Fast Marching Method for Outpainting of Certain Gradient Backgrounds

D. Gueorguiev , 11/7/2024

# Introductory Notes

//TODO: explain the history of the FMM, discuss the inpainting modeling work historically, describe the inpainting models used currently and how the one developed in Telea’s paper will be used in modified form to create outpainting model for slowly varying gradient backgrounds.

For introduction into Fast Marching Method refer to the document in [0].

# Model for Inpainting

The model for inpainting and the related algorithm are taken from Telea’s paper (refer to [1] for details)

Consider the figure below in which one must inpaint the point situated on the boundary of the region to inpaint .

We take a neighborhood with radius of the known image around . The inpainting of should be determined by the values of the known image points close to (i.e. in ). Grayscale images are considered in this model , color images are handled as an obvious extension by the presented model and algorithm.

boundary

region to be

inpainted

known

neighborhood

known image

**a)**

**b)**

When is small enough a first order approximation of the image in point , given the image and the gradient values of point :

(1)

Next, we inpaint point as a function of all points in by summing the estimates of all points , weighted by a normalized weighting function :

(2)

The weighting function is designed in such way that the inpainting of propagates the gray value as well as the sharp details of the image over .

In [1] , is determined based on three components – a *directional* component , *geometric distance* component , and the *level set* distance component .

(3)

where

(4)

(5)

(6)

The *directional component* ensures that the contribution of the pixels close to the normal direction (which is the information propagation direction) is higher than for those farther from .

The *geometric distance* component decreases the contribution of pixels geometrically farther from .

The *level set* distance component ensures that pixels close to the contour through p contribute more than farther pixels.

Notice the presence of the reference distances - in the geometric component and in the level set distance component. In the algorithm, those are set to the interpixel distance which is 1.

Overall, the above factors model the manual inpainting heuristics that describe how to paint a point by strokes bringing color from a small region around it.

For up to about six pixels i.e. when inpainting thin regions, and have a weak effect. For thicker regions to inpaint we could use an of 12 pixels and using and provides better results than using alone. Inpainting is the best visually when all three components have been used.

A black oval with a white square

Description automatically generated with medium confidence

Figure: thick region to inpaint (a) and result (b). Effect of the weighting functions: direction (c), direction and geometric distance (d) , direction and level set distance (f). The figure is taken from [1].

## Adding Fast Marching Method to the Inpainting Process

Eq. (2) gives the rule how to inpaint a point on the unknown’s region boundary as a function of known image pixels only.

To inpaint the whole , we iteratively apply Eq. (2) to all the discrete pixels of , in increasing distance from ’s initial position , and advance the boundary inside until the whole region has been inpainted. Inpainting points in increasing distance order from ensures that areas closest to known image points are filled in first, mimicking manual inpainting techniques.

Implementing the above requires a method that propagates into by advancing the pixels of in order of their distance to the initial boundary . For this we use the Fast-Marching Method (FMM). FMM is an algorithm that solves the Eikonal equation.

## The Eikonal equation

The Eikonal equation is given with

on , with on (7)

The boundary condition applied to the Eikonal equation is an example of a [*Dirichlet boundary condition*](https://en.wikipedia.org/wiki/Boundary_value_problem).

The solution of Eq. (7) is the distance map of the pixels in to the boundary . The level sets (or *isolines*) of are exactly the successive boundaries of the shrinking that need to be inpainted.

We solve the Eikonal equation by finite difference discretization as:

(8)

In (8) we have denoted with , and the following finite difference operators on :

## The FMM algorithm

{

}

Excerpt 1: the FMM algorithm discussed in Telea’s paper

The FMM algorithm guarantees that the pixels of are always processed in increased order of their distance-to-boundary . Thus the algorithm will inpaint the closest pixels to the known image area first. The FMM algorithm explicitly defines and maintains a *narrow band* that separates the known from the unknown image *and* specifies which is the next pixel to inpaint. The narrow band can be implemented as *min heap*. There are five steps in the Telea’s algorithm. The Python code of the algorithm is added to the Appendix of this document.

