Report on the creation and prior methodology for converting images to Revit and AutoCAD detail items

By Luke McDevitt

# **Previous methods used**

**Option 1:**

Insert image directly into Revit / AutoCAD and trace every line manually.

This can be made faster on Revit by clicking on the pdf and checking “enable snaps” under properties. This will allow you to use the “Pick Lines” option when drawing detail lines to automatically place lines that are clicked. (This takes a long time and **only works on vector graphic PDFs**)

Pros: accurate and exactly what you want to recreate. Not limited to file type (can use .jpg .png .pdf)

Cons: Very slow and cannot be transferred between AutoCAD and Revit

**Option 2:**

Use Importpdf command on AutoCAD.

If the PDF contains vector graphics it will automatically be placed in as detail lines that are editable in AutoCAD.

Pros: very fast, easy, and accurate.

Cons: only works on AutoCAD and **requires a PDF that has vector graphics**.

**Option 3:**

Convert AutoCAD drawings to detail lines using either ArchSmarter ToolBox Revit macros or the ArchSmarter Add-in for Revit.

This can be done by using Option 2 to convert a PDF to an AutoCAD drawing (.DWG file) that can be imported into Revit using Import CAD in the Insert menu. Navigate to the macro manager or the Add-ins tab. Find Convert DWG to Detail lines and select the CAD drawing you would like to convert. This will draw detail lines in Revit over the CAD insert. After this is done the CAD model can be removed leaving the detail lines. This **only works on vector graphic PDFs** or AutoCAD drawings (dxf or dwg).

Pros: Allows for AutoCAD drawings to be transferred into Revit without quality loss, extremely fast compared to tracing.

Cons: only works if you already have an AutoCAD drawing of the detail item.

**Option 4:**

Use online converters or programs to convert from an image (.png or .jpg) to a vector image. Many exist but here are a few that I have used successfully:

[SVGcode—Convert raster images to SVG vector graphics](https://svgco.de/) is a fully online image to vector converter.

How to create Revit detail item from image (.png .jpg NOT .pdf) using SVGcode:

Select Open Image -> Select the image you want converted -> wait for conversion -> Save SVG

Go to a SVG to DXF conversion website like <https://cloudconvert.com/svg-to-dxf> and then use Option 3 to convert the image to a Revit detail item if you need it for more than an AutoCAD detail item.

Pros: Support for converting .jpg and .png to detail items and Increased customization

Cons: requires the use of multiple online conversions and constant internet connection + lower quality and more cleanup than tracing

[Draw Freely | Inkscape](https://inkscape.org/) is a program that allows for vectorization of images and exporting in .DXf format.

How to create Revit detail item from image (.pdf .png .jpg .bmp) using online Inkscape:

Inkscape requires bitmap (.BMP) files for vectorization so unless you already have a .bmp file a converter is required.

[JPG to BMP | CloudConvert](https://cloudconvert.com/jpg-to-bmp) or [JPG to BMP (Online & Free) — Convertio](https://convertio.co/jpg-bmp/)

[PNG to BMP | CloudConvert](https://cloudconvert.com/png-to-bmp) or [PNG to BMP (Online & Free) — Convertio](https://convertio.co/png-bmp/)

[PDF to BMP | CloudConvert](https://cloudconvert.com/pdf-to-bmp) or [PDF to BMP (Online & Free) — Convertio](https://convertio.co/pdf-bmp/)

Once you have the .bmp file install and run Inkscape (This will require a restart). Once Inkscape is fully installed load the .bmp file using this interface:

A screenshot of a computer

Description automatically generated with medium confidence

Select Browse other files (if you have previous files) or Load (if you do not) and load the .bmp file using these settings:

Graphical user interface, text, application

Description automatically generated

Once your image is imported select the path option at the top of the page.

Graphical user interface, application, timeline

Description automatically generated

Select the “Trace Bitmap” option in the dropdown menu and select the bitmap.

This will open an interface that will allows for a lot of customization of how you want the detail item to look. Just pressing “Apply” will offer a fair vectorization of the image if it is not noisy.

Graphical user interface

Description automatically generated with medium confidence

After you have applied the vectorization settings to go File -> Save As -> DXF

Graphical user interface

Description automatically generated

Save the file as a DXF. This will allow you to use Option 3 to convert the DXF to a Revit detail item.

