

---

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 clarity.

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

---