MicroTalk – pdfsplit

MicroTalk – pdfsplit

Luigi Scarso

Abstract: MicroTalk is a short and technical paper that shows some unusual, hopefully useful, ideas following the schema “figure to code”. The main topic is always typographic programming in ConTEXt & Lua. A bit of Lua code, the `\clip` macro and Leptonica extensions are the ingredients for this recipe to cook a pdfsplit macro that take a pdf and try to split into parts as the `\vsplit` does with `\vbox`es.

Key words: Lua, Leptonica, PDF slicing, SWIG, MuPDF, Sumatra.

Abstrakt: Článek poukazuje na způsob nařezání PDF na proužky tak, jak je to známé pomocí příkazu `\vsplit` s `\vbox`y.

Klíčová slova: Lua, Leptonica, vysekávání z PDF, SWIG, MuPDF, Sumatra.

References

[1] Leptonica. Available at URL: `http://www.leptonica.com/`
[2] MuPDF – a lightweight PDF viewer and toolkit written in portable C. Available at URL: `http://www.mupdf.com/`
[3] Sumatra PDF. Available at URL: `http://en.wikipedia.org/wiki/Sumatra_PDF`

luigi (dot) scarso (at) gmail (dot) com
Padova, Italy

doi: 10.5300/2011-2-4/102

1 Slice a pdf

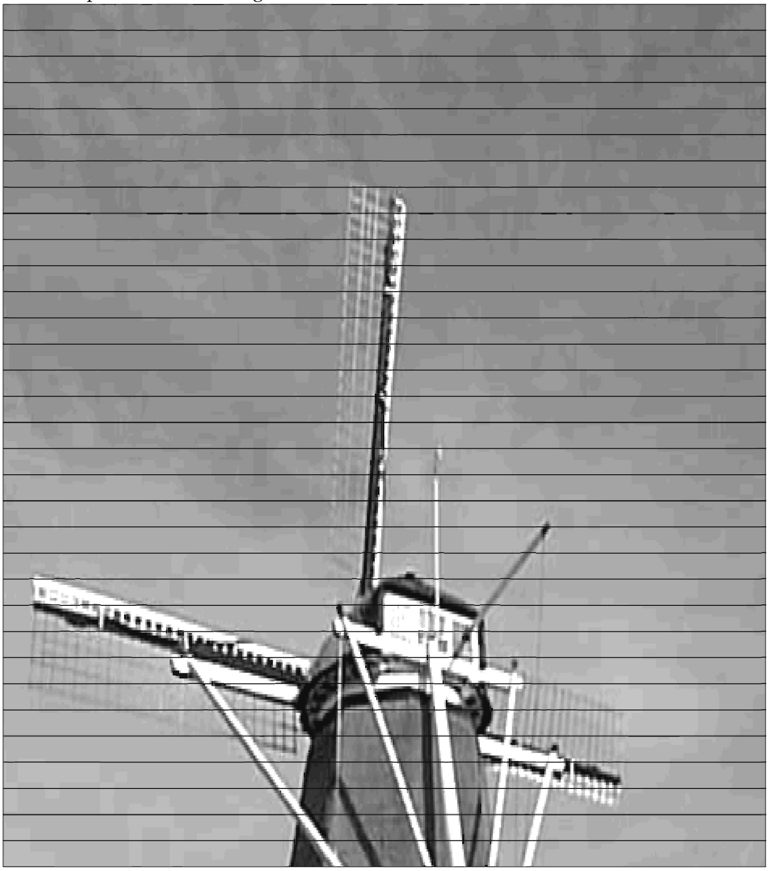
Let's start with a bit of Lua code:

