Loopy Belief Propagation
TRIVA course, ENPC

Dhia Garbaya

Mai 2024

1 Question 1

- **i. Formula for Computing the Vector of Messages:**
In the Loopy Belief Propagation (LBP) algorithm, messages are passed between nodes in an iterative manner.
At each iteration t, every node p sends a message to its neighboring node q.
The message from node p to node q at iteration t can be computed using the following formula:

$$m_{p \rightarrow q}^{t}\left(l_{q}\right)=\min _{l_{p}}\left[D_{p}\left(l_{p}\right)+V\left(l_{p}, l_{q}\right)+\sum_{r \in N(p) \backslash q} m_{r \rightarrow p}^{t-1}\left(l_{p}\right)\right]$$

Where: $D_{p}\left(l_{p}\right)$ is the data cost of assigning label l_p to pixel p. $V\left(l_{p}, l_{q}\right)$ is the pairwise cost of assigning labels l_p and l_q to neighboring pixels p and q, respectively. And $N(p)$ is the set of neighbors of pixel p.

- **ii. Complexity of Computing the Vector:**
The complexity of computing the message vector from node p to node q is primarily determined by the cost of evaluating the minimum operation over all possible label assignments l_p.
Given that there are $d_{\max}$ possible disparities:
1. Data Cost Term $D_{p}\left(l_{p}\right)$: Evaluating this term is $O(1)$.
2. Pairwise Cost Term $V\left(l_{p}, l_{q}\right)$: Evaluating this term is $O(1)$.
3. Sum of Incoming Messages: There are up to 3 neighbors for each pixel (excluding q), and for each label l_p, we sum messages. This requires $3 \times d_{\max}$ operations.
4. Minimization Over l_p : This requires $d_{\max}$ evaluations.

Therefore, the total complexity for computing the message $m_{p \rightarrow q}^{t}\left(l_{q}\right)$ is $O\left(d_{\max}^{2}\right)$ for each message.

- **iii. Formula for Obtaining Beliefs and MAP Labels:**
The belief at node q for label l_q at iteration t is obtained by summing the incoming messages from all its neighbors and adding the data cost:

$$b_{q}^{t}\left(l_{q}\right)=D_{q}\left(l_{q}\right)+\sum_{p \in N(q)} m_{p \rightarrow q}^{t}\left(l_{q}\right)$$

To obtain the MAP labels, we assign to each node q the label l_q that minimizes its belief:

$$l_{q}^{MAP}=\arg \min _{l_{q}} b_{q}^{t}\left(l_{q}\right)$$

2 Question 2

With the Potts model, the pairwise potential $V\left(l_p-l_q\right)$ can be either 0 or 1 .
This allows us to optimize the computation by splitting it based on whether l_p equals l_q or not:

- If $l_p=l_q, V\left(l_p-l_q\right)=0$.
- If $l_p \neq l_q, V\left(l_p-l_q\right)=1$.

Given this, the message update formula becomes (see also P.S (*)):
$m_{p \rightarrow q}^t(l_q)=\min \left(\min_{l_p=l_q}\left(D_p(l_p)+\sum_{s \in N(p) \backslash\{q\}} m_{s \rightarrow p}^{t-1}(l_p)\right), \min_{l_p \neq l_q}\left(D_p(l_p)+\sum_{s \in N(p) \backslash\{q\}} m_{s \rightarrow p}^{t-1}(l_p)+\lambda\right)\right)$.

Simplification:
Given that the second term always includes the constant 1xλ , we can simplify the computation as follows:
- Define $f_p^t\left(l_p\right)=D_p\left(l_p\right)+\sum_{s \in N(p) \backslash\{q\}} m_{s \rightarrow p}^{t-1}\left(l_p\right)$ for all l_p.
- Then, $m_{p \rightarrow q}^t\left(l_q\right)=\min \left(f_p^t\left(l_q\right), \lambda+\min_{l_p} f_p^t\left(l_p\right)\right)$.

Computational Complexity:
This efficient approach reduces the need to compute the pairwise potential for every label combination. Instead, we:
- Compute $f_p^t\left(l_p\right)$ for all l_p in $O(d)$.
- Then find the minimum of $f_p^t\left(l_p\right)$ across all l_p also in $O(d)$.
- Finally, for each l_q, compute $m_{p \rightarrow q}^t\left(l_q\right)$ in constant time $O(1)$ using precomputed f_p^t.
$\rightarrow$ Therefore, the overall complexity per message is $O(d)$ (linear) after this Potts model simplification, more efficient than $O\left(d^2\right)$ in the general case.
This improvement is especially beneficial for images with a large range of disparities.

3 Question 3

Remark: Let's clarify the meaning of messages in this code:

- **MsgUPrev**(y,x): message **sent** from (y,x) to the pixel above it (Up) $\uparrow$.
- **MsgRPrev**(y,x): message **sent** from (y,x) to the pixel to its Right $\rightarrow$.

So to get for each pixel (y,x) his **received messages**, we should encode that:

- **msg_from_U_neighbour**: MsgDPrev of pixel (y-1,x) $\downarrow$.
- **msg_from_R_neighbour**: MsgLPrev of pixel (y,x+1) $\leftarrow$.

Thus the `np.roll` calls in the function `update_msg`.

