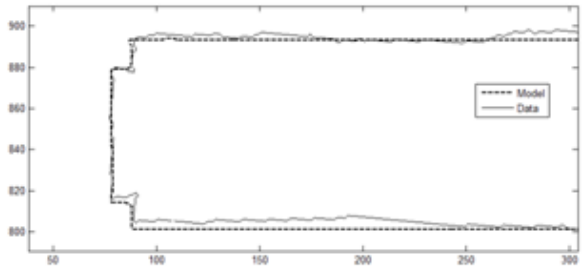


*Figure 9. The top-left corner of Figure 8 zoomed in to show the slight deviation from the model boundary due to the error given by drawing by hand*

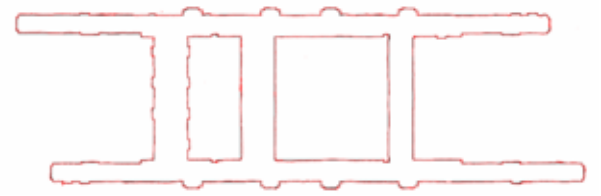

*Figure 10. The data map inverse-transformed (red) from the transformation variables extracted from the hand-traced border shown in Figure 6*

*Figure 11. The top-left corner of Figure 10 zoomed in to show the slight deviation from the model boundary due to the error given by tracing the map by freehand*

descriptors and would most certainly cause the center of mass of the map border to be off, and, almost as certainly, cause the orientation to be off as well. This issue can be minimized in practice because in order to compare a data map with a model map (manually or automatically), we already have to preprocess both the data map and model map to some degree. During that preprocessing, it is easy to ensure that no extra rooms exist due to opened doors, etc.

An issue also exists with the derivation of the transformation variables. We use only the first four Fourier descriptors to determine the transformation variables. Doing so greatly reduces the number of computations to roughly 4N where N is the number of boundary points. This method is much faster than a regular Fourier transform or even the Fast Fourier Transform however it only allows the capture of information in the very lowest frequencies. It does capture general shape and orientation, but it does not consider information from the higher frequencies. Thus, a better method might be to calculate a few values of orientation, scaling, etc, based on more, or even all, Fourier descriptors and take the Euclidean distance which gives the best matching value. We have considered this, but our primary objective with this paper is to show that this approach will give very good results using even the simplest approach. Furthermore, it gives a good balance between speed and accuracy. Any optimization beyond this needs to consider preprocessing, the number of boundary points (i.e. the sampling frequency), the number of Fourier descriptors used to calculate the transformation variables, the number of calculations involved, and the number of maps to be compared, among other things and should be addressed as a topic by itself.

As mentioned above, any noise in this approach should be regular and high in frequency. Figure 9 presents a good example of such noise as well as the addition of low frequency noise. Although the high frequency noise is relatively low in amplitude with respect to the size of the map, the noise is clearly seen in the "roughness" of the data points superimposed about the model map. The key is that regardless whether the noise is high in amplitude or not, the algorithm will always average the high frequency noise out through the low frequency-based transformation calculations.

Lastly, it is important to mention that in this paper we assume that the data maps and the model maps are maps of the same location. If, for some reason (e.g. by mistake), the data map and the model map are maps of two different places, our method will still force the maps to be localized and orientated in the optimal position for comparison, even though if it the result is nonsensical. So while this method will align two maps regardless if they are different or not, it will not tell us if the maps are actually the same. In order to do so, we could take the Euclidean distance between the normalized Fourier descriptors of the data map and model map where the smaller the error would indicate the better the match. The details for similarity matching are outside the scope of this paper, but many of the other references listed in this paper such as [5]-[12] discuss this topic in depth.

## 7. Conclusion and Future Work

In this paper we extracted transformation variables based on information derived from the shape of model and data map boundaries then used those variables to align the data and model maps so that the data map can be checked for accuracy. This method is very fast as it only needs four simple calculations to determine the values of the transformation variables in the most basic alignment approach. Optimization needs to be addressed for accuracy-constrained applications, but overall, the methods we developed here are very useful for automating map comparison. While we limited the scope of our discussion to map alignment, it is easy to realize the many different areas that this can be applied to such as image comparison, defect detection, or object tracking, to name a few. In future work, we will apply these methods in a piecewise sense to perform robot localization and mapping.