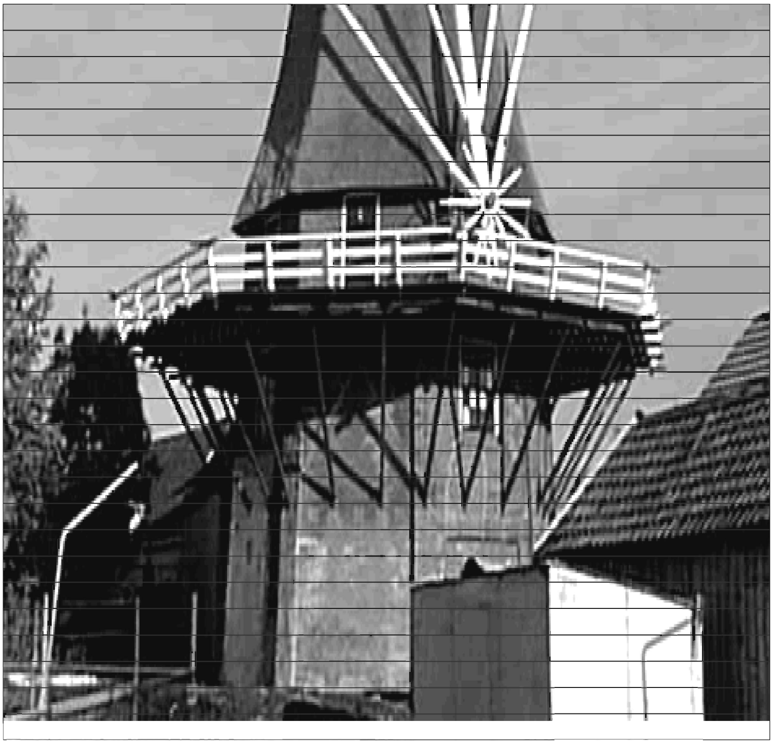
```
\startluacode
document.lscarso = document.lscarso or {}
function document.lscarso.LuaSliceIt(Fig,H,W,L,FigOpt)
 local H =  math.floor(string.gsub(H,"pt","")*2^16)
 local W =  math.floor(string.gsub(W,"pt","")*2^16)
 local L =  math.floor(string.gsub(L,"pt","")*2^16)
 local vh = 0
 local step = L
 local h = step
 local S = ""
 local FigOpt = FigOpt or ""
 while vh <= H do
   S =string.format("{\\clip[voffset=\%dsp,
                             width=\%dsp,
                             height=\%dsp]
                            {\\externalfigure[\%s][\%s]}}
                            \\par\\nointerlineskip\\blank[1sp]",
                            vh,W,h,Fig,FigOpt)
   tex.sprint(tex.ctxcatcodes,S)
   vh = vh + step
 end
 if math.mod(H,step) > 0 then
      vh = vh - step + math.mod(H,step)
      S =string.format("{\\clip[voffset=\%dsp,
                             width=\%dsp,
                             height=\%dsp]
                            {\\externalfigure[\%s][\%s]}}
                            \\par\\nointerlineskip\\blank[1sp]",
                            vh,W,math.mod(H,step),Fig,FigOpt)
      tex.sprint(tex.ctxcatcodes,S)
 end
end
\stopluacode
```

`document.lscarso.LuaSliceIt` takes a pdf and "slices it" with slices of height L.

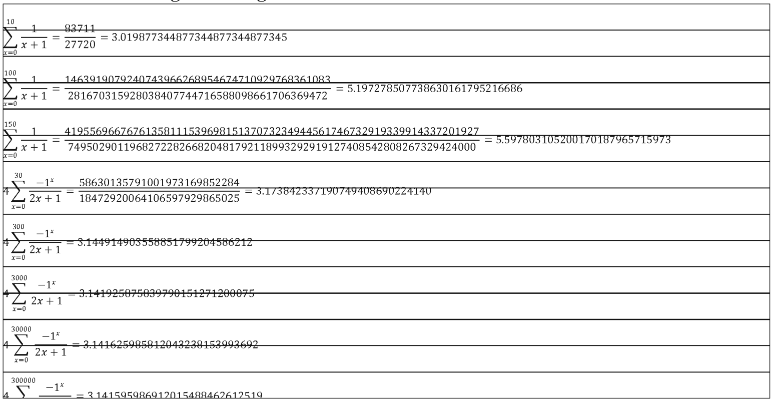
```
\bgroup
\newdimen\Hfig \newdimen\Wfig
\setbox1000=\hbox{\externalfigure[mill.png][width=\textwidth,
                                             height=1.3\textheight]}
\Wfig=\wd1000 \Hfig=\dimexpr\ht1000+\dp1000\relax%
\ctxlua{document.lscarso.LuaSliceIt("mill.png","\the\Hfig",
                         "\the\Wfig","\the\dimexpr 1.0\lineheight\relax",
                         "width=\\textwidth,height=1.3\\textheight")}
\egroup
```

