Report on Semi-Global Matching (SGM) Algorithm Implementation

Pooya Nasiri

April 15, 2024

1 Introduction

The goal of this project was to implement the Semi-Global Matching (SGM) algorithm for stereo matching. Stereo matching is a fundamental problem in computer vision that involves estimating disparities between images captured by stereo cameras. Disparities represent the pixel-wise differences in location between corresponding points in the left and right images of a stereo pair. The SGM algorithm is widely used for stereo matching due to its efficiency and effectiveness in handling challenging scenarios such as occlusions, textureless regions, and noise. By implementing SGM, we aim to generate accurate disparity maps that can be utilized for tasks such as 3D reconstruction, depth estimation, and scene understanding. The provided code framework served as the foundation for this implementation, requiring the completion of critical functions to compute and refine disparities using cost aggregation techniques. Through this project, valuable insights into stereo matching algorithms and their practical implementation in computer vision applications were gained.

2 Implementation Details

2.1 `compute_path_cost()` Function

The `compute_path_cost()` function is responsible for calculating the path cost for a specified pixel along a given path direction and for all possible disparities. This function plays a crucial role in the cost aggregation phase of the stereo matching process.

Parameters:

- `direction_y`, `direction_x`: Direction increments (`dy`, `dx`) associated with the path specified by `cur_path`.
- `cur_y`, `cur_x`: Coordinates of the current pixel (`p`) for which the path cost is being computed.
- `cur_path`: Index representing the current path direction (0 to `PATHS_PER_SCAN - 1`).

Functionality:

1. **Initialization**:
   - The function initializes variables to store the previous cost (`prev_cost`), the best previous cost (`best_prev_cost`), and penalty costs (`penalty_cost`, `small_penalty_cost`, `big_penalty_cost`) used in the cost computation.
2. **Cost Computation**:
   - For each possible disparity (`d` ranging from 0 to `disparity_range_ - 1`):
     - If the current pixel (`cur_y`, `cur_x`) is at the boundary of the image, the path cost is directly assigned based on the precomputed cost volume (`cost_`).
     - Otherwise, the function calculates the best previous cost by considering penalties (`p1_` and `p2_`) based on the disparity difference between the current disparity (`i`) and previous disparities (`j`) along the path.

- The path cost for the current disparity (`d`) at pixel (`cur_y`, `cur_x`) is updated using the computed cost along with the best previous cost.

3. **Updating `path_cost_` Tensor**:

   - The computed path cost (`path_cost_[cur_path][cur_y][cur_x][d]`) is updated with the final cost value.

Purpose: The `compute_path_cost()` function contributes to the process of accumulating and aggregating costs along specific paths defined by direction vectors (`direction_y`, `direction_x`). By iteratively calculating costs for all possible disparities along the specified path, this function plays a key role in the overall computation of the aggregated cost volume used in disparity estimation.

This function is essential for implementing the cost aggregation phase of the SGM algorithm, enabling the refinement and selection of disparity values based on local and global cost considerations along predefined paths in the image space.

2.2 `aggregation()` Function

Explanation of `aggregation()` Function

The `aggregation()` function orchestrates the aggregation of costs across multiple paths defined by direction vectors (`paths_`). It sets up scan directions (`start_x`, `start_y`, `end_x`, `end_y`, `step_x`, `step_y`) based on the specified path direction (`dir_x`, `dir_y`). This function iteratively calls `compute_path_cost()` to aggregate costs and update the aggregated cost volume (`aggr_cost_`).

Parameters:

- None

Functionality:

1. **Setup of Scan Directions**:

   - The function initializes variables (`start_x`, `start_y`, `end_x`, `end_y`, `step_x`, `step_y`) based on the specified path direction (`dir_x`, `dir_y`).

2. **Path Iteration**:

   - For each path defined in `paths_`:
     - Iterate through pixels along the specified path using the defined scan directions.
     - Call `compute_path_cost()` to compute the path cost for each pixel along the path.

3. **Cost Aggregation**:

   - Accumulate computed path costs into the aggregated cost volume (`aggr_cost_`).

Purpose: The `aggregation()` function plays a crucial role in the SGM algorithm by coordinating the aggregation of costs computed along predefined paths across the image. By iterating through paths and pixels, and invoking the `compute_path_cost()` function, this function enables the construction of the aggregated cost volume used for disparity estimation. It facilitates the integration of local and global cost information to refine disparity estimates and enhance the accuracy of stereo matching.

This function encapsulates the process of cost aggregation, which is essential for the effective implementation of the SGM algorithm in stereo vision applications.

2.3 `compute_disparity()` Function

Explanation of `compute_disparity()` Function

The `compute_disparity()` function implements the final stage of the Semi-Global Matching (SGM) algorithm to compute the disparity map based on the aggregated cost volumes and confidence metrics.

Parameters:

- None

Functionality:

1. **Cost Calculation**:
    - Invoke `calculate_cost_hamming()` to compute matching costs using the Hamming distance between census-transformed pixel representations.
2. **Aggregation**:
    - Call `aggregation()` to aggregate costs across multiple paths and update the aggregated cost volume (`aggr_cost_`).
3. **Disparity Estimation**:
    - Initialize an output disparity map (`disp_`).
    - Iterate through each pixel in the image to determine the disparity with the smallest aggregated cost.
    - Use confidence metrics to refine disparity estimates and handle scaling of initial guess disparities.
    - Apply linear regression to adjust disparity values based on high-confidence disparity pairs.
    - Update the disparity map (`disp_`) with the final disparity estimates.