4 Question 4

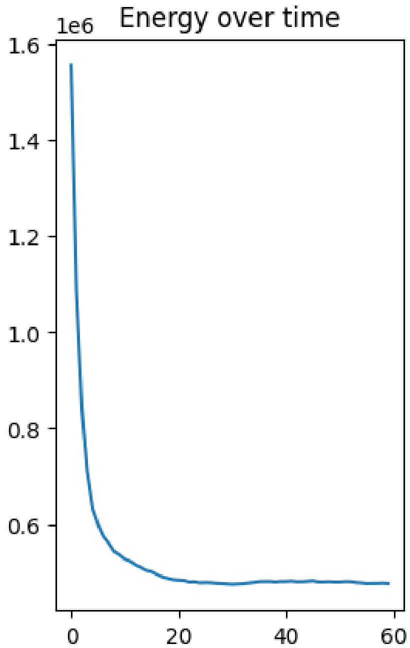
The `normalize_msg` function in the Loopy Belief Propagation algorithm is critical for:

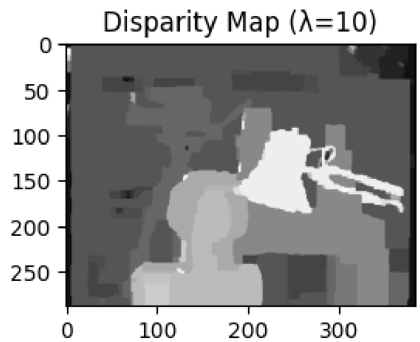
1. **Maintaining Numerical Stability:** It prevents numerical overflows or underflows by ensuring that message values do not grow too large or too small.
2. **Improving Convergence:** Normalization helps the algorithm converge more smoothly and uniformly by preventing any single message from becoming dominant (cf. [1] for details).
3. **Reducing Computational Bias:** It reduces bias towards certain disparity values by making adjustments that center the messages around zero.

Moreover, it ensures a **preservation of Minima:** Despite normalization, the relative ordering of beliefs for each pixel's disparity remains unchanged. This means that the disparity that minimizes the belief before normalization will continue to do so afterward.

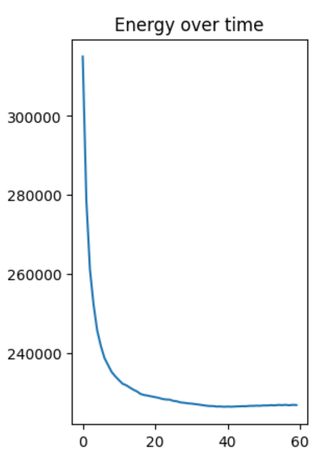
5 Question 5

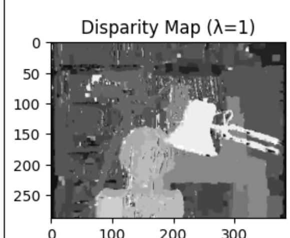
Here is the result for $\lambda = 10$.

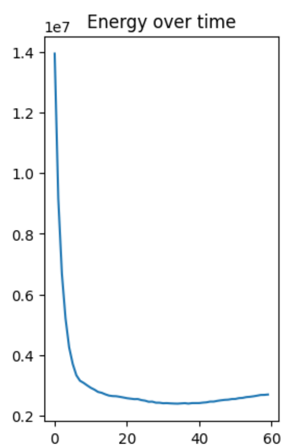


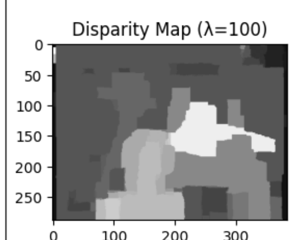


And here are the results for respectively $\lambda = 1.$ and $\lambda = 100$.









Observations on the Effect of λ:

- **Disparity Maps:**
  - $\lambda = 1$: Shows more fine details and sharper edges but contains a lot of noise, indicating less regularization.
  - $\lambda = 10$: Achieves a balance between detail and smoothness, keeping some detail with moderate regularization. I find it the best.
  - $\lambda = 100$: Very smooth maps with uniform regions, suggesting strong regularization and loss of fine details.
- **Energy Plots:**
  - The energy drops sharply in the initial iterations across all λ values, indicating rapid convergence.

  - Higher λ values result in higher starting energy due to stronger initial regularization effects.
  - For λ=100, energy starts to increase in the last 20 iterations.

Defects in the Estimation:

- **Over-smoothing:** High λ values lead to over-smoothed maps, losing critical details, especially around edges and small features.
- **Noise and smoothing:** Low λ values show higher noise levels, potentially causing wrong disparity estimations. And high values lead to over-smoothing (as smoothness cost gets higher), loosing critical details
- **Fast energy convergence:** Rapid convergence ($\approx$ 30 iterations) might indicate settling at a local minimum rather than a global minimum.
- **Edge Artifacts:** Edge computations aren't finely done. They're often ignored (cf. code where I set DataCost to τ

P.S:
In the formulas (and code) derived from the Potts Model, I add λ and not 1 because I consider λ as a part of the pairwise cost. Plus, code showed that there won't be any effect on disparity map if we deal with 1 instead of λ in `update_msg`.

References

[1] Victorin Martin, Jean-Marc Lasgouttes, and Cyril Furtlehner. *The Role of Normalization in the Belief Propagation Algorithm*. 2011. arXiv: 1101.4170 [cs.LG].