It's not a problem with images:





But uniform slicing is wrong with text:



$$\sum_{x=0}^{10} \frac{1}{x+1} = \frac{83711}{27720} = 3.0198773448773448773448773345$$

$$\sum_{x=0}^{100} \frac{1}{x+1} = \frac{14639190792407439662689546747109297683610831}{281670315928038407744716588098661706369472} = 5.1972785077386301617952166 86$$

$$\sum_{x=0}^{150} \frac{1}{x+1} = \frac{4195569667676135811153969815137073234944561746732919339914337201927}{749502901196827228266820481792118993292919127408542808267329424000} = 5.5978031052001701879657159 73$$

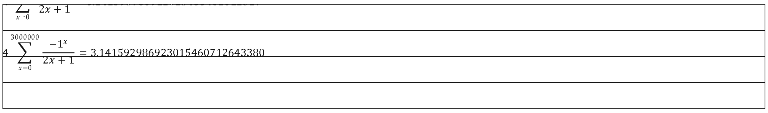
$$4\sum_{x=0}^{30} \frac{-1^x}{2x+1} = \frac{586301357910019731698522 84}{184729200641065979298650 25} = 3.1738423371907494086902241 40$$

$$4\sum_{x=0}^{300} \frac{-1^x}{2x+1} = 3.1449149035588517992045862 12$$

$$4\sum_{x=0}^{3000} \frac{-1^x}{2x+1} = 3.1419258758397901512712000 75$$

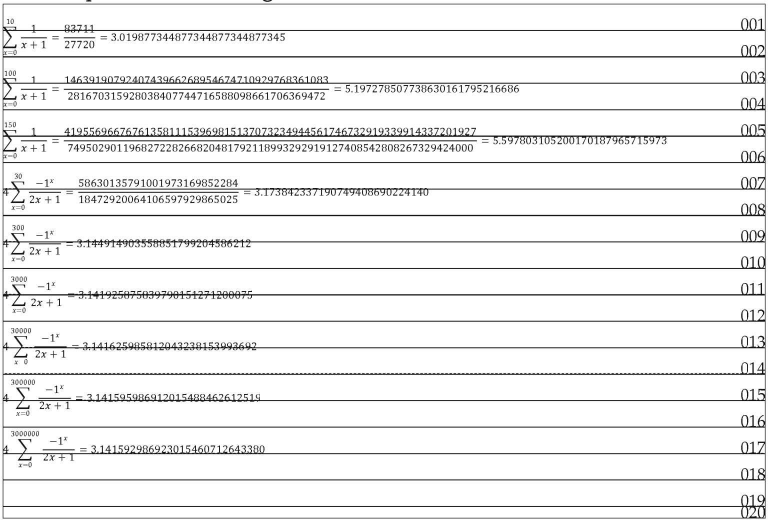
$$4\sum_{x=0}^{30000} \frac{-1^x}{2x+1} = 3.1416259858120432381539936 92$$

$$4\sum^{300000} \frac{-1^x}{} = 3.1415959869120154884626125 19$$



$4\sum_{x=0}^{3000000}\frac{-1^x}{2x+1} = 3.1415929869230154607126433380$

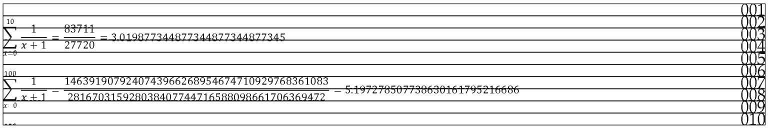
We can enumerate each slice, and observe that it depends only on the given `step`: with `step = 1.0\lineheight` we have