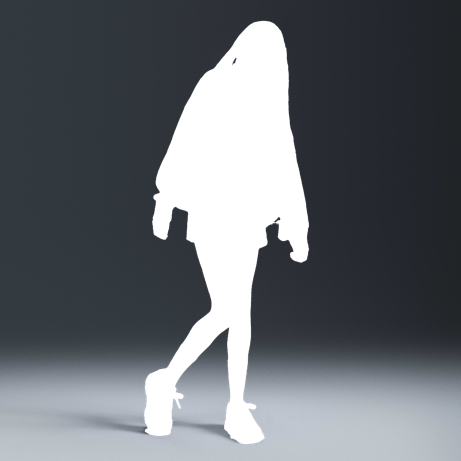
Following the idea from Sethian’s first paper on FMM for image inpainting (refer to [2] for details on *the upwind manipulation*) we solve (8) in a 2x2 grid and retain the solution with the minimal value.

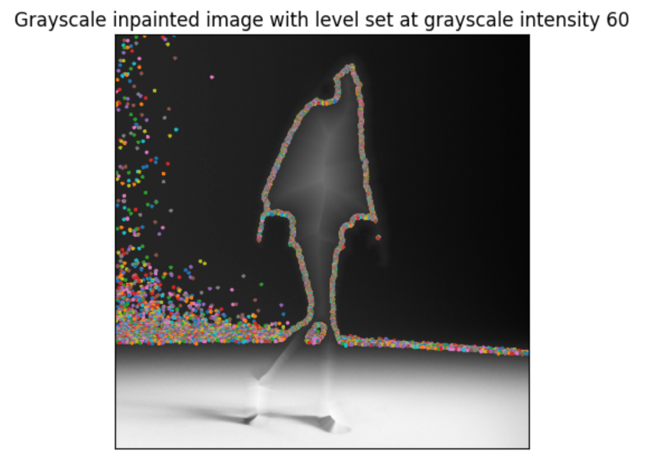
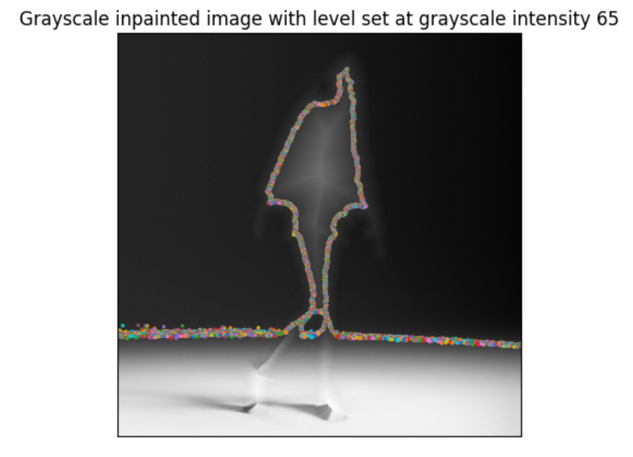
# Applying the Telea’s Inpainting Algorithm to Large Closed Regions on slowly varying gradient background

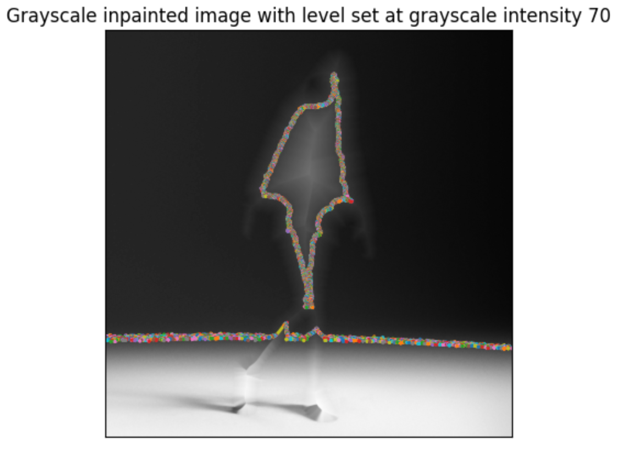
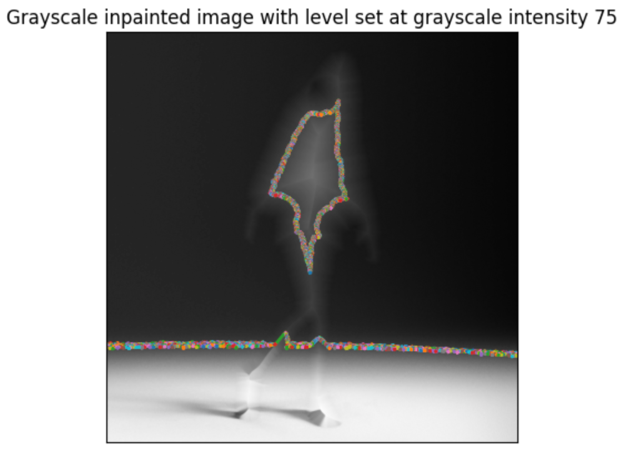
**Problem**: The Telea’s algorithm does not do good job in inpainting the removed object of interest using the slowly varying gradient background as a template.

