Improved Seam Carving for Video Retargeting - Sned Noodles

Aakash Dantre, Roll Number - 2018101039 **Date - 19th Nov, 2020**
Sab apna dallo

Contents

1.Team Members and Roles

Team Member Name	Team Member Role	Number of Hours Devoted (7 hrs/week x 3 weeks)
Aakash Dantre	•	Around 22 hours
Jay Sharma	•	Around 21 hours
Dhurv Arya	•	Around 20 hours
Varun Chhangani		
Vishal Verma		

2.Overview

2.a. About seam carving

Seam: monotonic and connected path of pixels going from the top of the image to the bottom, or from left to right. Satisfying the following constraints:
➢ Monotonicity: the seam must include one and only one pixel in each row (or column for horizontal seams).
➢ Connectivity: the pixels of the seams must be connected

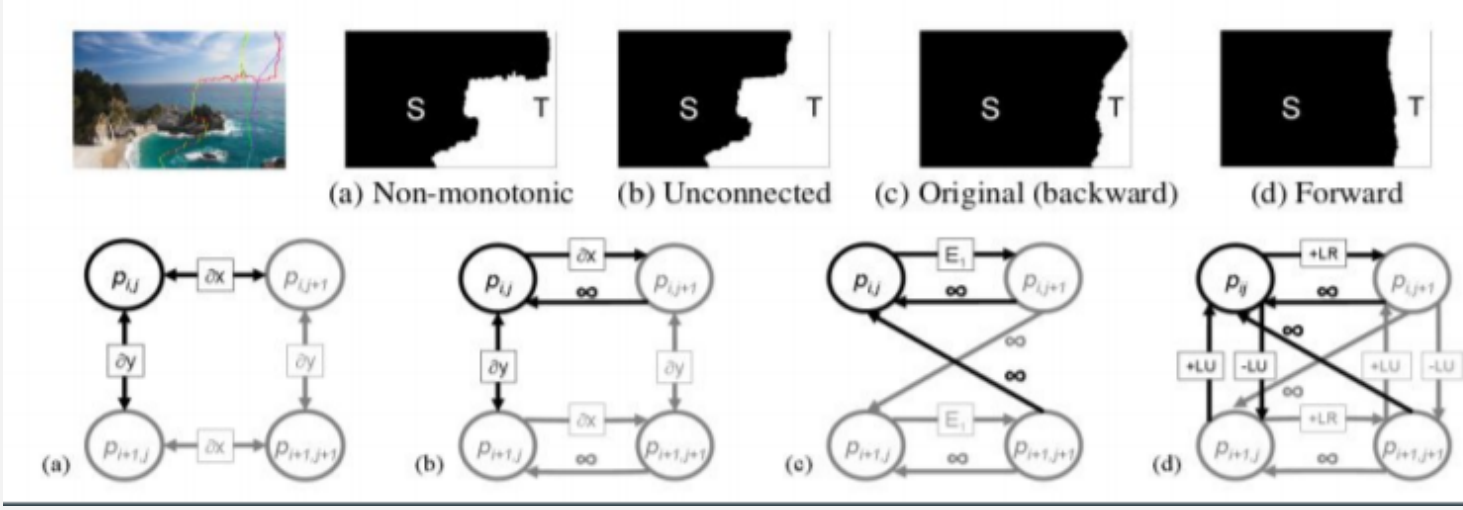
Seam carving is an effective technique for content aware image retargeting. Video, like images, should support content aware resizing. Instead of removing 1D seams from 2D images we remove 2D seam manifolds from 3D space-time volumes.

2.b. About Graph cut

To achieve this we use graph cuts that are suitable for 3D volumes.

We formulate the seam carving operator as a minimum cost graph cut problem on images and then extend this formulation to videos.
❖ Each pixel is considered a node and arc (edges in graph terminology) is drawn between neighbouring nodes.