Pros: Highly Customizable, can use more file types like .pdf (even ones without vector graphics) .png .jpg.

Cons: requires a conversion into bitmap online, downloading / learning new software, lower quality and more cleanup than tracing.

# **New Solution for Image to Detail Items**

Option 5:

Custom programmed solution.

My program PIC2CAD when run will prompt a user to select an image (.pdf .png .jpg) and convert it into a DXF file through the use of Potrace image Vectorization Algorithm.

How to use PIC2CAD:

In the Installation folder there will be 2 .exe executables; Potrace-1.11-64-bit-setup.msi and PIC2CAD\_McD.exe.

Graphical user interface, text, application

Description automatically generatedFirst run Potrace-1.11-64-bit-setup.msi and setup Potrace to your machine. Afterwards run pic2cad. You will be greeted with a popup like this:Graphical user interface, application

Description automatically generated

Select More info -> Run anyway. The initial setup requires an internet connection. There will then be a popup to open a file. Select whichever file you want converted -> Open. This will convert the image to a .dxf through a series of steps depending on the file type given. This .dxf can then be converted to a Revit detail item through Option 3.

Pros: Accepts .pdf .png .jpg fast and easy

Cons: less customization, more cleanup, lower quality than tracing.

# **Code Explanation**

The program was written in python and packaged into an executable using the pyinstaller package.

The code is 96 lines and is split into a few distinct sections that I will do my best to explain.

Each of these three steps requires a series of sub steps to accomplish

Packages to include

import cv2  
from PIL import Image  
import os  
from zipfile import ZipFile  
import urllib.request  
from tkinter import Tk # from tkinter import Tk for Python 3.x  
from tkinter.filedialog import askopenfilename  
from pdf2image import convert\_from\_path

Creating a folder in the “Documents” directory

documentspath = os.path.expanduser('~/Documents')  
directory = "PIC2CAD\_McD"  
parent\_dir = documentspath  
folderpath = os.path.join(parent\_dir, directory)  
if not os.path.exists(folderpath):  
 os.mkdir(folderpath)  
 print("Directory '% s' created" % directory)

Prompt user to select a file

Tk().withdraw() # we don't want a full GUI, so keep the root window from appearing  
filename = askopenfilename() # show an "Open" dialog box and return the path to the selected file  
print(filename)

Splitting file path into the important parts of the path

filename3, file\_extension = os.path.splitext(filename)

Converting input to png if it is a pdf

Converting to png if file is jpg

if file\_extension == r".jpg":  
 im1 = Image.open(filename)  
 im1.save(filename3 + '.png')  
 newjpg2png = (filename3 + '.png')  
 myfile2 = f"{newjpg2png}"

if file\_extension == r".pdf":  
 print("it is a pdf")  
 if not os.path.exists(folderpath + r'/Release-22.04.0-0.zip'):  
 popzipfile = (folderpath + r'/Release-22.04.0-0.zip')  
 urllib.request.urlretrieve(  
 "https://github.com/oschwartz10612/poppler-windows/releases/download/v22.04.0-0/Release-22.04.0-0.zip",  
 popzipfile)  
 print("downloading poppler")  
 file\_name = (folderpath + '/Release-22.04.0-0.zip')  
 with ZipFile(file\_name, 'r') as zip:  
 # extracting all the files  
 print('Extracting all the files now...')  
 zip.extractall(folderpath)  
 print('Done!')  
 poppler\_path = (folderpath + r'\poppler-22.04.0\Library\bin')  
 images = convert\_from\_path(filename, poppler\_path=poppler\_path)  
 for image in images:  
 image.save(filename3 + '.png')  
 newpngfilepath = (filename3 + '.png')  
 myfile2 = f"{newpngfilepath}"

Converting the png to a bitmap

if file\_extension == r".png":  
 myfile2 = f"{filename}"  
print("myfile2 is " + myfile2)  
  
# read the image file  
img = cv2.imread(myfile2)  
ret, bw\_img = cv2.threshold(img, 135, 255, cv2.THRESH\_BINARY)  
# converting to its binary form  
bw = cv2.threshold(img, 135, 255, cv2.THRESH\_BINARY)  
# save image as bitmap  
myfile3 = myfile2.replace(".png", ".bmp")  
cv2.imwrite(myfile3, bw\_img)  
print("converted to bitmap")

