Rail Extraction for Driver Support in Railways

Bogdan Tomoyuki Nassu and Masato Ukai
Signalling and Telecommunications Technology Division
Railway Technical Research Institute, Tokyo, Japan
{bogdan,ukai}@rtri.or.jp

Abstract— Rail extraction is one of the basic tasks for video-based driver support in railways, being used, for example, to determine regions of interest for detecting obstacles and signs. In this paper, we introduce an approach that performs rail extraction by matching edge features to candidate rail patterns modeled as sequences of parabola segments. Patterns are pre-computed for areas near the camera and generated on the fly for more distant regions. The proposed approach addresses the challenges posed by the open and complex environment, such as bad visibility, shadows, motion blur, clutter, occlusion, and confusing structures. Moreover, it does not require explicit knowledge about train speed or camera position, running fast enough for practical use without specialized hardware. The feasibility of the proposed approach was verified in experiments using videos captured under real operation conditions.

I. INTRODUCTION

Automation based on infrastructure has been used for improving railway safety for many years. However, some tasks still depend heavily on the perception of the driver, such as detecting obstacles on the rails. In-vehicle sensor-based driver support systems can become a tool for improving safety in such situations, in a manner similar to what has been done for automobiles in recent years [1], [2].

Rail extraction is one of the fundamental tasks for sensor-based driver support in railways. It consists in determining the position of the rails, and is used, for example, to delimit regions of interest for detecting obstacles and signs. Some approaches achieve this by placing specialized equipment along the tracks [3], but purely visual approaches [4], [5], [6] are more general and can reduce installation and maintenance costs. In this paper, we address the problem considering sequences of images taken by a fixed in-vehicle camera.

Ideally, the rails appear as a pair of long, bright stripes with high contrast with their surroundings. However, as trains operate in open and complex environments, many challenges arise (see Fig. 1). Bad illumination or visibility, motion blur, shadows, rust, occlusion and poor contrast with the trackbed can make the rails appear as heterogeneous or discontinuous stripes with unclear boundaries. Moreover, clutter and confusing structures such as multiple tracks can make it difficult to interpret the image.

To cope with the above mentioned challenges, we propose a new approach for rail extraction. The setting makes segmentation based on color or texture very hard, so we focus on locating rail shapes in the image. The Chamfer distance [7] is used as a similarity metric for matching edge features to candidate rail patterns, which are modeled as sequences of parabola segments, as shown in Fig. 2. This model can represent more complex shapes than a single parabola or clothoid. For a region near the camera, pre-computed patterns such as those shown in Fig. 3 are used. This is possible because the appearance of the rails changes only to a limited degree up to a certain distance from the camera. For more distant areas, the patterns are generated on the fly, based on the results obtained for the short distance. The sequential nature of videos is explored for robustness and reduced processing time. Our approach does not require explicit knowledge about train speed or camera parameters/position, and runs fast enough for practical use without specialized hardware. Its feasibility was verified in experiments using videos captured under real operation conditions, including a comparison to an existing method [6].

The remainder of this paper is as follows. Related work is discussed in section II. The proposed approach for rail extraction is detailed in section III. Section IV describes the experiments, and conclusions are presented in section V.
different assumptions can lead to better results. Extraction [4], [5], using similar principles but relying on identification techniques can be directly employed for rail lane markings. These differences imply that, although lane the rails in the long distance. On the other hand, in areas identification, preventing the use of common models such longer distances than needed by most applications of lane equipment. Moreover, rails usually must be detected in roads, containing ballast, sleepers, vegetation and railway assuming the rails have strong gradients and are almost straight near the camera, the vanishing point and rail gauge are estimated by a Hough line transform. This information is used along with problem-specific knowledge to build a cost matrix for locating the rails using dynamic programming. Although it works well for some settings, that approach can handle curves only to a limited degree, does not explore temporal information, and is confused by strong gradients near the rails. In section IV, we show that our approach compares favorably to this alternative.