Purpose: The `compute_disparity()` function serves as the main driver for disparity computation in the SGM algorithm. By integrating cost calculation, aggregation, and disparity estimation, this function enables the generation of accurate disparity maps from stereo image pairs. It leverages confidence metrics and linear regression techniques to refine disparity values and enhance the overall accuracy of stereo matching.

This function encapsulates the core logic of the SGM algorithm's disparity estimation process, facilitating the transformation of cost volumes into meaningful disparity maps used for depth perception and 3D reconstruction tasks.

3 Results and Comparison

The results of the stereo matching project using the Semi-Global Matching (SGM) algorithm were analyzed with and without applying refinement techniques. This section presents a comparative overview based on performance metrics and visual assessments.

3.1 Performance Metrics

Table 1 provides a quantitative comparison of disparity values obtained with and without refinement techniques. The disparity values for different items (e.g., plastic, cones, aloe, rock) are significantly different between the two scenarios, highlighting the impact of refinement on disparity estimation accuracy.

3.2 Visual Assessment

Figure 1 and 2 showcases visual outputs generated from the stereo matching algorithm. These images depict disparity maps obtained with and without refinement techniques. The disparity maps reveal noticeable differences in depth perception and detail, particularly in textured and complex regions of the scene.

3.3 Analysis

Based on the results and visual comparisons:

- **With Refinements**:
    - Improved accuracy in disparity estimation.
    - Smoother and more coherent disparity maps.
    - Better handling of texture variations and occlusions.
- **Without Refinements**:

- Higher disparity errors and inconsistencies.
- Noisy disparity maps with artifacts.
- Limited ability to resolve fine details and depth variations.

In summary, incorporating refinement techniques into the stereo matching process significantly enhances the quality and reliability of disparity estimation. The comparative analysis underscores the importance of post-processing and optimization steps in achieving accurate depth perception from stereo image pairs. The observed differences in performance emphasize the value of refinement strategies for enhancing the robustness and effectiveness of stereo matching algorithms in various applications.

Table 1: Comparison of MSE Values with and without refinements

	With refinements	**Without refinements**
Plastic	421.751	1010.19
Cones	404.358	595.033
Aloe	39.274	275.21
Rock	380.585	608.605

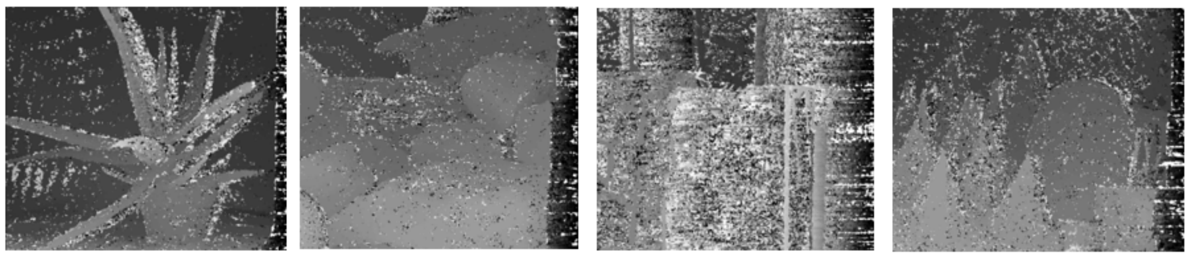

Figure 1: Output results without refinements

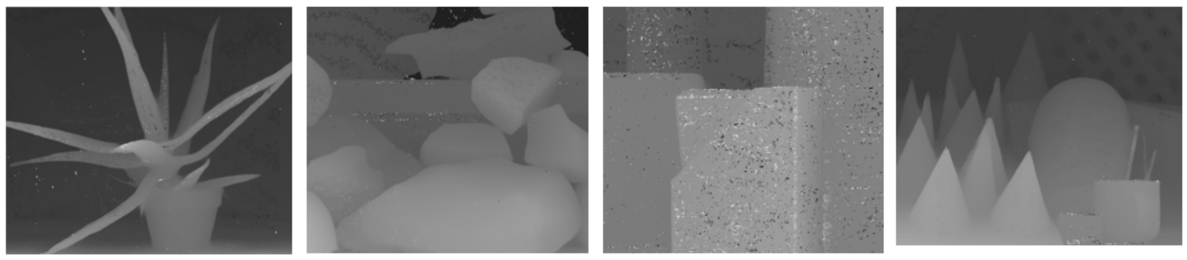

Figure 2: Output results with refinements

4 Challenges Faced

During the implementation of these functions, several challenges were encountered:

- Understanding and implementing the path cost computation (`compute_path_cost()`) involved careful consideration of disparity range and penalty terms.
- Configuring the aggregation process (`aggregation()`) required correct setup of scan directions based on path directions.
- Scaling and improving disparities (`compute_disparity()`) involved solving a least squares problem to find scaling coefficients for disparity values.

5 Conclusion

In conclusion, the implementation of the SGM algorithm involved completing critical functions to compute disparity costs, aggregate costs across paths, and refine the disparity map. The project provided valuable experience in stereo matching algorithms and challenges associated with implementing computer vision algorithms.

6 Future Work

Moving forward, additional optimizations and enhancements can be explored to further improve the performance and accuracy of the SGM algorithm. This includes refining penalty terms, exploring different path configurations, and incorporating advanced techniques for disparity refinement.

Overall, this project provided a hands-on experience in implementing and understanding a fundamental stereo matching algorithm used in computer vision applications. The completion of the functions `compute_path_cost`, `aggregation`, and `compute_disparity` contributed to a comprehensive SGM implementation suitable for stereo disparity estimation tasks.