Downloading and installing Potrace zip that the Potrace setup will reference

Opening terminal and interfacing with Potrace

variable = r'potrace "myfile3" -b dxf -t 10 --progress'  
variable2 = variable.replace("myfile3", myfile3)

def mycmd():  
 os.system(variable2)  
mycmd()  
  
print("Converted to DXF")

if not os.path.exists(folderpath + "/potrace-1.16.win64.zip"):  
 ...  
 print("path to zip: " + folderpath + "\potrace-1.16win64.zip")  
 potracezip = (folderpath + "/potrace-1.16.win64.zip")  
 urllib.request.urlretrieve("http://potrace.sourceforge.net/download/1.16/potrace-1.16.win64.zip", potracezip)  
 file\_name = (folderpath + '/potrace-1.16.win64.zip')  
 with ZipFile(file\_name, 'r') as zip:  
 # printing all the contents of the zip file  
 zip.printdir()  
  
 # extracting all the files  
 print('Extracting all the files now...')  
 zip.extractall(folderpath)  
 print('Done!')  
 setx = ('setx potrace ' + folderpath + "\potrace-1.16.win64\potrace.exe")  
 def mycmd():  
 os.system(setx)  
 mycmd()

# Line by Line Code Explanation

The commands from and import pull information from a package module or library outside of the base commands afforded by python.

Os.path.expanduser allows for the path to a directory to be pulled from its name, in this case the documents folder is used. This will act as the parent directory for the creation of the folder PIC2CAD\_McD where Poppler and Potrace will be downloaded and unzipped. The code first checks if the folder already exists via os.path.exists() before creating it to reduce time to run the program.

The tkinter modules (in this case tkinter.filedialog) are used for interfacing with the user. The askopenfilename() command is used to pull the path of the file selected.

Os.path.splitext is used to separate the path from the file extension, for example separating image.jpg into the variables image and .jpg

If that file extension variable is a .pdf then it must be converted into a .png. This is done through the use of a package called Poppler, but the entirety of the dependencies cannot be contained within the package / module itself and must use the urllib.request.urlretrieve(url, path) command to download a file from the internet into a specific directory location. In this case the location is in the PIC2CAD\_McD file created previously and the file is the Poppler zip file. Next we unzip the folder downloaded using the package Zipfile and command Zip.extractall to the same path. This unzipped path is then used in the command convert\_from\_path(file\_path, poppler\_path) to convert the pdf to png. The code first checks if the Poppler folder already exists via os.path.exists() before downloading and unzipping it to reduce time to run the program.

If file extension is .jpg conversion to .png from .jpg is done via the PIL module and Image.open() and save() commands.

Conversion from .png to .bmp is done via thresholding. cv2.threshold() takes the arguments image path, minimum threshold for black, maximum threshold for black, and cv2.THRESH\_BINARY. A typical threshold is cv2.threshold(img, 135, 255, cv2.THRESH\_BINARY) where img is the cv2.imread(path) output variable. After that replace the .png with .bmp and save image as a new file via imwrite(path, image type).

To download the necessary parts of Potrace the urllib.request.urlretrieve(url, path) command is used again to download the zip archive of potrace to the folder PIC2CAD\_McD in Documents. It is then unzipped using the Zipfile and Zip.extractall() commands. Then the command terminal is opened via def mycmd(): command, messages are sent to the terminal via os.system(vars), and the terminal is closed via mycmd() again. The variable sent in this case is 'setx potrace ' + folderpath + "\potrace-1.16.win64\potrace.exe" to set a path variable to the potrace .exe file within the unzipped Potrace folder. The code first checks if the Potrace folder already exists via os.path.exists() before downloading and unzipping it to reduce time to run the program.

To interface with Potrace commands are sent to the terminal again via def mycmd(): and the message sent to the terminal is 'potrace "myfile3" -b dxf -t 10 --progress' where myfile3 is the path to the bitmap. -b dxf is the file type output, -t 10 is to de-speckle or in this case “deturd” the output vector image. Where 10 is the “turdsize”. –progress is to give a progress bar, this is for aesthetics.

Full Code:

**Github link here**

References:

[Potrace (sourceforge.net)](http://potrace.sourceforge.net/)

# **The next steps in image to detail items**

The issues with the new method are a) lack of OCR (text recognition) and b) AI image upscaling as a preprocessing step to increase quality.