❖ Virtual terminal nodes, S (source) and T (sink) are created and connected with infinite weight arcs to all pixels of the leftmost and rightmost columns of the image respectively.
❖ The optimal seam is defined by the minimum cut which is the cut that has the minimum cost among all valid cuts.



2.c. Graph cut for videos

➔ Consider the X × T planes in the video cube and use the same graph construction as in X × Y including backward diagonal infinity arcs for connectivity.
➔ A partitioning of the 3D video volume to source and sink using graph cut will define a manifold inside the 3D domain
➔ The graph cut algorithm runs in polynomial time, but in practice was observed to have linear running time on average [Boykov and Kolmogorov 2004].
➔ The graph cut approach to seam carving allows us to extend the benefits of content-aware resizing to video.

3. Design Constructs

3.1 Narrative

Here, we examine the classes and files used.

Narrative

Remove_seam.py is first run, which opens the video file and array it such that it comes in 3d volume. It further store that in array.txt file and runs a bat file(flow.bat) or bash file(bash flow.sh) on the basis of your OS. These files drive the rest of code. Preproc.cpp is executed which calculates graph edge based energy functions and stores it. The output of it is used by graphcut.exe which is compiled using main.cpp, graph.h, block.h, graph.cpp, maxflow.cpp. All the major seam calculations happens in here, as it gives seamout.txt as the final output. Get_seamtry.py uses seamout.txt and remove all the seams from our 3d volume of video and stores it back in array.txt. Visual_compare.py coverts array.txt to video format.

3.Responsibility(ies) of each major file

Responsibility(ies) of each major file

Classes	Description
Remove_seam.py	- Driver, first file to be run - Coverts video, resize it according to config.py - Calls flow.bat or bash_flow.sh - Remove NUM_SEAMS number of seams from video.
Flow.bat or bash_flow.sh	- Drives the rest of code - Removes one seam from video - Compiles preproc.cpp to create preproc.exe - Compilesmain.cpp, graph.h, block.h, graph.cpp, maxflow.cpp and create executable graphcut.exe.
preproc.cpp	- Uses energy function - Creates edge based graph - Output of this file is passed on to graphcut.exe
block.h	- Template classes Block and DBlock - Implement adding and deleting items of the same type in blocks. - If there there are many items then using Block or DBlock is more efficient than using 'new' and 'delete' both in terms of memory and time
graph.h	- **UI** for Lane

	- Event Listener for **viewing game scores, pausing game and pinsetter**
graph.cpp	- Creates an array of lane(call **Lane()**) - iterate through the available lanes and **assign the parties** in the wait queue if lanes are available. - Returns a Vector of party names to be displayed in the GUI representation of the **wait queue**.
main.cpp	- Simulates the game - **Initiate pinsetter** for that game - After game finishes, initiate end game report for that game **EndGamePrompt()** - Initiate ScoreReport if user decide to end game -> **ScoreReport()** - EventListener for pinsetter event - Reset Lane for new games - **resetscore()** - MarkScore of players -> **markscore()** - Calculate the score using all possibility in bowling -> **getscore()**
maxflow.cpp	- **UI** for viewing score of running game of a given lane - On receiving lane events, it calls getscore() method of lane() class to **write scores of a given throw**.
Get_seamtry.py	- seam removed array is created using seamout.txt and stored in array.txt
visual_compare.py	- Coverts array.txt into final seam removed video format
PinsetterView()	- **UI** for pinsetter - Receive pinsetter event and **update the view**
ScoreHistoryFile()	- Have access of SCOREHISTORY.DAT - Add result of every game -> **addscore()** - Get every result of a given players -> **getscores()** -

4. Final Output

4.1 input

4.2 output

5. Installation

5.1 prerequisites

5.2 running

Install code from github

- git clone https://github.com/Digital-Image-Processing-IIITH/project-sned-noodles.git
- cd src
- python seam_remove.py
- python visual_compare.py