$\sum_{x=0}^{10}\frac{1}{x+1} = \frac{83711}{27720} = 3.0198773448773448773448773345$
$\sum_{x=0}^{100}\frac{1}{x+1} = \frac{14639190792407439662689546747109297683610083}{2816703159280384077447165880986617063694 72} = 5.1972785077386301617952166886$
$\sum_{x=0}^{150}\frac{1}{x+1} = \frac{419556966767613581115396981513707323494456174673291933991433720192 7}{749502901196827228266820481792118993292919127408542808267329424000} = 5.5978031052001701879657159 73$
$4\sum_{x=0}^{30}\frac{-1^x}{2x+1} = \frac{586301357910019731698522 84}{184729200641065979298650 25} = 3.1738423371907494086902241 40$
$4\sum_{x=0}^{300}\frac{-1^x}{2x+1} = 3.1449149035588517992045862 12$
$4\sum_{x=0}^{3000}\frac{-1^x}{2x+1} = 3.1419258758397901512712000 75$
$4\sum_{x=0}^{30000}\frac{-1^x}{2x+1} = 3.1416259858120432381539936 92$
$4\sum_{x=0}^{300000}\frac{-1^x}{2x+1} = 3.1415959869120154884626125 19$
$4\sum_{x=0}^{3000000}\frac{-1^x}{2x+1} = 3.1415929869230154607126433 80$
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020

It's now clear that 002, 004, 006... are good break points while 001, 003, 005... must be avoided:. if we collect all good break points we can then subdivide the pdf into the right `\vbox`es. These break points depend on `lineheight`, but we can choose another `step`, let's say 2mm, so that slices are independent from the body font:

```
\bgroup
\newdimen\Hfig \newdimen\Wfig
\setbox1000=\hbox{\externalfigure[pari_example.pdf][width=\textwidth]}
\Wfig=\wd1000 \Hfig=\dimexpr\ht1000+\dp1000\relax%
\ctxlua{document.lscarso.LuaSliceItNR("pari_example.pdf","\the\Hfig",
                          "\the\Wfig","\the\dimexpr 2mm\relax",
                          "width=\\textwidth")}
\egroup
```



$\sum_{x=0}^{10}\frac{1}{x+1} = \frac{83711}{27720} = 3.0198773448773448773448773345$
$\sum_{x=0}^{100}\frac{1}{x+1} = \frac{14639190792407439662689546747109297683610083}{281670315928038407744716588098661706369472} = 5.1972785077386301617952166886$
001
002
003
004
005
006
007
008
009
010

011

$$\sum_{x=0}^{150}\frac{1}{x+1}=\frac{41955696676761358111539698151370732349445617467329193399143372019 27}{74950290119682722826682048179211899329291912740854280826732942400 0}=5.5978031052001701879657159 73$$

012
013
014
015

$$4\sum_{x=0}^{30}\frac{-1^x}{2x+1}=\frac{586301357910019731698522 84}{184729200641065979298650 25}=3.1738423371907494086902241 40$$

016
017
018
019

$$4\sum_{x=0}^{300}\frac{-1^x}{2x+1}=3.1449149035588517992045862 12$$

020
021
022
023
024

$$4\sum_{x=0}^{3000}\frac{-1^x}{2x+1}=3.1419258758397901512712000 75$$

025
026
027
028

$$4\sum_{x=0}^{30000}\frac{-1^x}{2x+1}=3.1416259858120432381539936 92$$

029
030
031
032
033

$$4\sum_{x=0}^{300000}\frac{-1^x}{2x+1}=3.1415959869120154884626125 19$$

034
035
036
037

$$4\sum_{x=0}^{3000000}\frac{-1^x}{2x+1}=3.1415929869230154607126433 80$$

038
039
040
041
042

We group together slices `1,5,9,13,14,18,22,26,31,35,39`:

001

$$\sum_{x=0}^{10}\frac{1}{x+1}=\frac{83711}{27720}=3.0198773448773448773448773 45$$

005

$$\sum_{x=0}^{100}\frac{1}{x+1}=\frac{146391907924074396626895467471092976836108 3}{281670315928038407744716588098661706369472}=5.1972785077386301617952166 86$$

009

$$\sum_{x=0}^{150}\frac{1}{x+1}=\frac{41955696676761358111539698151370732349445617467329193399143372019 27}{74950290119682722826682048179211899329291912740854280826732942400 0}=5.5978031052001701879657159 73$$

013
014

$$4\sum_{x=0}^{30}\frac{-1^x}{2x+1}=\frac{586301357910019731698522 84}{184729200641065979298650 25}=3.1738423371907494086902241 40$$

