# LUVN-Tracing: Unique Viable-Neighbor based Contour Tracing

**D. S. Govard**[1] | **B. Theorist**[2]

[1]Unaffiliated, USA

[2]Unaffiliate, Marseille, France

**Correspondence**
Dzhalaev Stasievich Govard, Unaffiliated, USA
Email: jhowar39@emich.edu

**Index Terms**
*Contour, Moore-Neighbor Tracing, Signal Processing, Leftmost Unique-Viable-Neighbor, SLIC, Moore Window, Spur/Tail, Back-Tracking, Isthmus, Segmentation, Super-Pixel.*

**Abstract**
Natural vision systems have evolved sophisticated mechanisms to optimize survival in dynamic environments, far surpassing the limitations of single-aperture systems. Stereopsis provides a computationally efficient means of extracting precise depth from retinal disparity.
Contour tracing is a fundamental operation in image processing for extracting boundaries of objects in labeled images. This paper presents a simple, deterministic algorithm based on local neighborhood analysis that efficiently traces contours in a clockwise manner while ensuring connectivity and avoiding backtracking. The proposed method permutes a Circular Buffer of Moore neighbors to select a relevant subset that prioritizes left turns, drawing inspiration from classical border-following techniques. We describe the algorithm in detail and relate it to prior work in the field. Experimental considerations and applications in computer vision are discussed.



## 1 Introduction

In digital image processing, contour tracing—also known as boundary following or border tracing—plays a crucial role in tasks such as object segmentation, shape analysis, and feature extraction [1]. These algorithms operate on binary or labeled images to delineate the perimeters of connected components, enabling further analysis like chain code generation or topological structure determination [2].

Traditional approaches, such as those based on the Moore neighborhood, have been widely adopted due to their simplicity and effectiveness in handling 8-connected regions [3, 4]. However, variations in neighbor selection and direction prioritization can lead to differences in traversal order and computational efficiency. This paper introduces a compact variant that restricts neighbor consideration to five candidates per step, ensuring deterministic clockwise traversal while maintaining the label consistency of the traced region.

The remainder of this paper is organized as follows: Section 2 reviews related work on contour tracing algorithms. Section 3 details the proposed method. Section 6 discusses potential extensions and limitations, and Section 7 concludes.

## 2 Related Work

Contour tracing algorithms have evolved since the early days of computer vision. One seminal work is the border-following algorithm proposed by Suzuki and Abe [1], which performs topological structural analysis of binary images by identifying surroundness relations among borders. This method, implemented in libraries like OpenCV and scikit-image, efficiently handles multiple connected components and hierarchical contours.

Pavlidis' algorithm [6] offers another perspective, checking a forward triplet of pixels to decide the next move. More recent advancements include parallel implementations for GPUs and pixel-following optimizations for speed.

### 2.1 Moore-Neighbor Tracing

Lastly, the Moore-Neighbor Tracing Algorithm [5] describes the *Moore-Neighborhood M* as the 8-connected pixels surrounding pixel $\vec{p}_t$ in clock-wise order beginning with $\vec{p}_{t-1}$:

$$M(p) = \{(x, y-1),\ (x+1, y-1),\ (x+1, y),\ (x+1, y+1),\ (x, y+1),\ (x-1, y+1),\ (x-1, y),\ (x-1, y-1)\} \tag{1}$$

Variants of this method emphasize avoiding backtracking by adjusting the starting search direction based on the entry point. Our approach builds on these foundations by incorporating a `circular-buffer & sliding-window` to reduce the search scope of neighbors.

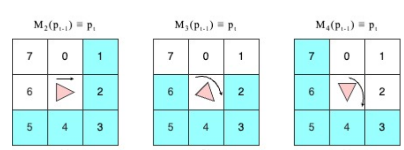


Figure 1: permutations of the Moore Window explicitly stating the index of the pixel $\vec{p}_t$ in $M(\vec{p}_{t-1})$; (**a**). $\sigma(2) = 1$, (**b**). $\sigma(3) = 2$, (**c**). $\sigma(4) = 3$