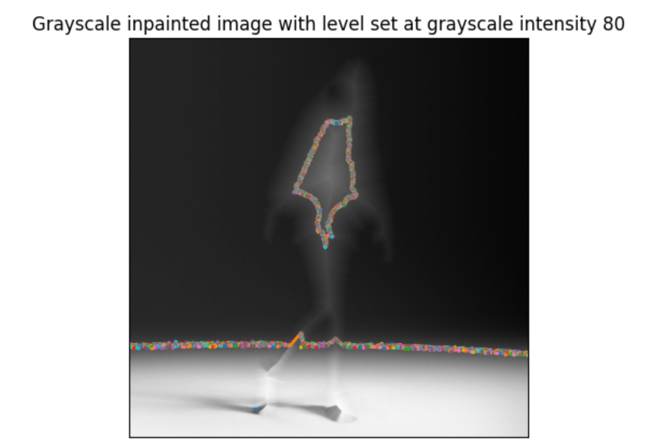
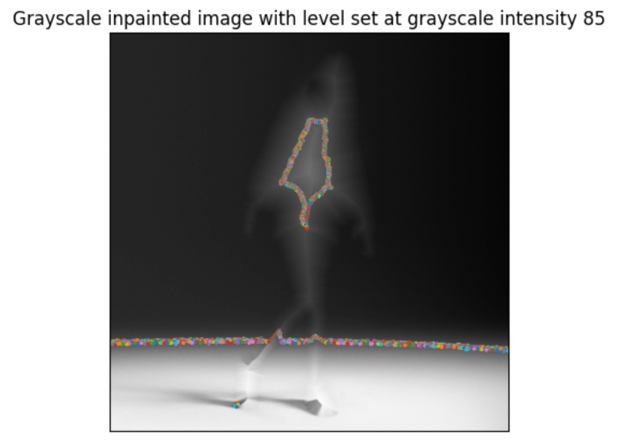
Before we continue with the adaptation of the algorithm for our outpainting task we need to fine tune it so that it handles slowly varying gradient backgrounds in realistic way. That is, we want to propagate the gray content in the inpainted region correctly.

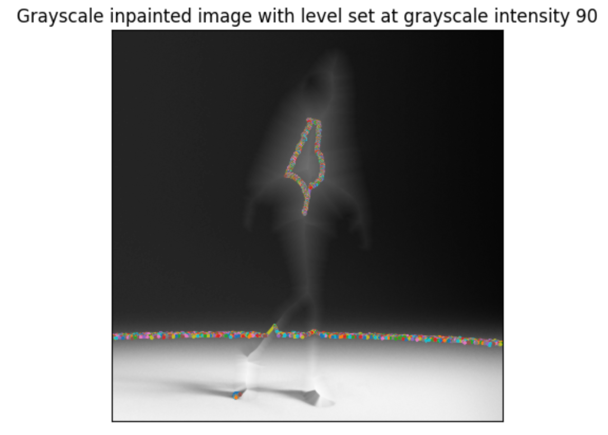
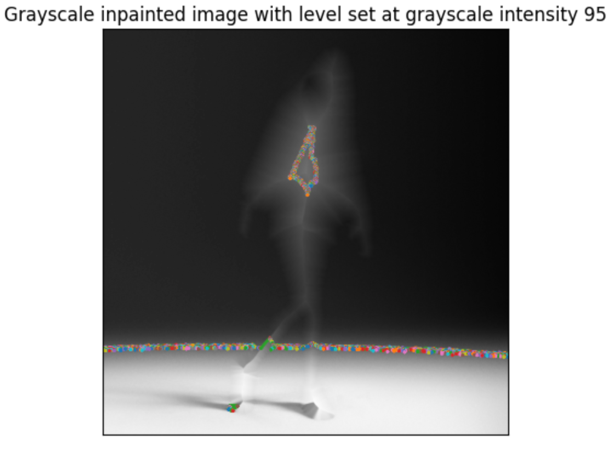












# 

# The Outpainting Problem

We want to utilize the algorithm presented in Telea’s paper (refer to [1]) based on the model described in Sethian’s paper (refer to [2] ). For the purpose we will utilize the solution of the Eikonal equation subject to appropriate boundary conditions. There are several packages which offer the solutions of the Eikonal equation and one of them is [scikit-fmm](https://github.com/scikit-fmm/scikit-fmm).