018

$$4\sum_{x=0}^{300}\frac{-1^x}{2x+1}=3.1449149035588517992045862 12$$

022

$$4\sum_{x=0}^{3000}\frac{-1^x}{2x+1}=3.1419258758397901512712000 75$$

026

$$4\sum_{x=0}^{30000}\frac{-1^x}{2x+1}=3.1416259858120432381539936 92$$

031

$$4\sum_{x=0}^{300000}\frac{-1^x}{2x+1}=3.1415959869120154884626125 19$$

035

$$4\sum_{x=0}^{3000000}\frac{-1^x}{2x+1}=3.1415929869230154607126433 80$$

039
042

Now slicing works well:

001

$$\sum_{x=0}^{10}\frac{1}{x+1}=\frac{83711}{27720}=3.0198773448773448773448773 45$$

005

$$\sum_{x=0}^{100}\frac{1}{x+1}=\frac{146391907924074396626895467471092976836108 3}{281670315928038407744716588098661706369472}=5.1972785077386301617952166 86$$

009

$$\sum_{x=0}^{150}\frac{1}{x+1}=\frac{41955696676761358111539698151370732349445617467329193399143372019 27}{74950290119682722826682048179211899329291912740854280826732942400 0}=5.5978031052001701879657159 73$$

013
014

$$4\sum_{x=0}^{30}\frac{-1^x}{2x+1}=\frac{586301357910019731698522 84}{184729200641065979298650 25}=3.1738423371907494086902241 40$$

018

$$4\sum_{x\ 0}^{300}\frac{-1^x}{2x+1} = 3.1449149035588517992045862l2$$

022

$$4\sum_{x\ 0}^{3000}\frac{-1^x}{2x+1} = 3.14192587583979015127120007５$$

026

$$4\sum_{x=0}^{30000}\frac{-1^x}{2x+1} = 3.1416259858120432381539936９2$$

031

$$4\sum_{x=0}^{300000}\frac{-1^x}{2x+1} = 3.14159598691201548846261251９$$

035

$$4\sum_{x=0}^{3000000}\frac{-1^x}{2x+1} = 3.14159298692301546071264338０$$

039

042

```
\bgroup
\newdimen\Hfig \newdimen\Wfig
\setbox1000=\hbox{\externalfigure[pari_example.pdf][width=\textwidth]}
\Wfig=\wd1000 \Hfig=\dimexpr\ht1000+\dp1000\relax%
\startluacode
 document.lscarso.GoodPoints = {1,5,9,13,14,18,22,26,31,35,39}
\stopluacode
\ctxlua{document.lscarso.LuaSliceItAndCollect("pari_example.pdf",
                "\the\Hfig","\the\Wfig","\the\dimexpr 2mm\relax",
                "width=\\textwidth","")}
\egroup

function document.lscarso.LuaSliceItAndCollect(Fig,H,W,L,FigOpt)
 local H =  math.floor(string.gsub(H,"pt","")*2^16)
 local W =  math.floor(string.gsub(W,"pt","")*2^16)
 local L =  math.floor(string.gsub(L,"pt","")*2^16)
 local vh = 0
 local step = L
 local h = step
 local S = ""
 local FigOpt = FigOpt or ""
 local NR = 0
 local Payload = "{\\ruledhbox{\\clip[voffset=\%dsp,width=\%dsp,height=\%dsp]
{\\externalfigure[\%s][\%s]}\\llap{\\tfx\%03d}}}
\\par\\nointerlineskip\\blank[1sp]"
 local GoodPoint = document.lscarso.GoodPoints or {}
 local j = 1
 local Vboxes = {}
 local prevvh = 0
 while vh < H do
   NR =NR +1
   vh = vh + step
   if GoodPoint[j] == NR then
      Vboxes[#Vboxes+1] = {from=prevvh,to=vh,mark=NR}
      --tex.sprint(tex.ctxcatcodes,"(",prevvh,",",vh,") ")
      prevvh = vh
      j = j +1
    end
  end
  if math.mod(H,step) == 0 then H = H +1  end
  if math.mod(H,step) > 0 then
    NR = NR +1
    vh = vh - step + math.mod(H,step)
    Vboxes[#Vboxes+1] = {from=prevvh,to=vh,mark=NR}
  end
  Payload = "\\ruledvbox{\\hbox{\\clip[voffset=\%dsp,width=\%dsp,height=\%dsp]
{\\externalfigure[\%s][\%s]}\\llap{\\tfx\%03d}}}
\\par\\nointerlineskip\\blank[1sp]"
  for i,v in ipairs(Vboxes) do
     S =string.format(Payload,v.from,W,v.to-v.from,Fig,FigOpt,v.mark)
     tex.sprint(tex.ctxcatcodes,S)
  end
```

It's now rather trivial to modify this macro to build a \vbox to be used with \vsplit.