# 3 Proposed Method

**Algorithm 1** LUVN-Tracing

**Input:** $k, \vec{p}_0$ ▷ $t = 0$
**Output:** $E$
  **while** $\vec{p}_{t>0} \neq \vec{p}_0$ **do**
    $u_t \leftarrow pop(U)$ ▷ $u_0 = 0$
    $m_t \leftarrow m(u_t)$ ▷ $m_0 = eq.[3]$
    $v_{\min} \leftarrow v_{\min}(m_t)$ ▷ Identify the minimal index
    $push(U, v_{\min})$ ▷ Note: $|U| = 1$
    $\vec{p}_{t+1} = M(\vec{p}_t)[v_{\min}]$ ▷ Advance pointer
    $E \leftarrow \vec{p}_{t+1}$ ▷ Append $\vec{p_{t+1}}$ to $E$
    $t \leftarrow t + 1$ ▷ Increment timestep
  **end while**

## 3.1 Moore Window

Consider border pixel $\vec{p}_t$ at timestep $t$ in the approach described by *Toussaint (MNT)* [5]. The moore-neighborhood $M(\vec{p}_t) \mapsto I$, where our window $m_t \subset I \wedge M(\vec{p}_t)[m_{t,0}] \equiv \vec{p}_{t-1}$. An indexing function $\sigma$ is employed utilizing Bitwise operators to transpose $M(\vec{p}_t)$ into a circular buffer:

$$\sigma(u) = (i \mod 8) \mid \sigma(u) \in I \tag{2}$$

and our window, where the index used to advance to the current pixel $u_t$ is the first element of the sequence $m_t$:

$$m_t = \{\sigma(u_t),\ \ldots,\ \sigma(u_t + 7)\} \tag{3}$$

Key insights on the behavior of the original approach[5] were discovered by analyzing adjacent *Moore-Window(s)*: $m_{n \to n+1}$ which lead to the optimizations described in this paper. Atleast two redundancies were found to exist between any adjacent *Moore-Window* pair, with an upper limit of four. The clock-wise logic of this approach[5] halves this inefficiency, but we've further reduced the search scope by limiting the window size to $|m| = 5$, and prioritizing the *Leftmost Unique Viable Neighbor (LUVN)* [3.2] as shown in Figure [1].

## 3.2 Unique Viable Neighbors

To begin we'll define a few term(s) introduced in the previous section, consider our *label-matrix* $\ell_{0 \to n}^{H \times W}$ & *local-label* $k$, where $f(\vec{p}) \equiv k \wedge k \in \ell$. A *viable-neighbor* $f(M_v(\vec{p}_t)) \equiv k$ is considered a *unique* if it was not apart of the *Moore-Window* $(v \in m_t \mid v \notin m_{t-1})$ at the previous timestep $t$. The identification of *UVN(s)* provides insight into the transient dynamics of the current edge pixel $\vec{p}_t$, through which we've coined the term *Leftmost Unique Viable-Neighbor (LUVN)* as the index to the left of $u_t$.

To prioritize The *LUVN* we've modified $eq.$[2]:

$$\sigma(u) = ((u \oplus 4) + 3) \wedge 7 \tag{4}$$

and The *Moore-Window*[3]:

$$m_t = \{\sigma(u_t),\ \ldots,\ \sigma(u_t + 4)\} \tag{5}$$

such that we query at most five *neighbors* as described above, with the exception of the initial timestep $t = 0$, where the entire *Moore-Neighborhood* is searched beginning with $u_0 = 0$. We believe futher discoveries can be founded upon the history of $u_t \in U$, but it's important to note that storing the full history vector for *n-super-pixel(s)* rapidly increases the computational complexity and memory footprint. We maintaining a fixed-size buffer $|U| = 1$ to manage complexity in this paper.

## 3.3 LUVN-Tracing

We begin tracing iteratively from an initial boundary pixel $\vec{p}_0$. At each timestep $t$, given *local-label* $k$ & *Moore-Window*[5] $m_t$ find the first *unique viable-neighbor* 3.2 $v_{\min}$:

$$v_{\min} = \min\{v \in m_t \mid f(M(\vec{p}_t)[v]) \equiv k\} \tag{6}$$

The process continues until returning to $\vec{p}_0$, yielding a closed contour. The selection rule prioritizes the *LUVN* in clockwise ordering, resulting in a clock-wise outer boundary trace similar to *Moore-based* methods.