Rail extraction has many characteristics in common with the identification of lane markings in highways. This is a widely studied problem for which many solutions have been proposed [1], [2], [9], usually considering image features that are extracted, selected and grouped based on domain-specific assumptions. The same way as rails, lane markings appear as groups of stripes with smooth and slowly changing curvature. Many of the challenges imposed by the environment are also identical in both cases. However, some assumptions that are valid for roads do not hold for railways, and vice-versa. For example, it is a common assumption that lane markings are brighter than the asphalt, but different parts of the rails can appear brighter or darker than their surroundings, sometimes with unclear boundaries. Assumptions about the asphalt having near uniform color or texture also find no parallel in railways: trackbeds tend to be more richly textured than roads, containing ballast, sleepers, vegetation and railway equipment. Moreover, rails usually must be detected in longer distances than needed by most applications of lane identification, preventing the use of common models such as parabolas or clothoids, which are unable to represent the rails in the long distance. On the other hand, in areas near the camera rails show less variation in appearance than lane markings. These differences imply that, although lane identification techniques can be directly employed for rail extraction [4], [5], using similar principles but relying on different assumptions can lead to better results.

II. RELATED WORK

Compared to conventional infrastructure-based automation, video-based monitoring for railways is still a new and unexplored area. There are works on monitoring using static [8] and in-vehicle [4], [5] cameras, but literature dealing specifically with rail extraction is scarce. We highlight the approach from [6]. Assuming the rails have strong gradients and are almost straight near the camera, the vanishing point and rail gauge are estimated by a Hough line transform. This information is used along with problem-specific knowledge to build a cost matrix for locating the rails using dynamic programming. Although it works well for some settings, that approach can handle curves only to a limited degree, does not explore temporal information, and is confused by strong gradients near the rails. In section IV, we show that our approach compares favorably to this alternative.

Rail extraction has many characteristics in common with the identification of lane markings in highways. This is a widely studied problem for which many solutions have been proposed [1], [2], [9], usually considering image features that are extracted, selected and grouped based on domain-specific assumptions. The same way as rails, lane markings appear as groups of stripes with smooth and slowly changing curvature. Many of the challenges imposed by the environment are also identical in both cases. However, some assumptions that are valid for roads do not hold for railways, and vice-versa. For example, it is a common assumption that lane markings are brighter than the asphalt, but different parts of the rails can appear brighter or darker than their surroundings, sometimes with unclear boundaries. Assumptions about the asphalt having near uniform color or texture also find no parallel in railways: trackbeds tend to be more richly textured than roads, containing ballast, sleepers, vegetation and railway equipment. Moreover, rails usually must be detected in longer distances than needed by most applications of lane identification, preventing the use of common models such as parabolas or clothoids, which are unable to represent the rails in the long distance. On the other hand, in areas near the camera rails show less variation in appearance than lane markings. These differences imply that, although lane identification techniques can be directly employed for rail extraction [4], [5], using similar principles but relying on different assumptions can lead to better results.

III. PROPOSED APPROACH FOR RAIL EXTRACTION

Take the following problem-specific assumptions:
• Image areas containing rails usually respond strongly to filters such as those discussed in [9]. However, there can be parts of the rails for which this does not occur. Furthermore, other objects on the trackbed often produce stronger responses than the rails.
• Rails have smooth and slowly changing curvature. Their shapes in the image plane can be modeled as sequences of parabola segments, as shown in Fig. 2.
• The rails can be detected in an approximated manner, without a pixel-per-pixel classification.
• As the train is always on the rails and the camera is fixed inside the cabin, the appearance of the rails in a region near the camera varies to a limited degree.
• The rails are parallel. The perspective effect makes the distance between them (the rail gauge) in the image inversely proportional to the distance from the camera.
• The position and curvature of the rails do not change radically between two consecutive frames.
• Explicit knowledge about train speed or camera parameters/positioning is not available.

Based on the above assumptions, we propose a two-step approach for rail extraction. In both steps, edge features are extracted from the input image and matched to patterns, using the Chamfer distance [7] as a similarity metric. In the first step, rails are located in a region near the camera, with edges being matched to a set of pre-computed patterns (Fig. 3 shows some examples of patterns). In the second step, rails are extracted in more distant areas, with the patterns being generated on the fly, based on the results obtained during the first step. To determine the image area covered by each step, we take the region from the bottommost image row to an estimated horizon line, and consider the bottom and upper halves of this region in the first and second step, respectively.

Below, we describe the methods used for extracting edges and matching patterns, as well as the two rail extraction steps.

