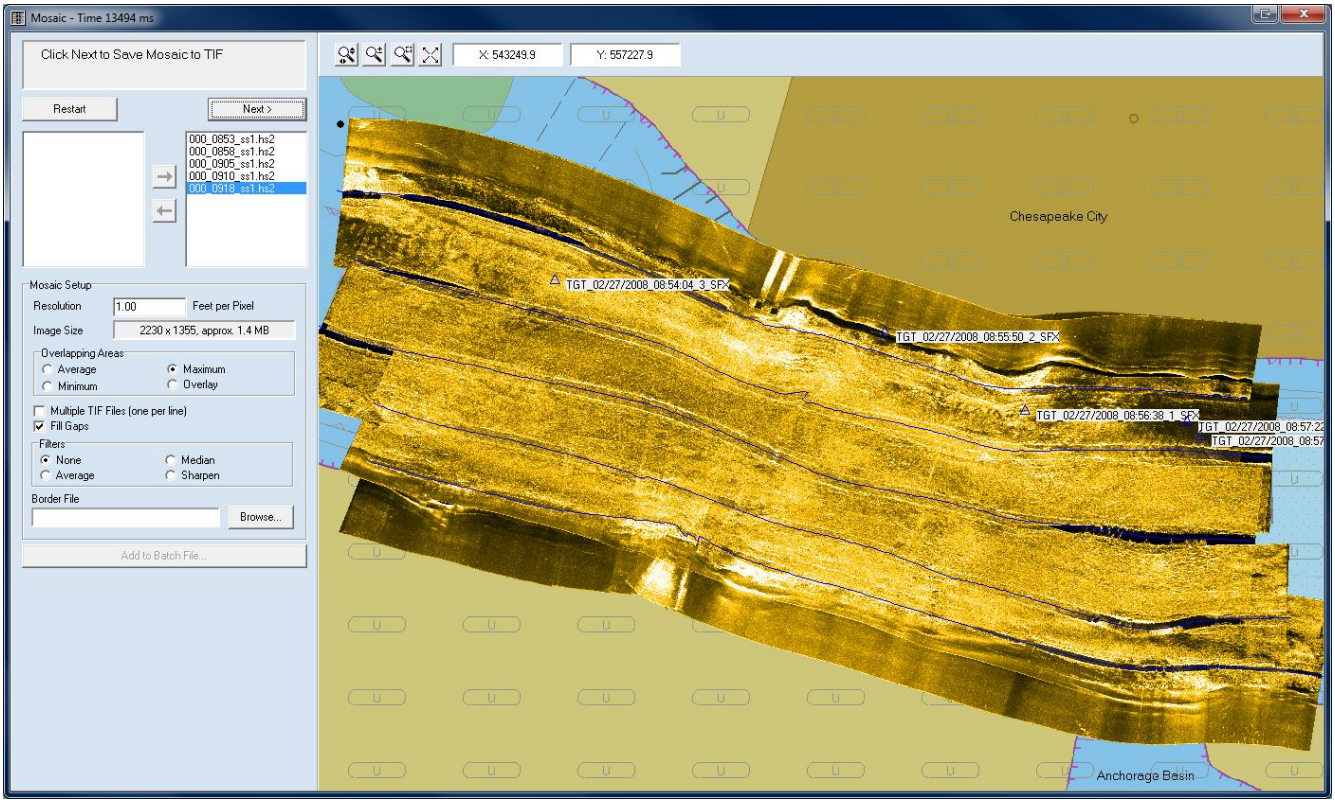
HYSCAN Multi-core Speed and Memory Tests

By Dave Maddock

HYSCAN (SIDE SCAN TARGETING AND MOSAICKING) version 11.0.8 and newer includes improvements to speed and memory requirements and support for multi-core processing in Stage 3 mosaicking. I made some mosaics using several different configurations to determine how the new multi-core support compares against the older versions. Using the same data set and settings, several scenarios were compared:

- 11.0.7 in single-core mode,
- 11.0.8 in single-core mode, and
- 11.0.8 in multi-core mode with both 2 and 8 cores.

FIGURE 1. *Test Data Set. 5 Overlapping Lines at 1ft Resolution*



Earlier versions mosaicked in a two-pass methodology. In the first pass, each line was mosaicked by itself and saved to a temp file. In the second pass, the individual lines were merged. Version 11.0.8 is able to accomplish the same effect in a single pass which provides significant speed improvements even in single-core mode. Another effect of eliminating the second pass is a reduction in the amount of memory requirements. To merge lines, HYSCAN needed to have two mosaics open in memory at the same time. Since this is no longer necessary in 11.0.8 and later, one can expect a corresponding increase in the effective upper limit of mosaic size that your computer can generate.

Table 1 displays the results of my tests. The speed increase as a percentage of 11.0.7 time saved is shown in black, bold font. In blue, bold font is the result of turning off the 'fill gaps' option.

TABLE 1. *HYSCAN Mosaicking Speed Comparison (in seconds)*

	Fill Gaps			**No Gaps**				
	11.0.7	*11.0.8*		*11.0.7*		*11.0.8*		
single-core	22105	13494	**39%**	12277	**44%**	11451	**7%**	**15%**
dual-core		10234	**54%**			6942	**43%**	**32%**
8-core	16708	5257	**69%**	10390	**38%**	2200	**79%**	**58%**

Of course, the first thing we can say is that, in every case, 11.0.8 is faster. Depending on your hardware and mosaic settings, one can expect anywhere from a 7% to 79% speed increase. Just streamlining the algorithms and methodology results in a drastic 39% savings with 'fill gaps' enabled on single-core computers. Theoretically, one might expect that doubling the number of cores would cut processing time in half. In practice, some time is lost when a thread has to wait for its turn to update the mosaic so we will never achieve this ideal. We can see this effect when comparing 'no gaps' mode to 'fill gaps' mode. 'Fill gaps' generates a lot more mosaic updates than 'no gaps' does and consequently more 'thread collisions' that reduces the multi-core efficiency.

Let's explore the effect of turning off 'fill gaps' mode in detail. Not only does this make the multi-core hardware more efficient, it can make even the 11.0.7 version faster. If you recall my prior *Sounding Better* article on determining your optimal mosaic resolution, you know that if you have chosen your resolution wisely, this option is largely unnecessary. The sample rate and ping rate are sufficient to get good coverage in your mosaic at optimal resolution. Notice that when comparing 11.0.7 to itself 'no gaps' is 44% faster in comparison with 'fill gaps' and my final mosaic looks identical. In the 11.0.8 blue column, we see that turning 'fill gaps' off can save us an additional 15-58% *after multi-core savings*.

To put this in perspective, compare 11.0.7 single-core with 'fill gaps' and 11.0.8 dual-core, 'no gaps.' This represents a time savings of 69% over 11.0.7. In other words, a dual-core computer without filling gaps performs comparable to a 8-core computer that is filling gaps! And remember, if the resolution is optimal the resulting mosaic will be almost identical.