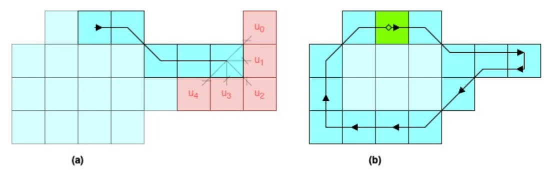


Figure 2: Tracing irregularly shaped *super-pixel*(s). (**a**) When encountering a spur/tail, Algorithm 1 will never reach $\vec{p}_0$ and loop indefinitely. (**b**) Algorithm 2 will reverse direction if a valid label is not found, and by prioritizing the *LUVN* will eventually reach $\vec{p}_0$.

# 4 Limitations

This method is extremely efficient for regular shaped *super-pixel(s)*, but falls short on high-fidelity stereo pairs in real-world applications. *Moore-based* methods tend to avoid backtracking, and are unable to handle *SLIC* algorithms that don't enforce super-pixel smoothness or regularity.

## 4.1 Back-Tracing

Due to the variability of *super-pixel* shapes, or lack of regularity. Our initial approch in *Algorithm* 1 is unable to handle narrow pertrusions in which there is a single entry-point *see Figure [2a]* on page 2. To address this limitation we've added a fallback to our initial Tracing-Algorithm, that reverses direction when reaching the end of a spur/tail illustrated in Figure [2b] page 2. To maintain a unique set $\exists! \vec{p} \in E$ under these conditions we've also introduced a new state matrix $\mathbf{S}_{\leq 3} \in \mathbb{R}_{\geq 0}^{H \times W}$ to track how many times an edge has been visited. While this does increase space complexity, it a very common component of classical search algorithms.

**Algorithm 2** Dynamic LUVN-Tracing

**while** $\vec{p}_{t>0} \neq \vec{p}_0$ **do**
  $u_t \leftarrow pop(U)$ ▷ $u_0 = 0$
  $m_t \leftarrow m(u_t)$ ▷ $m_0 = eq.[3]$
  **for** $v \in m_t$ **do** ▷ find $v_{\min}$
    **if** $f(M(\vec{p}_t)[v]) \equiv \ell$ **then**
      $v_{\min} \leftarrow v$ ▷ Identify the minimal index
      $\vec{p}_{t+1} = M(\vec{p}_t)[v_{\min}]$ ▷ Advance pointer
      $push(U, v_{\min})$ ▷ Push $v_{\min}$, $|U| = 1$
      $E \leftarrow \vec{p}_{t+1}$ ▷ Append $\vec{p_{t+1}}$ to $E$
      $S_{\vec{p_{t+1}}} \leftarrow S_{\vec{p_{t+1}}} + 1$ ▷ Increment visited $S_{\vec{p_{t+1}}}$
      $t \leftarrow t + 1$ ▷ Increment timestep
      **continue while**
    **end if**
  **end for** ▷ $\nexists v_{\min}$ Figure [2b]
  $u_t^{-1} \leftarrow (m_{t,4} + 9) \mod 8$ ▷ Invert entry direction
  $\vec{p}_{t+1} = \vec{p}_{t-1}$ ▷ Backtrack pointer
  $push(U, u_t^{-1})$
  $S_{\vec{p_{t-1}}} \leftarrow S_{\vec{p_{t-1}}} + 1$ ▷ Increment visited $S_{\vec{p_{t-1}}}$
  $t \leftarrow t + 1$ ▷ Step t
**end while**

### 4.2 Non Compact Super-Pixels

*Super-Pixel(s)* with high perimeter-to-area ratios, often lack the structure for closure. These *non-compact clusters* have elongated forms that deviate significantly from a balanced shape(s) *i.e (circle, square, or hexagon)*. This is implied in *Back-Tracing* [4.1] Algorithm [2], but worth mentioning if not explicity written as a condition. Neglecting to verify that $\vec{p}_{t-1} \neq \vec{p}_0$ could cause the pointer to over-shoot the *start* $\vec{p}_0$ resulting in an infinite loop.

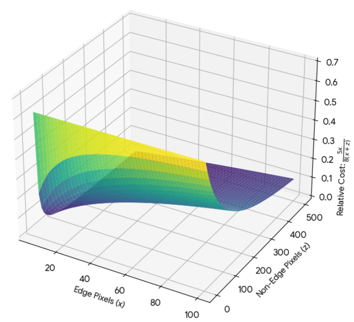


Figure 3: The relative cost of *LUVN-Tracing* measured on a single *super-pixel*. *LUVN-T$_{cost}$* is at worst 62.5% the cost of convolving $\ell^{H \times W}$. This indicates that very large *super-pixel(s)* are traced in a fraction of the time.