### Authors:

**R. P. Ouellette*** is a robotics consultant at Open Thoughts Research. He has held positions in embedded hardware & software, signal processing and robotics research in various companies such as Raytheon, GMD-Japan (now Fraunhofer FhG), The Foxboro Company, among others. He has a BS in Engineering Physics from the University of Maine, an MS in Electrical & Computer Engineering from Kyushu Institute of Technology, and is currently pursuing his PhD in Robotics at Waseda University. (phone: +81-90-1191-1514; e-mail: rpo@ openthoughts.com).

**Kotaro Hirasawa** received the B.S. and M.S. degrees from the Kyushu University, Japan, in 1964 and 1966, respectively. From 1966 to 1992, he worked at Hitachi Ltd. at the Hitachi Research Laboratory. From December 1992 to August 2002, he was a Professor at the Graduate School of Information Science and Electrical Engineering of Kyushu University. Since September 2002, he has been a Professor at the Graduate School of Information, Production and Systems, Waseda University. Dr. Hirasawa is a member of the Society of Instrument and Control Engineers, the Institute of Electrical Engineers of Japan and IEEE.

*Corresponding author.


## References


1. A. Eliazar, R. Parr, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks". In: *Proc. 18th Int. Joint Conf. on Artificial Intelligence* (IJCAI'03).
2. A. Nüchter, H. Surmann, K. Lingemann, J. Hertzberg, S. Thrun, "6D SLAM with an Application in autonomous mine mapping". In: *Proc. of the IEEE International Conference on Robotics and Automation*, New Orleans, USA, April 2004, pp. 1998-2003.

3. W.Y. Jeong, K.M. Lee, "Visual SLAM with Line and Corner Features". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006, pp. 2570-2575.
4. R. Ouellette, K. Hirasawa, "A comparison of SLAM Implementations for Indoor Mobile Robots". In: Proc. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2007)*, Oct. 2007, pp. 1479-1484.
5. C. T. Zahn, R. Z. Roskies, "Fourier Descriptors for Plane Closed Curves", *IEEE Transactions on Computers*, vol. 21, 1972, pp. 269-281.
6. G. H. Granlund, "Fourier Preprocessing for Hand Print Character Recognition", *IEEE Transactions on Computers*, vol. 21, 1972, pp. 195-201.
7. O. Bertrand, R. Queval, H. Maitre, "Shape Interpolation using Fourier Descriptors with Application to Animation Graphics", *Signal Processing*, vol. 4, 1982, pp. 53-58.
8. D. S. Zhang, G. Lu. "A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures". In: *Proc. of the International Conference on Intelligent Multimedia and Distance Education (ICIMADE'01)*, Fargo, ND, USA, June 1-3, 2001, pp.1-9.
9. L. Keyes, A.C. Winstanley, "Fourier Descriptors as a General Classification Tool for Topographic Shapes". In: *Proc. of the Irish Machine Vision and Image Processing Conference (IMVIP '99)*, Dublin City University, 1999, pp. 193-203.
10. Y. Rui, A. C. She, T. Huang, "A Modified Fourier Descriptor for Shape Matching in MARS". In: S. K. Chang (ed.), *Image Databases and Multimedia Search*, Series of Software Engineering and Knowledge Engineering, World Scientific Publishing, 1998, pp. 165-180.
11. D.J. Lee, S. Antani, L. Rodney Long, "Similarity Measurement Using Polygon Curve Representation and Fourier Descriptors for Shape-based Vertebral Image Retrieval", *Journal of Visual Communication and Image Representation*, vol. 15, no. 3, Sept. 2004, pp. 285-302.
12. C. S. Lin, C. L. Hwang, "New Forms of Shape Invariants from Elliptic Fourier Descriptors", *Pattern Recognition*, vol. 20, no. 5, 1987, pp. 535-545.
13. J. W. Cooley, J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", *Math. Comput.*, 19, 1965, pp. 297-301.
14. R. C. Gonzalez, R. E. Woods, *Digital Image Processing*, 3rd ed., Pearson Education, Inc., 2008.
15. A. K. Jain, Fundamentals of Digital Image Processing, Pearson Education, Inc., 1989.
16. B. Jahne, *Digital Image Processing*, 6th ed., Springer, 2005.
17. A. Oppenheim, R. Schafer, J. Buck, *Discrete-time Signal Processing*, 2nd ed., Prentice Hall, 1998.
18. G. A Moore, "Automatic Scanning and Computer Processes for the Quantitative Analysis of Micrographs and Equivalent Subjects". In: C. G. Cheng *et al.* (ed.), *Pictorial Pattern Recognition*, 1968, pp. 275-326.