A. Extracting Edge Features

Edges are extracted in a framework similar to the the Canny edge detector: after a pre-smoothing stage, filter responses are computed and hysteresis thresholded with non-maxima suppression. Different configurations were tested, for both the short and the long distance steps. The pre-smoothing step can use Gaussian, mean or median filters, with a horizontal, vertical or bidirectional window with width between 1 (i.e. no smoothing) and 19. For computing filter responses, the following methods were tested:
• Sobel operator, with aperture size 1, 3, 5 or 7.
• Top-hat filter [9], with diameter 3 to 19. Multiple filters with different diameters can be used, with the maximum response being taken for each pixel.
• Difference-of-Gaussians/Means (DoG/DoM), with window sizes between 1 (no smoothing) and 19. The Local Threshold described in [9] is a particular case of DoM.
• Symmetrical local threshold (SLT) [9], width radius between 1 and 24.

Except for the Sobel operator, these methods respond positively to bright areas surrounded by darker areas, and negatively to the opposite situation. As rails can be brighter or darker than their surroundings, we have also tested
versions with inverted and absolute responses. Responses are computed only in the horizontal direction for the short distance, and in both directions for the long distance.

Bad illumination and clutter make it hard to set fixed thresholds for the filter responses. To handle these cases, a desired proportion of edge pixels is specified, and the thresholds are obtained dynamically based on the results from the previous frame. Take $0 < e_{min}, e_{max} < 1$ the minimum and maximum acceptable proportions for edge pixels, and $e$ and $T_0$ respectively the proportion of edge pixels and the threshold from the previous frame. The threshold $T$ for the current frame is set to $max(0, T_0 - 1)$ if $e < e_{min}, T_0 + 1$ if $e > e_{max}$, and $T_0$ otherwise. The upper hysteresis threshold is set to $T \cdot \alpha$, where $\alpha \geq 1$ is a given configuration parameter ($\alpha = 1$ results in common thresholding instead of hysteresis thresholding). For the first frame, $T$ is iteratively increased until the largest value so that $e \leq e_{max}$ is found.

B. Matching Patterns Using the Chamfer Distance

The edges extracted over the rails are often discontinuous, due to bad illumination/visibility, motion blur, shadows, rust, or occlusion. Moreover, most edges are detected over non-rail structures. That prevents the classification of individual pixels using purely local information. The proposed approach uses instead a high level model to approximate the rail shape, and a robust metric for matching it to the extracted edges. As illustrated in Fig. 2, each rail is modeled as a sequence of parabola segments: a single segment for the region near the camera and multiple segments for more distant regions. The model is composed by two such sequences. A candidate rail pattern is a partial or complete instance of the rail model.

To test how well a candidate matches the extracted edge image $I_e$, we create a pattern image $P$ by drawing the parabolas as edge maps, and compute the Chamfer distance [7] $d$ from $P$ to a sub-region of $I_e$ with the top-left corner at $(x, y)$. The Chamfer distance is defined as the average distance$^1$ from each edge pixel in $P$ to an edge pixel in $I_e$. When $d$ is smaller than a given maximum distance $M > 0$, we say there is a match between $P$ and $I_e$ at $(x, y)$. We normalize $d$ as a matching score $s = \frac{M - d}{M}$.

C. Locating Rails in the Short Distance

Detecting parabolas in a cluttered edge map and judging if they correspond to the rails is a processor intensive task. For that reason, instead of arbitrary parabolas, our approach uses pre-computed candidate patterns for locating rails in the short distance. The patterns implicitly encode information about camera parameters and positioning, rail gauge and curvature, and variations caused by train movement. That makes rail extraction not only faster, but also safer in a real setting, as the patterns can be validated before use. In our tests, we have used sets of up to 6 hand-defined patterns.

To match the candidate rail patterns to an image, we slide horizontally over the lower part of the extracted edge map a window with the same dimensions as the pattern images, computing the matching score for each pattern at each window position. The rails are segmented based on the best match. Figure 4 illustrates this concept.