### 4.3 Comparatively

While backtracking may be costly, and is generally avoided by most *Moore-based* methods. It is important to note that systematically introducing backtracking to handle variability, still results in a complexity that is dwarfed by methods that convolve The *Moore-Neighborhood*[1] across the entire *label-martix* $M(\vec{p}) \in \ell$, $O(8 \times H \times W)$.

For comparison, the number of edges $E$ is approximately proportional to the number of *super-pixel(s)* $V$. While the total number of pixels $H \times W$ is orders of magnitude larger than $V$ for any practical **region_size** parameter, thus proving the $5E < (8 \times H \times W)$ See Figure [3].

## 5 Topological Anomalies

In *SLIC* algorithms over-segmentation often occurs, where the shape constraints may be too loose, resulting in case(s) of *super-pixel* irregularity like that of Figure [2] solved by *Back-Tracing* [4.1]. Through which Algorithm [2] is capable of traversing *spur/tail(s)*, or any degree of topological variability for that matter. Unfortunately, narrow pertrusions such as the aforementioned often lead into larger regions of *like-labels* referred to as *Isthmus(s) (see Figure [4a])* in topological analysis. While traversal of any degree of topological variability is guaranteed by Algorithm[2], entry/traversal of an *Isthmus* does not guarantee an exit.

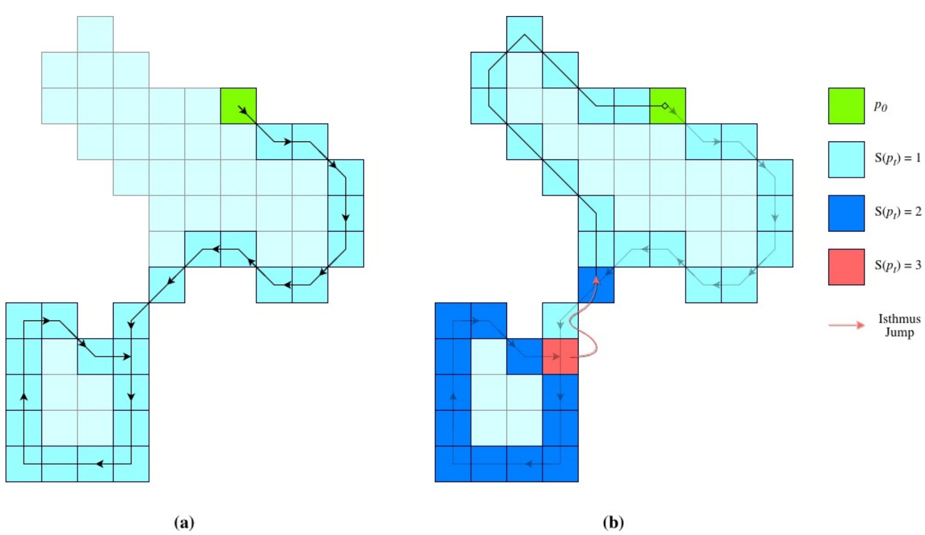


Figure 4: *Isthmus-jump* illustration
(**a**) Initial traversal of *Isthmus*, where $\{s \in S \mid s \leq 1\}$. (**b**) Complete trace path, where *Back-trace(s)* are highlighted in royal-blue $S_{\vec{p}_t} \equiv 2$, as well as the *Isthmus-jump* node/path in red $S_{\vec{p}_t} > 2 \rightarrow S_{\vec{p}} \equiv 1$.