2 Leptonica

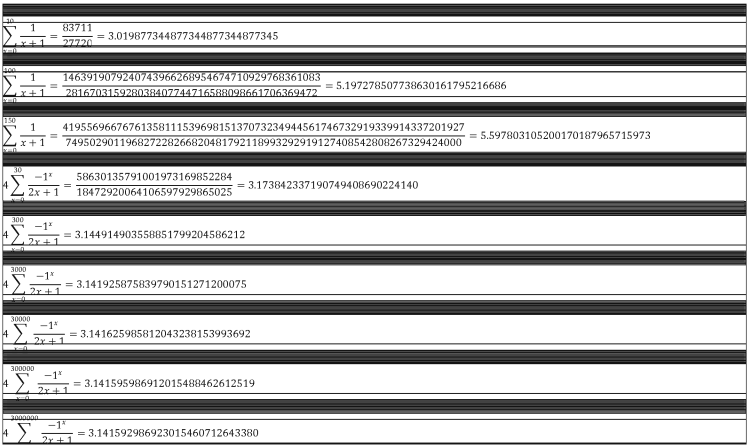
Leptonica (`http:\\www.leptonica.com`) is a pedagogically-oriented open source site containing software that is useful for image processing and image analysis applications. It's not difficult to build a binding for Lua with SWIG. To use Leptonica we must first convert the pdf in a black and white bitmap with

```
pdftoppm -mono -r 72 -f 1 -l 1 pari_example.pdf pari_example
```

(this saves page 1 in `pari_example-000001.pbm` with a resolution of 72dpi). Then we scan the bitmap with `lept_get_breaks` searching for a white row and we store the y coordinates in `GoodPoints`. In the end `LuaCollect` builds the appropriate `vboxes`.

```
\bgroup
\newdimen\Hfig \newdimen\Wfig
\setbox1000=\hbox{\externalfigure[pari_example.pdf][width=\textwidth]}
\Wfig=\wd1000\relax\Hfig=\dimexpr\ht1000+\dp1000\relax%
\executesystemcommand{pdftoppm -mono -r 72 -f 1 -l 1 pari_example.pdf
pari_example}
\ctxlua{document.lscarso.GoodPoints =
           document.lscarso.lept_get_breaks("pari_example-000001.pbm",72)}
\ctxlua{document.lscarso.LuaCollect("pari_example.pdf",
                              "\the\Hfig","\the\Wfig",[[width=\textwidth]])}
\egroup
```

Let's see what happens:



$$\sum_{x=0}^{10} \frac{1}{x+1} = \frac{83711}{27720} = 3.0198773448773448773448773448773345$$

$$\sum_{x=0}^{100} \frac{1}{x+1} = \frac{14639190792407439662689546747109297683610 83}{2816703159280384077447165880986617063694 72} = 5.19727850773863016179521 6686$$

$$\sum_{x=0}^{150} \frac{1}{x+1} = \frac{4195569667676135811153969815137073234944561746732919339914337201927}{7495029011968272282668204817921189932929191274085428082673294240 00} = 5.5978031052001701879657 15973$$

$$4\sum_{x=0}^{30} \frac{-1^x}{2x+1} = \frac{58630135791001973169852284}{18472920064106597929865025} = 3.17384233719074940869022 4140$$