The sequential nature of the input is explored for making rail extraction faster and more robust. In consecutive frames, the rails should have similar position and curvature, a restriction that becomes stronger as the confidence in the results increases. As shown in Fig. 5, this is explored by using two arrays with weights that range from 0 to 1: one for the position, and the other for the direction trend. If there were no matches in the previous frame, all the weights are set to 1. Otherwise, take $W$ the image width in pixels, and $s$ and $(x, y)$ the score and position of the best match from the previous frame. For the position array, we use a Gaussian function with $\sigma = W/8 \cdot (1 - s)^2$ and center at $x$. If in the previous frame there were no matches at columns $x$ and $x \pm 1$, the weight for $x$ is set to 0. As for the direction trend array, we use a Gaussian function with $\sigma = W/16 \cdot (1 - s)^2$, centered at the point between the rails at $y$. The matching scores are multiplied by the weights, so there is no need for computing them if any of the weights is 0.

D. Locating Rails in the Long Distance

Rail extraction in the long distance is more challenging than in the short distance: the complex rail shapes prevent the use of pre-computed patterns, and the smaller resolutions make it hard to discriminate between the rails and nearby structures. To tackle with these problems, we employ an

---

$^1$We have tested the Euclidean, Manhattan and Chebyshev distances.
The rules for each candidate pattern are as follows:

- The left and right rails are parallel, and never cross each other. For any $y$, if $x^l < x^r$, the weight is zero.
- The rail gauge in the image is inversely proportional to the distance from the camera. Given points $(x^l, y)$ and $(x^r, y)$, the weight is zero. Moreover, the gauge at $y_i$ can be estimated by $x^l - x^r = (y_0 - y_i) \cdot (x_0^l - x_0^r - x^l + x^r)/(y_0 - y_2)$. If the difference between this estimate and the gauge at $y_i$ is larger than a given threshold $G$, the weight is zero.
- Candidates near the estimated horizontal coordinates at $y_i$ have higher weight. The weights are given by two Gaussian functions centered at $x^l_i$ and $x^r_i$, with $\sigma = W/32$. These functions take as parameters the horizontal coordinates of the candidate pattern at $y_i$.
- The rails have similar shape and position in consecutive frames. For each rail, the weight is given by a Gaussian function centered at the horizontal coordinate of the rail at $y_i$ in the previous frame, with $\sigma = W/8 \cdot (1 - s)^2$, where $s$ is the best matching score from the previous frame. These functions take as parameters the horizontal coordinates of the candidate pattern at $y_i$.

At each iteration, a new level is created in the tree for candidate patterns with a weighted matching score $s > 0$. The vertical coordinate $y_i = y_i \cdot \gamma$, with $0 < \gamma \leq 1$, is used for the control points at the new level. A cumulative score is computed for each node, given by $s_i = (y_{i-1} - y_i)/s_{i-1}$, where $s_{i-1}$ is the score for the node’s parent (zero for the nodes in levels 0 to 2). To reduce the required processing time, before starting a new iteration, the tree is pruned so that only the two leaves with the best scores are kept (we have used $B = 4$). After all iterations were completed, the rails are segmented based on the candidate rail pattern that ends in the leaf node with the highest cumulative score.

### IV. Parameter Selection and Evaluation

The proposed approach is controlled by several parameters. The large number of possible configurations make it hard to select for each parameter values that work well under general conditions. For that reason, we have tested several thousand configurations using automatic methods based on ground truth data. These tests allowed us to verify the feasibility of the proposed approach. We have used a regular PC with 3GHz dual core processor and 2GB of RAM and a C++ implementation, achieving frame rates between 10 and 20 frames per second. This is fast enough for practical use, even though the implementation is not fully optimized.

The inputs are short video sequences captured under real operating conditions in several Japanese railways, typically at resolutions of 720x480 or 768x432 pixels. Various camera models and lenses were used. All the railway images shown in this paper were taken from the input videos. Illumination conditions are often not optimal, but they are always good enough so that a human observer can identify the approximated rail position and shape. Thus, although there is a sequence shot at night, it was taken in a fairly illuminated area; although there are several short tunnels, in none of them...
the image becomes completely black. Moreover, to know which path will be followed when a track bifurcates, one has to interpret signals or understand the switching mechanism. Thus, the input sequences do not include bifurcations — but level crossings and joining tracks are present.

The input videos are divided in three sets. The first set has 10 videos, with a total of 3,549 frames. The second set has 27 videos, with 14,474 frames. The third set has 12 videos, with 5,879 frames, and is a sub-set of the second set, including only the cases in which the rails are mostly visible in the long distance. Ground truth data was generated for all the frames by hand-segmenting the rails.