**Algorithm 3** Dynamic LUVN-Tracing w/ Isthmus-jump

**while** $\vec{p}_{t>0} \neq \vec{p}_0$ **do**
  $u_t \leftarrow pop(U)$ ▷ $u_0 = 0$
  $m_t \leftarrow m(u_t)$ ▷ $m_0 = eq.[3]$
  **for** $v \in m_t$ **do** ▷ find $v_{\min}$
    **if** $f(M(\vec{p}_t)[v]) \equiv \ell$ **then**
      $v_{\min} \leftarrow v$ ▷ Identify the minimal index
      $\vec{p}_{t+1} = M(\vec{p}_t)[v_{\min}]$ ▷ Advance pointer
      $push(U, v_{\min})$ ▷ Push $v_{\min}$, $|U| = 1$
      $E \leftarrow \vec{p}_{t+1}$ ▷ Append $\vec{p_{t+1}}$ to $E$
      $S_{\vec{p_{t+1}}} \leftarrow S_{\vec{p_{t+1}}} + 1$ ▷ Increment visited $S_{\vec{p_{t+1}}}$
      **if** $S_{\vec{p_{t+1}}} > 2$ **then** ▷ If *local-minima*
        $\vec{p_{t+1}} \leftarrow s_{\min}$ ▷ Exit *Isthmus eq.[8]*
      **end if**
      $t \leftarrow t + 1$ ▷ Increment timestep
      **continue while**
    **end if**
  **end for** ▷ $\nexists v_{\min}$ Figure [2b]
  $u_t^{-1} \leftarrow (m_{t,4} + 9) \mod 8$ ▷ Invert entry direction
  $\vec{p}_{t+1} = \vec{p}_{t-1}$ ▷ Backtrack pointer
  $push(U, u_t^{-1})$
  $S_{\vec{p_{t-1}}} \leftarrow S_{\vec{p_{t-1}}} + 1$ ▷ Increment visited $S_{\vec{p_{t-1}}}$
  **if** $S_{\vec{p_{t+1}}} > 2$ **then** ▷ If *local-minima*
    $\vec{p_{t+1}} \leftarrow s_{\min}$ ▷ Exit *Isthmus eq.[8]*
  **end if**
  $t \leftarrow t + 1$ ▷ Step t
**end while**

### 5.1 Isthmus-jump

In section [4.1] we introduced *Back-Tracing* coupled with the *visited-matrix* **S** of *unsigned-int(s)*. It's important to note that an *Isthmus* is always preceeded by a *spur/tail*.
We can assert that:

$$\exists \vec{e} \in E \mid S_{\vec{e}} > 1 : \exists(\mathit{Spur} \vee \mathit{Isthmus})$$

but in order to escape a *local-minima* without breaking Algorithm[2] we must derive another rule to distinguish between the two. Fortunately **S** is already equipped for tracking multiple visits, and it is reasonable to conclude that a second *Back-Trace* on any node is computationally inefficient.
Thus, where:

$$S_{\vec{p_{t+1}}} > 2$$

we're at the entrance of a *local-minima* and must search for an exit $\vec{s}_{\min}$, where $S_{\vec{p}} \equiv 1$, given the *sequence* $m'$:

$$m' = \left\{\sigma(u_t),\ \ldots,\ \sigma(u_t + 5)\right\},\ |\ m'\ | = 6 \tag{7}$$

The *exit (Isthmus–jump)* $s_{\min}$:

$$s_{\min} = \left\{M_{\min\{v \in m'\}}(\vec{p}_{t+1}) \mid S_{M_v(\vec{p_{t+1}})} \equiv 1\right\} \tag{8}$$

will place our pointer just outside the *local-minima* in a direction of continuance *(see Figure [4b])*.

## 6 Discussion

The proposed algorithm's efficiency stems from its local, constant-time decisions per pixel, making it suitable for large images. Compared to Suzuki's method [1], it simplifies hierarchy detection but focuses on single-region tracing. Future work could extend it to GPU parallelism or integrate with deep learning-based segmentation.
Limitations include sensitivity to noise in labels, which may require preprocessing.

## 7 Conclusion

We have presented a concise deterministic contour tracing algorithm that leverages restricted neighbor subsets and minimal-index selection for efficient boundary following. By building on established techniques, it offers a straightforward implementation for modern image processing pipelines.

## References

[1] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[2] H. Freeman, "Boundary encoding and processing," in *Picture Processing and Psychopictorics*, 1970.

[3] M. J. E. Moore, "Machine perception of three-dimensional solids," in *Optical and Electro-Optical Information Processing*, MIT Press, 1965.

[4] A. G. Ghuneim, "Contour tracing tutorial," 2004. [Online]. Available: https://www.imageprocessingplace.com

[5] G. T. Toussaint, "Chapter 2: Grids, connectivity and contour tracing," McGill University, 1980s.

[6] T. Pavlidis, *Algorithms for Graphics and Image Processing*. Computer Science Press, 1982.