

# Pixels to Print Size

---

Ref:

1. <https://photo.stackexchange.com/questions/456/is-there-a-general-formula-for-image-size-vs-print-size>
2. [Omni Calculator logo](#)

A general formula:

$$\frac{Width[pixels]}{PrintRes[pixPerInch]} = printWidth[Inch]$$
$$\frac{Height[pixels]}{PrintRes[pixPerInch]} = printHeight[Inch]$$

1 Inch = 25.4 millimetre

The size of a Stamp Size photo = 25 mm (width) x 35 mm (height)

That means, the dimension of a Stamp Size photo in pixels at 300 DPI should be:

$$Width = \frac{25}{25.4} \times 300 = 295.275590511 \simeq 295.3 \text{ pix } [ \because 1 \text{ Inch} = 25.4 \text{ Millimeter} ]$$

$$Height = \frac{35}{25.4} \times 300 = 413.385826772 \simeq 413.4 \text{ pix } [ \because 1 \text{ Inch} = 25.4 \text{ millimetre} ]$$

The same formula in reverse:

$$printWidth[Inches] \times printResolution[pixPerInch] = width[pixels]$$

$$printHeight[Inches] \times printResolution[pixPerInch] = height[pixels]$$

Thus,

$$Width = \frac{25}{25.4} \times 300 = 295.275590511 \simeq 295.3 \text{ pix } [ \because 1 \text{ Inch} = 25.4 \text{ millimetre} ]$$

$$Height = \frac{35}{25.4} \times 300 = 413.385826772 \simeq 413.4 \text{ pix } [ \because 1 \text{ Inch} = 25.4 \text{ millimetre} ]$$

---

The size of an Employee ID Card Photo = 35 mm (width) x 45 mm (height)

Which means,

$$Width = \frac{35}{25.4} \times 300 = 413.385826772 \simeq 413.4 \text{ pix } [ \because 1 \text{ Inch} = 25.4 \text{ millimetre} ]$$

$$\text{Height} = \frac{45}{25.4} \times 300 = 531.496062992 \simeq 531.5 \text{ pix } [\because 1 \text{ Inch} = 25.4 \text{ millimetre}]$$

---

## How to Understand Pixels, Resolution, and Resize Your Images in Photoshop Correctly

If an image is 4500 x 3000 pixels it means that it will print at 15 x 10 inches if you set the resolution to 300 dpi, but it will be 62.5 x 41.6 inches at 72 dpi. While the size of your print does change, you are not resizing your photo (image file), you are just reorganizing the existing pixels.

### converting the coordinates of a 300 dpi image to coordinates of a 72 dpi image - Stack Overflow

To convert from 300DPI to 72DPI, you need to multiply by 72/300, not the other way round. Do it in floating point or the multiplication first and division then, as in  $(x * 72) / 300$ . PDF units are always 1/72 of an inch.

Scaling down the original image is not a good idea, since the loss of information will reduce the output text quality.

Our Stamp Size photo at 72 DPI will essentially be  $25 \times \frac{300}{72} = 25 \times 4.166666667 \simeq 25 \times 4.17 = 104.25$  millimetre (Width)

by

$$35 \times \frac{300}{72} = 35 \times 4.166666667 \simeq 35 \times 4.17 = 145.95 \text{ millimetre (Height)}$$

---

<https://superuser.com/questions/1635868/how-can-i-with-linux-easily-create-a-collage-of-passport-photos>

<https://ostechnix.com/how-to-create-a-montage-from-images-in-linux/>

<https://opensource.com/article/21/9/photo-montage-imagemagick>

<https://imagemagick.org/script/command-line-options.php>

<https://github.com/dpar39/ppp>

---

Install ImageMagick and Montage (GraphicsMagick):

```
yes | sudo apt install imagemagick graphicsmagick-imagemagick-compat
```

## Examples:

```
montage IMG_20211008_132718.jpg IMG_20211008_132718.jpg IMG_20211008_132718.jpg
IMG_20211008_132718.jpg IMG_20211008_132718.jpg IMG_20211008_132718.jpg -geometry
+2+3 out.jpg
```

```
montage \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  IMG_20211008_132718.jpg \
  -tile 3x4 \
  -geometry '+2+2' \
  -resize 893x1663 \
  4_6_out.jpg
```

## Image size and space between the images

`-tile` option helps you to instruct how the images should be arranged and placed on the montage.

`-tile 3x4` means 3x4 tiles in aggregate, 3 columns and 4 rows.

`-tile x1` means one row only.

`-resize 893x1663` means produce a 893x1663 pixel output montage.