In the following sub-sections, we describe the experiments performed for the short and long distances, and compare the results to those obtained by an existing approach [6].

A. Parameter Selection and Evaluation (Short Distance)

Rail extraction in the short distance is controlled by:

- The pre-smoothing parameters, including the filter type, window size and orientation.
- The method for computing filter responses. Except for the Sobel operator, all methods have three versions: original, inverted and absolute.
- The acceptable proportions for edge pixels ($e_{min}$ and $e_{max}$) and the $\alpha$ parameter.
- The distance metric (Euclidean, Manhattan or Chebyshev) and the maximum Chamfer distance ($M$).

Over 200,000 parameter configurations were tested using an evolutionary optimization technique. Promising configurations were found in 50 runs of a specialized Genetic Algorithm [10] in which a population of 400 configurations evolves through many generations. Continuous parameters were discretized to a 0.01 precision. The fitness for a configuration is measured by comparing the results it obtains to the ground truth, with the precision and recall being computed for each frame from each video in the first set. We consider a “perfect” precision if at least 70% of the detected rail pixels were in the ground truth, and a “perfect” recall if at least 60% of the pixels from the ground truth were detected. The fitness for a configuration is given by the average F-measure from all the frames. All the configurations with fitness above 0.985 were then tested using the second video set. A final score was computed for each configuration, given by the average F-measure from all the frames from all the videos it the first and second video sets. The best 50 configurations were further optimized in a hill-climbing process.

Good configurations usually perform pre-smoothing with window size 11 or 9 for vertical Gaussian smoothing, and 3 or 5 for the other cases. All the methods for computing filter responses produced good results, but the absolute methods performed better than the equivalent original or inverted methods. This happens because the absolute methods detect rails both brighter and darker than their surroundings, sometimes also generating redundant edges over the shadows cast by the rails. This is usually acceptable, as the shadows and the rails are a few pixels apart, which translates to a few centimeters in actual space, and the additional clutter is compensated by a higher robustness. A similar observation can be made about the Sobel operator, which detects two step edges for each rail. As for the other parameters, the best configurations had $0.02 \leq e_{min} \leq 0.13$, $0.07 \leq e_{max} \leq 0.21$, $1 \leq \alpha \leq 3$, and $1.2 \leq M \leq 3.4$.

The best configuration had a score of 0.99942. It performs pre-smoothing with a 3x3 mean filter, computes filter responses using the absolute version of the SLT method with a radius of 2, and considers the Euclidean distance. Other parameters are: $e_{min} = 0.11$, $e_{max} = 0.19$, $\alpha = 2.0$ and $M = 1.5$. We have visually checked the results obtained by this configuration, and the approximated position of the rails was correctly found in all the tested frames — i.e. there were no cases in which the rails were not detected or detected far from their actual position.

B. Parameter Selection and Evaluation (Long Distance)

Rail extraction in the long distance is controlled by the same parameters used for the short distance, plus:

- $\beta$ and $\gamma$, which determine the sub-area to be tested and kept during each iteration from the algorithm.
- The maximum deviation from the estimated gauge ($G$).

Parameter selection for the long distance was based on an iterative hill-climbing approach that takes at each iteration the 5 best candidate configurations and generates for each one of them up to 784 variations, which become the candidates for the next iteration. Candidate solutions are tested by comparing the segmentation obtained for the long distance image region to the ground truth, with scores being computed as the average F-measure from all the frames from the 10 videos in the first set. We consider a “perfect” precision if at least 95% of the detected rail pixels were in the ground truth, and a “perfect” recall if at least 85% of the pixels from the ground truth were detected. The 15% topmost rows are ignored, as in very long distances it is difficult to precisely determine the rail position even for a human observer. For the short distance, the best configuration described in section IV-A was used. This process was executed independently 50 times, with over 70,000 configurations being tested.

We have visually analyzed the results obtained by 40 of the best candidate configurations for the videos from the first and third sets. Deviations from the actual rail positions were counted and classified according to these guidelines:

- A “not found” error is counted if the rails could not be found in the long distance. Note this is usually less severe than detecting the rails in an incorrect position.
- An “imprecision” is counted if the segmentation is slightly incorrect, but the rail shape and position are approximately correct. Imprecisions are tolerable for most applications.
- A “small” deviation is counted if the segmentation is correct for the most part, but deviates, pointing to an area near the trackbed. In a practical setting, this kind of error can lead to degraded results which, depending on the application, can be tolerable or not.
- A “medium” deviation is counted if the segmentation is partially correct, but deviates, pointing to an area
outside the trackbed. In a practical setting, this kind of error is usually unacceptable.

