Introduction to Population Receptive Field models

Sam Schwarzkopf

Before the start using SamSrf for population receptive field (pRF) mapping, here is a basic introduction to how our method works. Our main procedure is based on Dumoulin & Wandell, 2008, NeuroImage, 39(2):647-60. Other analyses are also possible but are only mentioned in passing in this cookbook. Please refer to the ModelHelp tool for more details.

What is a pRF?

As any textbook on neurophysiology will tell you, a neuron's receptive field is the region of space that can excite the neuron when we put a stimulus in it. "Space" can mean a lot of things in this context. Most frequently, people will mean *visual space*, that is, the retinotopic location. However, there are other stimulus spaces, such as maps of the frequency of a tone or locations on your skin. In fact, a visual receptive field is just a tuning function for position in visual space. So in essence what we want to do is map tuning functions for a particular stimulus space – and this could be anything, be it visual position, sound frequency, selectivity to particular objects, your location in the environment, or even things like language. So, don't let anyone tell you that pRF models are just a fancy form of retinotopic mapping. There is potential for so much more.

Naturally, we are not measuring single neurons but the activity of a large number of neurons combined (and, in fMRI, through an indirect, metabolic proxy measure of neuronal activity). This is why we call it a *population* receptive field. The properties of the pRF are not necessarily an up-scaled version of the receptive fields of single neurons. The pRF will contain those parameters but is also dependent on the variability in the position of the individual receptive fields, how much visual space the receptive fields of all the neurons cover, and it also incorporates the extra-classical or contextual interactions between neurons within (and outside) the population.

Last but not least, the acronym is spelled with a lower-case *p* for some arcane reasons. This is probably in reference to the fact that in the neurophysiology world there are such things as *classical* receptive fields that are abbreviated as cRF. For all I care you can spell it PRF.

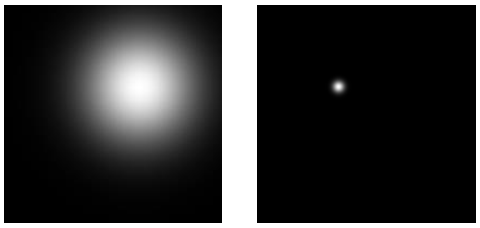
How do we map pRFs?

Simply put, for each population of neurons in the brain we want to estimate the parameters of the pRF from the fMRI (or other?) data we collected. The parameters of the pRF can be many things. In a simple case it would be the position (in Cartesian coordinates, x_0 and y_0) and the spatial spread of the pRF (since we use a 2D-Gaussian model, this is called σ, i.e. *sigma*). But more complex receptive field models are also possible, such as antagonistic centre-surround profiles (modelled as a difference-of-Gaussians function), or oriented and/or asymmetric profiles. Putting together all the parameters for neighbouring points on the cortical sheet gives us a map for this parameter.

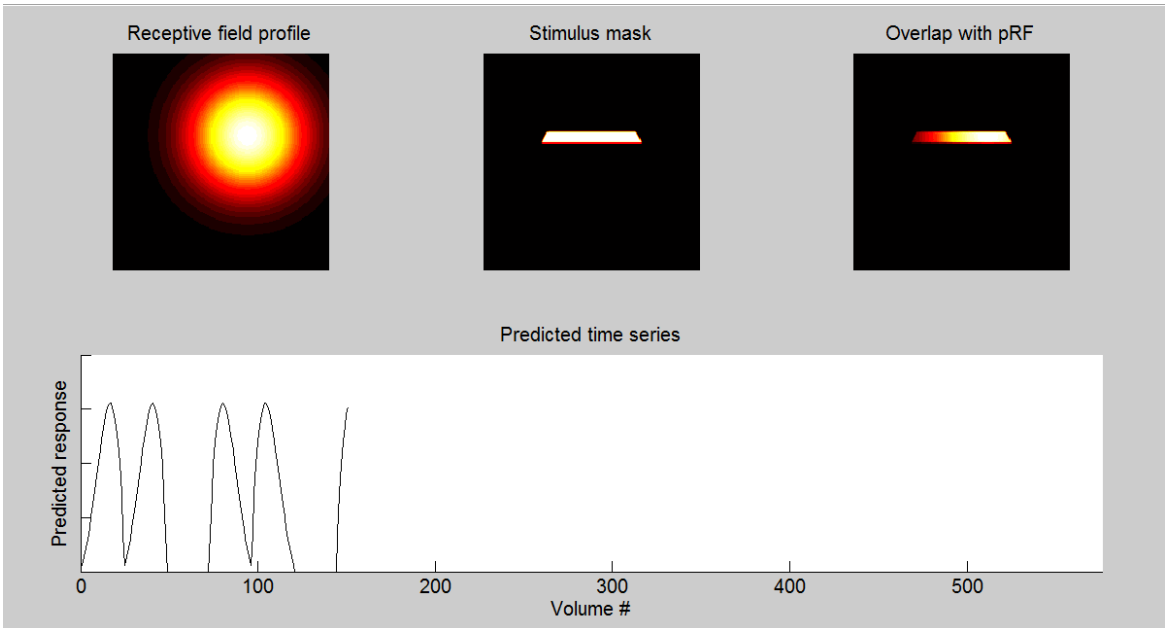
In order to estimate these parameters, we need to have two sets of information: the observed data (i.e. the fMRI time series) and the time series of the stimulus. In SamSrf the latter is called the **stimulus apertures**, but that's just a name for representation of the stimulus we presented to participants during the scan. In fact, it is usually a simplified version of the stimulus that just encodes whether a stimulus was present at each location in the visual field at a given time. In