The option `-geometry` helps you to set the thumbnail size and the space between each image. The default `-geometry` setting in Montage is `'120x120>+4+3'`. Geometry can be expressed either as `'120x120>+4+3'` or `'+4+3'`. The first instruction `'120x120>+4+3'` means it will produce 120×120 pixel thumbnails with 4 pixels on the left/right of each image and 3 pixels below/above, where `>` denotes the resize option. The second instruction `'+4+3'` specifies the gap between images that are going to be placed on the output montage, without being specific about scaling the input. The first option is useful for shrinking the size of the input images that are of higher dimensions, although the option is completely unnecessary. It is only useful when the user does not want to set the dimension of the final output. Even then, using this option may

affect the output files undesirably. If you have to get an output of a specific dimension, use the `-resize` option instead.

```
'width x height > +4 px spacing on the left/right +3 px spacing below/above'
```

Take note of the first command that specifies the dimension along with the gaps, since it can produce undesirable output. The `-resize` option will accurately determine the dimension of the final output montage. Thus, we will not specify the dimension in the `-geometry` option. We will restrict our use of the `-geometry` option to specify the gaps.

---

Command line arguments accepted by Montage:

GraphicsMagick 1.3.40 2023-01-14 Q16 <http://www.GraphicsMagick.org/>  
Copyright (C) 2002-2023 GraphicsMagick Group.  
Additional copyrights and licenses apply to this software.  
See <http://www.GraphicsMagick.org/www/Copyright.html> for details.  
Usage: montage montage [options ...] file [ [options ...] file ...]

Where options include:

-adjoin	join images into a single multi-image file
-affine matrix	affine transform matrix
-authenticate value	decrypt image with this password
-background color	background color
-blue-primary point	chromaticity blue primary point
-blur factor	apply a filter to blur the image
-bordercolor color	border color
-borderwidth geometry	border width
-colors value	preferred number of colors in the image
-colormap type	alternate image colormap
-comment string	annotate image with comment
-compose operator	composite operator
-compress type	image compression type
-crop geometry	preferred size and location of the cropped image
-debug events	display copious debugging information
-define values	Coder/decoder specific options
-density geometry	horizontal and vertical density of the image
-depth value	image depth
-display server	query font from this X server
-dispose method	Undefined, None, Background, Previous
-dither	apply Floyd/Steinberg error diffusion to image
-draw string	annotate the image with a graphic primitive
-encoding type	text encoding type
-endian type	multibyte word order (LSB, MSB, or Native)
-fill color	color to use when filling a graphic primitive
-filter type	use this filter when resizing an image
-flip	flip image in the vertical direction
-flop	flop image in the horizontal direction
-font name	font to use when annotating with text
-format string	output formatted image characteristics
-frame geometry	surround image with an ornamental border
-gamma value	level of gamma correction
-geometry geometry	preferred tile and border sizes
-gravity direction	which direction to gravitate towards
-green-primary point	chromaticity green primary point
-help	print program options
-interlace type	None, Line, Plane, or Partition
-label name	assign a label to an image
-limit type value	Disk, File, Map, Memory, Pixels, Width, Height or Threads resource limit
-log format	format of debugging information
-matte	store matte channel if the image has one

-mattecolor color	color to be used with the -frame option
-mode type	Frame, Unframe, or Concatenate
-monitor	show progress indication
-monochrome	transform image to black and white
-noop	do not apply options to image
+page	reset current page offsets to default
-page geometry	size and location of an image canvas
-pointsize value	font point size
-quality value	JPEG/MIFF/PNG compression level
-red-primary point	chromaticity red primary point
+repage	reset current page offsets to default
-repage geometry	adjust current page offsets by geometry
-resize geometry	resize the image
-rotate degrees	apply Paeth rotation to the image
-sampling-factor HxV[,...]	horizontal and vertical sampling factors
-scenes range	image scene range
-set attribute value	set image attribute
+set attribute	unset image attribute
-shadow	add a shadow beneath a tile to simulate depth
-sharpen geometry	sharpen the image
-size geometry	width and height of image
-strip	strip all profiles and text attributes from image
-stroke color	color to use when stroking a graphic primitive
-strokewidth value	stroke (line) width
-texture filename	name of texture to tile onto the image background
-thumbnail geometry	resize the image (optimized for thumbnails)
-tile geometry	number of tiles per row and column
-title string	thumbnail title
-transform	affine transform image
-transparent color	make this color transparent within the image
-treedepth value	color tree depth
-trim	trim image edges
-type type	image type
-verbose	print detailed information about the image
-version	print version information
-virtual-pixel method	Constant, Edge, Mirror, or Tile
-white-point point	chromaticity white point

In addition to those listed above, you can specify these standard X resources as command line options: -background, -bordercolor, -borderwidth, -font, -mattecolor, or -title

By default, the image format of `file' is determined by its magic number. To specify a particular image format, precede the filename with an image format name and a colon (i.e. ps:image) or specify the image type as the filename suffix (i.e. image.ps). Specify 'file' as '-' for standard input or output.

