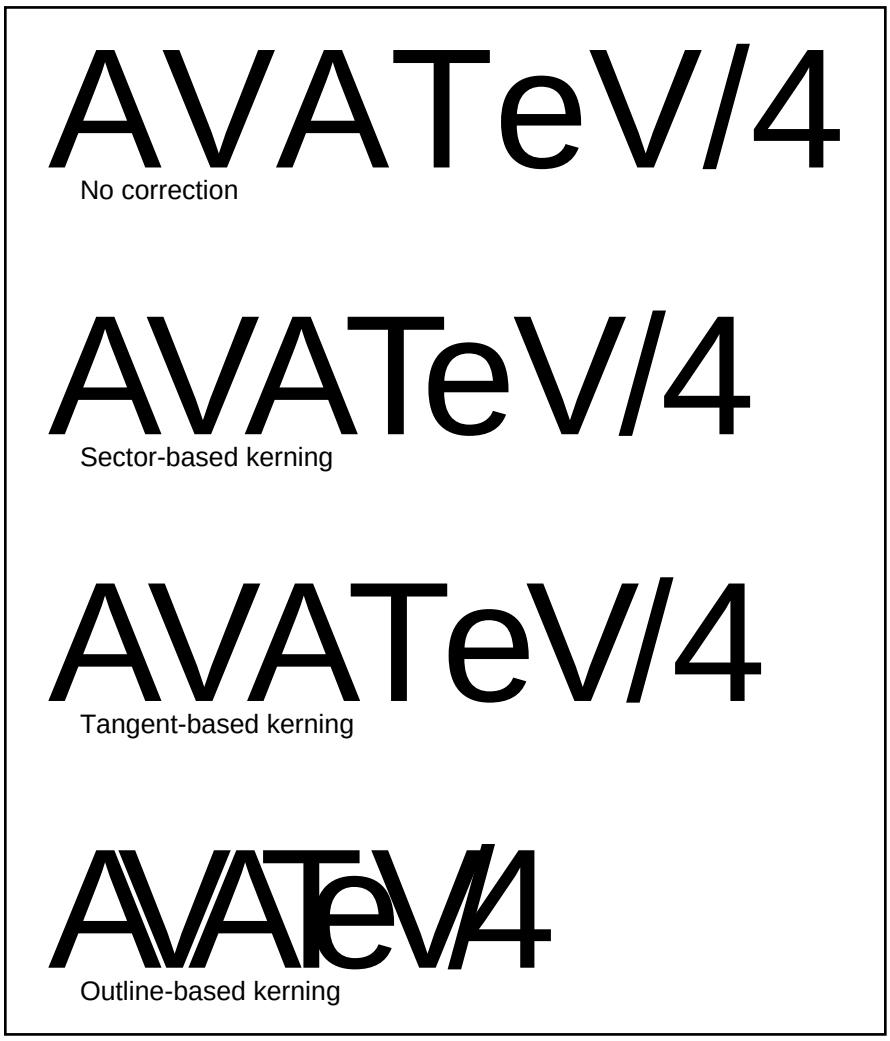
kerndict.ps



Copyright (c) 2003 Thomas Baruchel

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

```
%!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% KERNDICT a PostScript dictionary for automatical kerning                     %
%   written by Thomas Baruchel (may 2003)                                      %
%                                                                              %
% Several methods are provided                                                 %
%   * Tangent-based kerning (an idea of the author)                            %
%     45° tangents encapsulate the letter at the right or at the left.         %
%     How far they are from the left or right side of the BoundingBox          %
%     tells what can be done. The method is intended to be very quick.         %
%     Usage : { kernproc } (Hello) kshow                                       %
%                                                                              %
%   * Sector-based kerning (a classical method)                                %
%     The user chooses the length of the smallest horizontal gap               %
%     between the two letters. The number of sectors gives the                 %
%     precision of the computation.                                            %
%     Usage : { 175 12 kernsector } (Hello) kshow                              %
%     Warning : not intended to take account of spaces                         %
%                                                                              %
%   * Outline-based kerning                                                    %
%     The user chooses the real distance between the characters even           %
%     if the two closest points of the outline are not at the same             %
%     height. The user must build an outline version of the font he            %
%     uses before calling the procedure. The /StrokeWidth key of this          %
%     outline font must be set to the expected distance. A number of           %
%     sectors give the precision of the computation.                           %
%     Usage : { /Outline-Font findfont 12 kernoutline } (Hello) kshow          %
%     Warning : not intended to take account of spaces                         %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/kerndict 5 dict dup begin
/min { dup 2 index ge { pop } { exch pop } ifelse } bind def
/kerncompute { 1 index 1 index mul 0 ge { abs exch abs min
neg 3 div } { pop pop 0 } ifelse } bind def
/kernproc { gsave (..) dup 0 4 -1 roll put dup 1 4 -1 roll put
currentfont dup /FontMatrix get matrix invertmatrix makefont setfont
newpath 0 0 moveto dup true charpath flattenpath pathbbox pop exch pop
add exch [1 -1 -1 -1 0 0] setmatrix newpath 0 0 moveto true charpath
[1 0 0 1 0 0] setmatrix pathbbox sub 3 index sub 4 1 roll sub sub
grestore kerncompute currentfont /FontMatrix get matrix currentmatrix
exch concat exch 0 rmoveto setmatrix } bind def
/kernsector { matrix currentmatrix currentfont dup dup /FontMatrix get
dup concat gsave newpath matrix invertmatrix makefont setfont /FontBBox
get aload pop 2 {3 index neg} repeat moveto exch 3 index sub 2 mul (.)
dup 0 11 -1 roll put true charpath currentpoint pop (.) dup 0 12 -1 roll
put true charpath currentpoint pop exch flattenpath 0 moveto 3 -1 roll 4
-1 roll sub dup 0 exch rlineto clip dup 7 -1 roll div 0 0 2 index 5 -1
roll { gsave 0 exch 5 index 4 index rectclip clippath pathbbox pop exch
pop sub min grestore } for exch pop add exch pop add 3 -1 roll sub neg
grestore 0 rmoveto setmatrix } bind def
/kernoutline { matrix currentmatrix currentfont /FontMatrix get concat
gsave 3 -1 roll dup /FontMatrix get matrix invertmatrix makefont dup
setfont /FontBBox get aload pop 2 {3 index neg} repeat moveto exch 3
index sub 2 mul (.) dup 0 10 -1 roll put true charpath currentpoint pop
(.) dup 0 11 -1 roll put true charpath currentpoint pop exch flattenpath
0 moveto 3 -1 roll 4 -1 roll sub dup 0 exch rlineto clip dup 7 -1 roll
div 0 0 2 index 5 -1 roll { gsave 0 exch 5 index 4 index rectclip
clippath pathbbox pop exch pop sub min grestore } for exch pop add exch
pop add neg grestore 0 rmoveto setmatrix } bind def
end def
```