Subject: Least Cost Path using Dijkstra's Algorithm
Posted by Bill[1] on Thu, 21 Oct 2010 13:03:11 GMT
View Forum Message <> Reply to Message

I would like to take a raster, which represents the "cost" of moving
through that pixel, and find the shortest path (i.e. least cost path)
through that raster. For example, using geospatial analysis and
several layers of data, this can often be used to model cross country
mobility with each raster cell represent the ease or difficulty of
moving through that cell.

Has any written an IDL program that does a least cost path from a
starting cell to an ending cell using Dijkstra's Algorithm?
http://en.wikipedia.org/wiki/Dijkstra's_algorithm

It's a simple concept but can be very memory intensive depending on
the size of your raster, in my case could easily be 20,000 samples by
20,000 lines.

Thanks for your help. I am new to IDL so was looking for a jump start.

Subject: Re: Least Cost Path using Dijkstra's Algorithm
Posted by David Fanning on Fri, 22 Oct 2010 12:15:59 GMT
View Forum Message <> Reply to Message

Bill writes:

> Hopefully, speed. Is there a reason why I shouldn't?
>
> Currently, this is something typically accomplished with a GIS
> package, such as ArcGIS. Hopefully, sometimes these processes are
> painfully slower than they need to be. I work with a colleague that
> did this in MatLab. I would like to go it with IDL, so I can
> integrate this into ArcGIS.

I think the chances of implementing this algorithm in
anything other than a "painfully slow" way in IDL
are slim and none. Too many necessary FOR loops, probably.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.

Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Least Cost Path using Dijkstra's Algorithm
Posted by ben.bighair on Fri, 22 Oct 2010 13:25:22 GMT
View Forum Message <> Reply to Message

On 10/22/10 8:15 AM, David Fanning wrote:
> Bill writes:
>
>> Hopefully, speed. Is there a reason why I shouldn't?
>>
>> Currently, this is something typically accomplished with a GIS
>> package, such as ArcGIS. Hopefully, sometimes these processes are
>> painfully slower than they need to be. I work with a colleague that
>> did this in MatLab. I would like to go it with IDL, so I can
>> integrate this into ArcGIS.
>
> I think the chances of implementing this algorithm in
> anything other than a "painfully slow" way in IDL
> are slim and none. Too many necessary FOR loops, probably.
>

Hi,

I don't want to be contrary, but I wouldn't be too quick to dismiss IDL
in general regarding the algorithm. It isn't that hard to code up the
necessary search and nodes. IDL might do just fine.

But the size of the specific problem looks pretty daunting. I'm still on
the early side of my coffee, but a 20000 x 20000 single precision array
is about 1.6 GB isn't it? Yikes!

Ben

Subject: Re: Least Cost Path using Dijkstra's Algorithm
Posted by David Fanning on Fri, 22 Oct 2010 13:31:34 GMT
View Forum Message <> Reply to Message

Ben Tupper writes:

> I don't want to be contrary, but I wouldn't be too quick to dismiss IDL
> in general regarding the algorithm. It isn't that hard to code up the
> necessary search and nodes. IDL might do just fine.
>

> But the size of the specific problem looks pretty daunting. I'm still on
> the early side of my coffee, but a 20000 x 20000 single precision array
> is about 1.6 GB isn't it? Yikes!

Alright, if Ben is writing the program I'll up the
odds to thin and none. ;-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Least Cost Path using Dijkstra's Algorithm
Posted by James[2] on Tue, 26 Oct 2010 00:19:06 GMT

View Forum Message <> Reply to Message

I recently programmed the Fast Marching Method (
http://math.berkeley.edu/~sethian/2006/Explanations/fast_mar ching_explain.html
) as an external C program for IDL. The Fast Marching Method is the
continuous version of Dijkstra's algorithm: it simulates a boundary
moving through space monotonically with speed as a function of
position. The pixel/voxel grid is considered a sampling of the
continuous speed function.

You can find least cost paths by initializing the boundary at a single
point and then backtracking from the destination(s) through the
arrival time field using gradient descent. This will give you
interpolated points that might not be at exact grid points (pixels),
but it sounds like that's what you want. The algorithm is O(n lg n)
where N is the number of pixels, just like Dijkstra's.

Anyway, if this sounds interesting to you, I can send you the C code
and the IDL wrapper function. You can compile it with MAKE_DLL. I
optimized my implementation for sparse data sets (where the speed is 0
in most places), so if you want to travel through the entire
20000*20000 array, it will be slow. I might be persuaded to write a
non-sparse version if your application is interesting enough ;-)