On the Figure below it is depicted the solution of the Eikonal equation for the zero contour shown on the upper left plot.

The solution is depicted without periodicities along the axes (upper right plot) and with periodicities on both axes (lower left plot) and periodicity along the y axis only (lower right plot).

A diagram of different colored lines

Description automatically generated

Figure: Solution of the Eikonal equation with open initial contour (depicted with black). Marching contours produced with [scikit-fmm](https://github.com/scikit-fmm/scikit-fmm).

The Eikonal equation to obtain the contours on Figure:

on with Dirichlet boundary condition on the zero-contour which is the *initial level set* which we want to extrapolate.

Let us look at one non-trivial outpainting scenario

river

lake

islet

Figure: Non-trivial outpainting scenario utilizing regions “rivers”, “lake”

and an “islet”

How can we solve the outpainting problem in a realistic way? We need to partition the level sets near the edge of the original image on basic blocks. Three such basic blocks are the “river”, “lake” and “islet” basic blocks shown below.

**Statement**: Outpainting based on partitioning the level sets near the original image edge will create realistic outpainting terrain. We need to inpaint the “river” and the “lakes” blocks first and then the “islet”.

Figure: “River” basic block with both Dirichlet Figure: “Lake” basic block with both Dirichlet

and Neumann boundary conditions and Neumann boundary conditions

Figure : An “Islet” block with both Dirichlet and Neumann boundary conditions

# 

# References

[0] [A Fast Introduction to Fast Marching Methods and Level Set Methods, J.A. Sethian, Berkeley, 2001](https://nike.box.com/s/zf5mhdkzhgm14me3nai6psl5ystwxuy1)

[1] [An Image Inpainting Technique Based on the Fast Marching Method, Alexandru Telea, 2004](https://nike.box.com/s/5qvr97dovsiv4phlo5h61ogdax56nnr4)

[2] [A Fast Marching Level Set Method for Monotonically Advancing Fronts, J.A. Sethian, 1996](https://nike.box.com/s/p6kk7z9khpz9ewoxcefn8o07teoyhyne)

[3] [Image Inpainting, M. Bertalmio, G. Shapiro, V. Caselles, C. Ballester, 1999](https://nike.box.com/s/qf5on6sa3p0f5gk10vgn10ft9j5rb2as)

[4] [scikit-fmm: the fast marching method for Python](https://github.com/scikit-fmm/scikit-fmm)

[5] [pyheal: Fast Marching Inpainting, minimalistic Python implementation](https://github.com/olvb/pyheal)

[6] [Boundary Value Problem, Wikipedia](https://en.wikipedia.org/wiki/Boundary_value_problem)

# Appendix

## Code implementation of the algorithm discussed in Telea’s paper

import heapq

from enum import Enum

class PixelState(Enum):

KNOWN = 1

INSIDE = 2

BAND = 3

def inpaint\_region():

{

# STEP 1: extract P(i,j) = top of the narrow\_band heap

#//**TODO:** write this code

flag[i,j] = PixelState.KNOWN

for (k,l) in [(i-1,j), (i,j-1), (i+1,j), (i,j+1)]:

if flag[k,l] != PixelState.KNOWN:

if flag[k,l] == PixelState.INSIDE:

# STEP 2:

flag[k,l] = PixelState.BAND

# STEP 3:

inpaint\_pixel(k,l)

# STEP 4:

dist\_to\_boundary[k, l] = min(solve\_eikonal(k-1,l,k,l-1),

solve\_eikonal(k+1,l,k,l-1),

solve\_eikonal(k-1,l,k,l+1),

solve\_eikonal(k+1,l,k,l+1))

# STEP 5: insert pixel (k,l) in narrow\_band heap

#//**TODO:** write this code

}

def inpaint\_pixel(...):

#//**TODO:** write this code

def solve\_eikonal(...):

#//**TODO:** write this code