$$4\sum_{x=0}^{300} \frac{-1^x}{2x+1} = 3.1449149035588517992045 86212$$

$$4\sum_{x=0}^{3000} \frac{-1^x}{2x+1} = 3.1419258758397901512712 00075$$

$$4\sum_{x=0}^{30000} \frac{-1^x}{2x+1} = 3.1416259858120432381539 93692$$

$$4\sum_{x=0}^{300000} \frac{-1^x}{2x+1} = 3.1415959869120154884626 12519$$

$$4\sum^{3000000} \frac{-1^x}{2x+1} = 3.1415929869230154607126 43380$$

A good breakpoint is marked with an horizontal rule: a black stripe means that good points are very tight there.

```
require("leptonica")
function document.lscarso.lua_pixGetPixel(pixs,x,y)
    local scratch = 0
    local res = ''
    scratch = leptonica.uti_getref_l_uint32()
    if (leptonica.pixGetPixel(pixs,x,y,scratch) == 0) then
       res =  leptonica.uti_valref_l_uint32(scratch)
    else
     print("! Error on ",x,y)
     res = ''
    end
    return res
end

function document.lscarso.lept_get_breaks(filename,res)
     -- os.execute([[pdftoppm -mono filename]])
     local mainName = filename
     local pixs = leptonica.pixRead(mainName)
     local lua_pixGetPixel = document.lscarso.lua_pixGetPixel
     local GoodPoint = {}  -- document.lscarso.GoodPoints or {}
     if pixs then
        local w =  leptonica.pixGetWidth(pixs)
        local h =  leptonica.pixGetHeight(pixs)
        local d =  leptonica.pixGetDepth(pixs)
        local out = ''
        local bit
        local storey = true
        local go
        GoodPoint[1] = 0
        for y=0,h-1 do
           out = ''
           storey = true
           for x=0,w-1 do
              bit =  lua_pixGetPixel(pixs,x,y)
              if bit == 1 then
                storey = false
                break
              end
           end
           if storey and (math.floor(0.5+ (y/res) *72.27*2^16)
              ~= GoodPoint[#GoodPoint]) then
```

```
                GoodPoint[#GoodPoint +1] = math.floor(0.5+ (y/res)
*72.27*2^16)
           end
           storey = true
        end
        GoodPoint[#GoodPoint +1] = math.floor(0.5+ (h/res) *72.27*2^16)
      else
       tex.sprint("ERROR")
      end
      return GoodPoint
end


function document.lscarso.LuaCollect(Fig,H,W,FigOpt)
 local H =  math.floor(string.gsub(H,"pt","")*2^16)
 local W =  math.floor(string.gsub(W,"pt","")*2^16)
 local vh = 0
 local h = step
 local S = ""
 local FigOpt = FigOpt or ""
 local NR = 0
 local Payload = ""
 local GoodPoint = document.lscarso.GoodPoints or {}
 local j = 1
 local Vboxes = {}
 local prevvh = 0
 local truey =  GoodPoint[#GoodPoint]
 local y_ratio = H / truey
 for i,v in ipairs(GoodPoint) do
     if GoodPoint[i+1] == nil then
        break
     end
     NR = NR +1
     Vboxes[#Vboxes+1] = {from=v,to=GoodPoint[i+1],mark=NR}
 end
 Payload = "\\ruledvbox{\\hbox{\\clip[voffset=\%dsp,width=\%dsp,height=\%dsp]
          {\\externalfigure[\%s][\%s]}}}\\par\\nointerlineskip\\blank[1sp]"
 for i,v in ipairs(Vboxes) do
    S =string.format(Payload,y_ratio*v.from,W,y_ratio*(v.to-v.from),Fig,FigOpt)
    tex.sprint(tex.ctxcatcodes,S)
 end
end
```

3 MuPDF

We can replace the pdftoppm call by building a Lua binding to MuPDF, "a light-weight PDF viewer and toolkit written in portable C" (http:\\www.mupdf.com). Sumatra (http://en.wikipedia.org/wiki/Sumatra_PDF) is a fast pdfviewer based on MuPDF.