- A “severe” deviation is counted when the segmentation is mostly incorrect.

Good configurations have performed no pre-smoothing, or have used square or vertical windows with size 3. Different than observed for the short distance, the absolute methods for computing filter responses did not produce the best results: in long distances, the rails occupy a smaller part of the image, and the additional clutter becomes confusing. The best scores were obtained by configurations that use the Sobel operator or the inverted filter responses, which detect rails darker than their surroundings or their shadows. As for the other parameters, the best configurations had $0.08 \leq \epsilon_{\text{min}} \leq 0.14$, $0.10 \leq \epsilon_{\text{max}} \leq 0.18$, $1 \leq \alpha \leq 2$, and $2.9 \leq M \leq 4.8$, $0.3 \leq \beta \leq 0.5$, $0.5 \leq \gamma \leq 0.8$, and $2 \leq G \leq 7$.

The best configuration had a score of 0.94940, performing pre-smoothing with a 3x3 Gaussian filter, computing filter responses using the Sobel operator with a window size of 7, and considering the Euclidean distance. Other parameters are: $\epsilon_{\text{min}} = 0.10$, $\epsilon_{\text{max}} = 0.18$, $\alpha = 2.0$, $M = 3.1$, $\beta = 0.5$, $\gamma = 0.6$, and $G = 7$. Table I shows the number of frames containing deviations, classified by their severity. As in most cases the deviations occur not in isolated frames, but in sequences, the table also shows a number of error sequences, obtained by grouping deviations that occur less than 5 seconds apart, and classifying the group by the most severe error. These sequences are usually caused by a combination of factors: for example, the sequences containing small deviations occurred in situations that involved at the same time a curve, vertical train oscillations and one of the rails being occluded by grass. Rail position and shape were correctly approximated in about 98% of the frames.

### C. Comparison to an Existing Approach

For comparative purposes, we have tested the approach presented in [6] (discussed in section II). The parameters for each video were chosen so that the lower image region is always correctly detected by the Hough line transform. We have also tested a hybrid approach that replaces the Hough transform with the short distance detection step from our approach. These tests consider the first video set, with deviations being counted as described in section IV-B.

Table II shows the number of deviations obtained by the tested approaches. The approach from [6] is labeled “KA”. It was able to detect straight and slightly curved rails, but has failed for other curves, when the vanishing point is badly estimated by the Hough transform. Moreover, in many occasions it was confused by clutter or strong edges near the rails. The hybrid approach was able to better estimate the vanishing point, avoiding or reducing many of these errors. There were no severe deviations, but clutter and some curves still pose a challenge. Overall, the proposed approach performed considerably better than the tested alternatives.

### V. Conclusion and Future Work

We have introduced a new approach for rail extraction — one of the basic tasks for video-based driver support in railways. Our approach addresses the challenges created by the complex environments where trains operate. Using videos captured under real operation conditions, we have derived parameter configurations that perform well under general conditions, showing the proposed approach is feasible, running fast enough for practical use without specialized hardware, and outperforming an existing technique.

Future work shall address the automatic learning of rail patterns for the region near the camera. Moreover, rail extraction is not an end on itself, so the evaluation of the proposed approach as part of a larger system is also planned.

### REFERENCES


### TABLE I

<table>
<thead>
<tr>
<th>Frames with deviations and error sequences (long distance).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Frames</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>Imprecision</td>
</tr>
<tr>
<td>Small</td>
</tr>
<tr>
<td>Medium</td>
</tr>
<tr>
<td>Severe</td>
</tr>
<tr>
<td>Not Found</td>
</tr>
</tbody>
</table>

### TABLE II

<table>
<thead>
<tr>
<th>Comparison: number of frames with deviations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>KA ([6])</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>Imprecision</td>
</tr>
<tr>
<td>Small</td>
</tr>
<tr>
<td>Medium</td>
</tr>
<tr>
<td>Severe</td>
</tr>
<tr>
<td>Not Found</td>
</tr>
</tbody>
</table>