---

Now we will calculate the values to be submitted to Montage to generate a 4"x6" photo paper filled with Stamp Size Photo IDs.

We are limited to 72 DPI since Montage works with that resolution. That will not affect our workflow for now.

To calculate the values for the options, we will have to calculate the dimension of a 4"x6" photo at 300 DPI.

We know the formula:

$$\textit{printWidth}[\textit{Inches}] \times \textit{printResolution}[\textit{pixPerInch}] = \textit{width}[\textit{pixels}]$$

$$\textit{printHeight}[\textit{Inches}] \times \textit{printResolution}[\textit{pixPerInch}] = \textit{height}[\textit{pixels}]$$

That means, the pixel dimension of a 4"x6" photo at 300 DPI is:

$$4 \times 300 = 1200 \text{ pix}$$

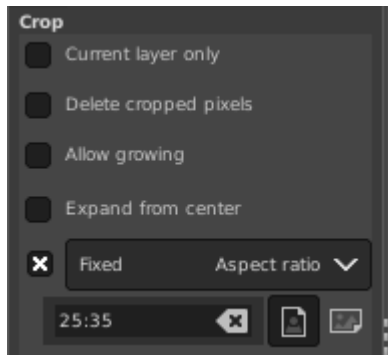
$$6 \times 300 = 1800 \text{ pix}$$

No unit conversion was required since the lengths were measured in inches here.

We want +2 pix spacing on the left/right sides and +2 pix spacing between the rows (at the bottom of each row to use the gaps as white markings for the paper cutter machine).

The dimension of each Photo ID is 104.25x145.95 millimetres (at 72 DPI). But in our case, it is not needed. We won't even have to calculate the pixel dimension at 300 DPI resolution, which is 295.3x413.4 pixels. The dimension of the photo in its print size in inches at 300 DPI is all we need. You may ask me the reason behind going through the troubles of calculating all sorts of dimensions at various resolutions in the earlier steps. Yes, we need it somehow. Montage cannot crop/rotate/centre our photo. We will have to use an Image Editing Application to do that. Knowing the dimensions will give us an overview of the final arrangement.

1. Open GIMP.
2. Press **SHIFT + C** (Crop).
3. In the Crop Option, click the checkbox **"Fixed"**. Select **"Aspect Ratio"** from the drop-down menu.



4. Set the ratio of the dimension of the Photo ID. In our case, 25x35 mm means 25:35.
5. Crop the photo and write it to disk with an appropriate name like DSC\_85XYZ\_25x35.JPG
6. Verify the dimension of the image by pressing **ALT + ENTER**. The dimension should be close to our calculated value without the fraction part.
7. From the Menu Bar, go to **Image -> Print Size...** The print dimension should be nearly equal to 25x35 mm.

We don't have to verify the cropped image every time after performing the crop operation. However, knowing the dimensions of the photos will be of immense help while automating the Photo ID Montage creation task through scripts.

Make sure you crop the image properly.

A 4"x6" photo paper can provide room for 12 Stamp Size photos at the most, 3 columns and 4 rows in order. How will you know that? Fill a 4"x6" photo with Stamp-Sized photos manually in GIMP. This will give you an idea of the final arrangement of the photos. Then you can write a script to automate the task of creating a Photo ID montage.

Calculate the total width that will be occupied by 3 photos placed side by side.  $(25 \times 3) \text{ mm} = 75 \text{ mm}$ .

And, the total height for 4 rows is  $(35 \times 4) \text{ mm} = 140 \text{ mm}$ .

In Inch, the dimension is:  $\frac{75}{25.4} = 2.952755906 \simeq 2.95 \text{ inch}$  by  $\frac{140}{25.4} = 5.511811024 \simeq 5.51 \text{ inch}$ , i.e., 2.95X5.51 inch.

Derive the dimension of the output in pixels at 300 DPI.

$$\text{printWidth}[\text{Inches}] \times \text{printResolution}[\text{pixPerInch}] = \text{width}[\text{pixels}]$$

$$\text{printHeight}[\text{Inches}] \times \text{printResolution}[\text{pixPerInch}] = \text{height}[\text{pixels}]$$

$$2.95 \times 300 = 885 \text{ pix}$$



$$5.51 \times 300 = 1653 \text{ pix}$$

However, some additional space will be occupied by the gaps.

$$(3 + 1) \times 2 = 8 \text{ (1 extra space for adjustments [the first +2 pixels gap])}$$

$$(4 + 1) \times 2 = 10 \text{ (1 extra space for adjustments [the first +2 pixels gap])}$$

So the calculated dimension of our output montage is:

$$(885 + 8) \times (1653 + 10) = 893 \times 1663 \text{ pixels}$$

```
montage \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
IMG_20211008_132718.jpg \  
-tile 3x4 \  
-geometry '+2+2' \  
-resize 893x1663 \  
4_6_out.jpg
```

Write a script: `stamp-pp-on-4x6.sh`

```
#!/bin/bash
```

```
# Create a montage of one photo on a 4x6-inch canvas
montage \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
IMG_20211008_132718.jpg \
-tile 3x4 \
-geometry '+2+2' \
-resize 893x1663 \
4_6_out.jpg
```

Change the filename parameters. Your photo will be of a different name.