Despite MuPDF is a well written library and there are several examples that explain how to use it, it's not easy to write a converter.
What follow is only a part of an almost literal translation into Lua of the C program pdfdraw.c and even so the debugging is difficult.

```
function drawpbm(pagenum)
   local xref = pdfdraw.targetpdf.xref
   local drawrotate = pdfdraw.drawrotate
   local drawzoom   = pdfdraw.drawzoom
   local drawbands  = pdfdraw.drawbands
   local drawpattern= pdfdraw.drawpattern
   drawloadpage(pagenum)
   local drawpage    = pdfdraw.drawpage
   local drawcache   = pdfdraw.drawcache
   local bbox,w,h, bh
   local pix
   local fd,name
   name        = pdfdraw.outname
   local  ctm = mupdf.fz_identity()
   ctm = mupdf.fz_concat(ctm, mupdf.fz_translate(0, - drawpage.mediabox.y1))
   ctm = mupdf.fz_concat(ctm, mupdf.fz_scale(drawzoom, - drawzoom))
   ctm = mupdf.fz_concat(ctm, mupdf.fz_rotate(drawrotate + drawpage.rotate))
   bbox = mupdf.fz_roundrect(mupdf.fz_transformrect(ctm, drawpage.mediabox))
   w = bbox.x1 - bbox.x0;
   h = bbox.y1 - bbox.y0;
   bh = h / drawbands;
   if NotNil(drawpattern) then
      fd = io.open(name,'wb')
      if (fd == nil) then
        die(fz_throw("ioerror: could not create raster file '%s'", name));
      end
      fd:write(string.format("P4\n%d %d\n", w, h));
   end
   pix = mupdf.fz_newpixmap(mupdf.pdf_devicergb, bbox.x0, bbox.y0, w, bh)
   mupdf.fz_clearpixmap(pix, 0xFF);
   for b = 0,  drawbands-1 do
      local dev, error
      dev = mupdf.fz_newdrawdevice(drawcache, pix)
      error = mupdf.pdf_runcontentstream(dev, ctm, xref, drawpage.resources, drawpage.contents)
      if (error>0) then
        die(fz_rethrow(error, "cannot draw page %d in PDF file '%s'", pagenum, basename))
      end
      mupdf.fz_freedevice(dev)
      if NotNil(drawpattern) then
        for y = 0, pix.h -1 do
          local column = y * pix.w * 4
           local dst = {}
           local bit = 0
          local r = 0
          for x = 0, pix.w-1 do
             local v = (1+(mupdf.uti_samples_arr_getitem(pix.samples,(x * 4 + 1) +
column)))*77
             v = v + (1+(mupdf.uti_samples_arr_getitem(pix.samples,(x * 4 + 2) + column)))*150
             v = v + (1+(mupdf.uti_samples_arr_getitem(pix.samples,(x * 4 + 3) + column)))*28
              v = math.floor(v/256)
             local d = 1
             if v > 200 then d = 0 end
              r = r + (d*2^(7-bit))
             bit = bit +1
             if bit == 8 then
               bit = 0
              dst[#dst+1] = string.char(r)
              r = 0
             end
          end
           if bit > 0 then
            dst[#dst+1] = string.char(r)
          end
           fd:write(table.concat(dst))
        end
      end
      pix.y = pix.y + bh;
      if (pix.y + pix.h > bbox.y1) then
        pix.h = bbox.y1 - pix.y;
      end
  end
  mupdf.fz_droppixmap(pix)
  if NotNil(drawpattern) then
    fd:close()
  end
  drawfreepage()
  print()
end
```

The complete binding is truly more difficult to debug: even if SWIG does an excellent work, it's necessary to manage the details of implementation of the library.
In this case `pdftopbm` does already an excellent work, it's well tested, stable and ready to use; the call of an external program is not so expensive compared to a plug in.

4 Conclusion

Lua code is much nearer to plain standard Lua: ConTEXt offers some shortcuts that is better to learn. For example `math.floor(string.gsub("10pt","pt","")*2^16)` can be replaced by `string.todimen("10pt")` and possibly other things related to print.

The algorithm that finds a good breakpoint is rather simple: for example the are some unwanted good breakpoints between a $\sum$ and the its subscript. This leads to break an object that should not be broken even if it has some white rows inside. A practical solution is to consider a line with thickness grater than 1 pixel (but it's better to use metric dimensions).

These ideas are also valid for scanned text but then we must take care of noise (and Leptonica can help a lot here).

Finally, it's better to consider carefully the opportunity of a binding. As shown with MuPFD, sometimes the traditional way is still the best way.

About µTalk

µTalk is a short and technical paper that shows some unusual, hopefully useful, ideas following the schema "figure $\rightarrow$ code". The main topic is always typographic programming in ConTEXt & Lua.