

Proposal for a new file format for handling line scanner data within PolyWorks

Objectives

- Define a new hardware-neutral file format that can be used to write line scanner data with digitizing vector information into a file.
- Make sure that the information needed by the PolyWorks to auto-organize the scans and compute normal vector information is available in the file format.
- Propose the file format to hardware manufacturers that have not yet defined their format.

General

- The format will be a binary file format.
- In this document, it is assumed that the type 'int' is 4-byte integer number.
- Unlike other PolyWorks formats, it will be written using a little-endian byte-ordering scheme (Intel processors). No byte swapping operations are needed for reading/writing the file format on a laptop/workstation using Intel or compatible processors.
- The extension of the file format will be .psl (PSL), which stands for PolyWorks Scan Line format.
- The format will have a global header specifying a magic number, a format version and the number of passes. A pass is a set of consecutive scan lines that have been captured while the trigger is down (articulated arm) or with the same orientation of the laser head (CMM setup).
- For each pass, it is necessary to specify the number of scan lines. For each scan line, it is necessary to specify the digitizing vector and the list of (x,y,z) points captured in this line.

Global header information

- The PSL format will have a fixed 512-byte header. The following C structure could be used to read/write the PSL header:

```
struct psl_header
{
char magic[4];    /* The first four bytes must be "PSLF" in capital letters. */
```

```

int version;      /* Specifies the version. Must be 1 at this stage. */
char comments[128]; /* 128 characters for adding a comment string. */

int nb_passes;    /* Specifies the number of passes written to the file. */
int digitizing_vector_flag;
                /* If set to 0: The digitizing vectors go from the digitizer to the part
                (opposite direction to surface orientation). If set to 1: The
                digitizing vectors go from the part to the digitizer (similar direction
                to surface orientation). */
int reserved [92]; /* Reserved for future use */

};

```

Passes and scan lines

After the global header, passes are directly written one after the other in the binary file. Each pass starts with a pass header, followed by a series of scan lines. Each scan line has a scan line header, followed by a list of points. The pass and scan line headers are fixed 64-byte headers. The following C structures could be used to read/write the passes:

For each pass (the number of passes is specified in psl_header), write the following header:

```

struct pass_header
{
int nb_scan_lines; /* Number of scan lines to follow after the pass header */
int scanner_id;
                /* Integer number that identifies a scanning device. Could be
                useful for multiple scanner configuration. For single scanners,
                write '0'. */
int reserved [14];
};

```

After the pass header is written, all the pass' scan lines are written. A scan line consists in two sections: a header and a data section. The scan line header is as follows:

```

struct scan_line_header
{
int nb_points; /* Number of x,y,z points to follow after the scan line header. */
float ijk[3];
                /* The digitizing vector for this line. Its orientation is set according to the
                digitizing_vector_flag set in the psl header. */

int reserved[12];
};

```

The data section contains `nb_points` triplets of (x,y,z) coordinates specified as single-precision floating point numbers. They should be written as follows:
`x1,y1,z1,x2,y2,z2,x3,y3,z3, ...`