Run the script as:

```
chmod +x stamp-pp-on-4x6.sh
```

```
./stamp-pp-on-4x6.sh
```

A better version of the script will be:

```
#!/bin/bash

# Create a montage of one photo on a 4x6-inch canvas

image_file="$1"
output_file="$2"

montage \
"${image_file}" \
"${image_file}" \
"${image_file}" \
"${image_file}" \
"${image_file}" \
"${image_file}" \
"${image_file}" \
"${image_file}" \
"${image_file}" \
"${image_file}" \
"${image_file}" \
"${image_file}" \
-tile 3x4 \
-geometry '+2+2' \
-resize 893x1663 \
"${output_file}"
```

This script accepts the image file name as the first argument and a name of your preference for the output as the second argument. The first argument will then be repeated 12 times and be passed to the `montage` command.

Call the script with the following command (example):

```
bash stamp-pp-on-4x6.sh IMG_20231020_010203.jpg my_4x6_montage.jpg
```

Or simply,

```
./stamp-pp-on-4x6.sh IMG_20231020_010203.jpg my_4x6_montage.jpg
```

Press **TAB** to autocomplete names.

The dimension of our output montage is 893x1663 pixels. The size of a 4"x6" photo is 1200x1800 pixels. That means, we cannot directly print an 893x1663 pixels image onto a 4"x6" photo paper.

1. Open GIMP.
2. Create a new 4"x6" canvas.
3. Drag the 893x1663 pix image from your file manager onto the 4"x6" canvas.
4. Press **CTRL+M** to merge the layers.
5. Export the image to a JPEG file. Give the output file a name of your choice.

Now you are ready to print the Photo ID montage on a 210 GSM 4"x6" photo paper.

---

Another approach:

Write a C program and run it as a script instead of a bash script.

Refer to: <https://github.com/ryanmjacobs/c>

```

#!/usr/bin/c -Wall -Wextra -pedantic --

/*
Refer to: https://github.com/ryanmjacobs/c
*/

/*
c: Use C as a shell scripting language:
https://github.com/ryanmjacobs/c
*/

/*
Installation (*-ubuntu)/Debian:
sudo apt install build-essential
sudo apt install trash-cli
cd ~/
wget https://raw.githubusercontent.com/ryanmjacobs/c/master/c
sudo install -m 755 c /usr/bin/c
trash c
*/

/*
Usage:
chmod +x stamp_pp.c
./stamp_pp.c <input_image> <output_image>
*/

// or, gcc -Wall -Wextra -pedantic -o2 stamp_pp.c -o stamp_pp -s

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXBUFFSIZE 8092
#define MAX_IMG_REPEAT_LENGTH 7056
#define TEMP_CHAR_S_LENGTH 1024
#define REPEAT_COUNT_FOR_STAMP_25X35_MM_ON_4X6 12
#define RESIZE_OPTION_FOR_STAMP_25X35_MM_ON_4X6 "-resize 893x1663"

int main(int argc, char *argv[]) {
    if(argc != 3) {
        fprintf(stderr, "Usage: %s <input_image> <output_image>\n", argv[0]);
        return 1;
    }

    char *input_image = argv[1];
    char *output_image = argv[2];

```

```

char command[MAXBUFFSIZE];
char img_name_repeated[MAX_IMG_REPEAT_LENGTH];
char temp[TEMP_CHAR_S_LENGTH];

for(int i = 0; i < REPEAT_COUNT_FOR_STAMP_25X35_MM_ON_4X6; i++) {
    snprintf(temp, sizeof(temp), "\"%s\" ", input_image);
    strncat(command, temp, (MAXBUFFSIZE - strlen(temp) - 1)); //
https://stackoverflow.com/questions/6903997/how-can-i-use-strncat-without-
buffer-overflow-concerns
}

/*printf("command: %s\n", command);*/
strcat(img_name_repeated, command);
/*printf("img_name_repeated: %s\n", img_name_repeated);*/
sprintf(command, "montage %s -tile 3x4 -geometry '+2+2' %s \"%s\"",
img_name_repeated, RESIZE_OPTION_FOR_STAMP_25X35_MM_ON_4X6, output_image);
printf("command: %s\n", command);
system(command);
return 0;
}

```