Subject: Re: intersection/union of two polygons
Posted by James Kuyper on Thu, 23 Jan 2003 15:09:48 GMT
View Forum Message <> Reply to Message

Roberto Monaco wrote:
>
> Hi,
>
> Does anyone know where I can find intersection and union between two
> polygons in IDL?
>
> Ideally a function that returns the polygon resulting of the intersection
> (union) of the input polygons. If not, the area (intersection / union) would
> be helpful.

I's sorry, Idon't know of such an function, but I hope someone else
does. However, I thought I should point out something: the intersection
or union of two arbitrary polygons is not necessarily itself a polygon;
it can be a disconnected set of polygons.

Subject: Re: intersection/union of two polygons
Posted by David Fanning on Thu, 23 Jan 2003 16:10:33 GMT
View Forum Message <> Reply to Message

Roberto Monaco (rmonaco@coresw.com) writes:

> Does anyone know where I can find intersection and union between two
> polygons in IDL?
>
> Ideally a function that returns the polygon resulting of the intersection
> (union) of the input polygons. If not, the area (intersection / union) would
> be helpful.

If the polygons are in the same plane, and you want
a purely graphical solution (in the sense of pixel
values), then you could try this:

1. Find the indices in each polygon with POLYFILLV.

2. Find the intersection or union of the two indices
   vectors with SetIntersection or SetUnion from my
   web page:

   http://www.dfanning.com/tips/set_operations.html

   You will probably have to add some rudimentary error
   checking to the programs you find here.

3. Find the boundary around this set of pixels with
Find_Boundary. (And, possibly, Label_Region if there
is more than one contiguous region in the union. Ben
Tupper has send me a program that does this automatically,
I think, if I can find it in the chaos around here. Let
me know if you think you need it.)

http://www.dfanning.com/programs/find_boundary.pro

Find_Boundary can return the area as well as the perimeter,
in addition to specifying the polygon of the result.

This idea may not be anything like what you want, but
the advantage is that it is simple. :-)

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: intersection/union of two polygons
Posted by meron on Thu, 23 Jan 2003 17:02:32 GMT
View Forum Message <> Reply to Message

In article <3e3002d6_1@news2.prserv.net>, "Roberto Monaco" <rmonaco@coresw.com> writes:
> Hi,
>
> Does anyone know where I can find intersection and union between two
> polygons in IDL?
>
> Ideally a function that returns the polygon resulting of the intersection
> (union) of the input polygons. If not, the area (intersection / union) would
> be helpful.

You can check out the routine SHAPE_OVERLAP in my library (MIDL, can
find it on the IDL users contributions page). This will give you the
intersection, proceeding from there to union should be simple enough.
Then, if you want, you can apply SHAPE_AREA (from same library) to the
result.

Mati Meron | "When you argue with a fool,

Subject: Re: intersection/union of two polygons
Posted by JD Smith on Thu, 23 Jan 2003 17:49:45 GMT
View Forum Message <> Reply to Message

On Thu, 23 Jan 2003 07:51:04 -0700, Roberto Monaco wrote:

> Hi,
>
> Does anyone know where I can find intersection and union between two
> polygons in IDL?
>
> Ideally a function that returns the polygon resulting of the
> intersection (union) of the input polygons. If not, the area
> (intersection / union) would be helpful.
>

I had to solve a simple subset of this problem: intersect arbitrary polygons against a square grid of pixels. Ideally, it should work for clipping two arbitrary polygons. There are libraries for doing this quite generally, but for normal polygons without holes or waists you can easily put something together using the very simple Sutherland-Hodgeman algorithm (search for that, it's about day 3 of intro computer graphics courses).

Unfortunately, the algorithm doesn't vectorize in any way I've yet found, and is pretty slow in straight IDL loops. Since I literally had hundreds of thousands of polygons to clip, I ended up writing it in C and linking to it externally. I can make the IDL version available if you're interested. It has the additional constraint of working only for convex polygons. This can easily be fixed by augmenting the intermediate storage array size.

A side note on writing external code: I had a conundrum -- the clipper was too slow, but I didn't want to deal with the hassle of compiling C code on all the different architectures on which my code would be run once distributed. Neither did I want to provide C-compiling support to users. Luckily, RSI has provided an excellent if little-known capability to solve just this problem: MAKE_DLL, which I used to advantage with the help of one of RSI's engineers. It's essentially an internal, system-specific Makefile, shielding you from all the hateful compile and link incantations required to actually build a shared library for use with CALL_EXTERNAL.

But what if MAKE_DLL fails, due, for instance, to a Windows user not having the proper compiler (not shipped by default)? Well, my

solution was to keep the old, slow, IDL-loop based method in place,
and only use the compiled version if compilation succeeds, which I
test only once per IDL session.

Here's the recipe:

At the beginning of the code which is deciding whether to use the
compiled or internal method:

```
common mycode_external,polyclip_compiled,polyclip_path
if n_elements(polyclip_compiled) eq 0 then begin
 catch,g err
 if err ne 0 then begin   ; any failure in compiling, just use the IDL vers
    polyclip_compiled=0
 endif else begin
    resolve_routine,'polyclip'
    path=(routine_info('polyclip',/SOURCE)).PATH
    path=strmid(path,0,strpos(path,path_sep(),/REVERSE_SEARCH))
    make_dll,'polyclip','polyclip',INPUT_DIRECTORY=path, $
            DLL_PATH=polyclip_path ;,/REUSE_EXISTING
    polyclip_compiled=1
 endelse
 catch,/cancel
endif
```

There I'm being cheeky and locating my polyclip.c file by knowing it's
in the directory with my polyclip.pro file. Also, I actually have
keywords to recompile or to switch to the internal non-compiled code
unconditionally, for testing purposes, but omitted these for clariy.

And actually deciding which method to use looks like:

```
if polyclip_compiled eq 0 then begin ; IDL version
  blah
  polyclip,i,j,px_out,py_out
  blah
endif else begin      ; Compiled code, use call_external and the shared lib
  blah
  tmp=call_external(polyclip_path,'polyclip',$
                    VALUE= $
                    [0b,0b,1b,   0b,0b,1b,    0b,    0b,    0b,   0b], $
                    vi,vj,npix,  px,py,npol,  px_out,py_out,areas,ri_out)
  blah
endelse
```

This works very well. If things are setup for MAKE_DLL to succeed,
the code will run faster. If not, no error will occur, and the code
will still run, producing the same results, albeit more slowly.

Someone once commented that it would be nice for IDL to permit assembly-style, in-place callouts to C; this is as close to that as is possible. All the benefits of external code with none of the pain.

JD