earlier SamSrf versions (and to our knowledge in other toolboxes), the aperture movie is used directly as input to pRF models. However, as of SamSrf 9 we need to use **vectorised apertures**. This means that we define a set of stimulus locations (such as the pixels in the aperture movie), each of which is assigned a coordinate in visual space. The aperture matrix contains locations in rows, and time points (fMRI volumes) in columns. The reason for doing it this way is that it makes the design of apertures much more flexible (you can use non-symmetric apertures or even just a line of pixels for one-dimensional tuning models). Coordinates can also be defined directly in stimulus space (e.g. degrees of visual angle) rather the image/aperture space. Most importantly though, this approach speeds up the analysis considerably – in our hands usually by a factor of 3-5!

Armed with pRF model and the stimulus apertures we can now fit the pRF model. What we need to do is find the pRF parameters (like position and size) that can best explain the observed time series. We can predict the time series a particular receptive field profile would produce if faced with the apertures of our stimulus sequence. At each moment in time (i.e. each TR of our fMRI scan) we calculate the overlap between the stimulus aperture and the receptive field profile. As already described, the apertures are just a representation of the stimulus in space and time. The receptive field profile is simply the predicted response for a given location. We multiply this prediction, pixel-by-pixel, with each time point of the aperture. Subsequently we calculate the mean of this pixel-wise overlap to estimate the neural response. Since SamSrf 7, this takes into account the pRF size but in previous versions it simply calculated the overlap across the whole aperture frame. To our knowledge, other pRF tools do the latter but arguably the former is more biologically correct – practically this isn't a big issue though, because it only affects the beta amplitude but doesn't change your pRF shape. If required, changing this is also very straightforward.



As a toy example, consider two pRFs: You see one is in the upper right visual field (coordinates 0.5, 0.5) and very large, while the other is in the upper left visual field (-0.5, 0.5) and very small. Next you we need the stimulus apertures. You will have one if you run a pRF mapping experiment but if you don't, ask one of the people who have done the analysis already. This allows us to predict a neural response which will look something like this:



In the top row you see the receptive field profile on the left, the stimulus aperture at the current time point in the middle, and the overlap between the two on the right. In the bottom row we see the predicted response of this hypothetical receptive field. Of course, because we are measuring BOLD signals with fMRI our time series does not reflect the underlying neural activity directly. Therefore, we must convolve the predicted time series with the haemodynamic response function (HRF). For this we can either take a canonical HRF or measure the HRF specifically for each subject.

Theoretically, using a subject-specific HRF is probably better because it takes the individual differences in the HRF into account. In practice, this does not seem to make much of a difference though as Dumoulin & Wandell showed in their seminal 2008 study. Naturally, any individual HRF is only as good as the data that you used to model it. Using a bad HRF can completely screw up your pRF model. Another, more efficient approach is to first fit a pRF based on a canonical HRF, then use the predicted time series as predictor for fitting the individual HRF, and then refitting the pRF model with this individual HRF. An advantage of this approach is that it uses all the pRF mapping data you acquired to fit the HRF rather than spending extra time on measuring the HRF. This approach can also improve the goodness of fit of your pRF models – but this is also somewhat circular. Whether or not this circularity is a problem we don't know, and Sam doesn't have time to find out (if you're interested in doing that study, get in touch and we are happy to consult…). Either way, in many scenarios the canonical HRF is completely acceptable, and most people probably do not need to worry about this.

Fitting pRF models

The tricky thing we must do now is to find the receptive field profile (or more precisely its associated parameters) that can produce a time series that matches the observed time series. What we therefore need to do is fit a model with several free parameters (in this case at least three) to the observed data. Ideally, we want to do this straight out of the box but this is tedious and difficult. So, what we can do is apply a coarse-to-fine approach in which we first find a way to disambiguate the parameters and then refine them further.

The way this is done for most models in SamSrf is to run an extensive grid search, the *coarse fit*, where we first generate a large search space with plausible combinations of the free parameters. For each point in this multidimensional search space we generate a predicted time series and correlate this with the actually observed data. The search parameters that produced the best correlation are then used in the *fine fit* where we run an optimization procedure to further hone the fitted parameters to achieve the (hopefully) optimal fit. The good thing about the coarse fit is that we can do this simultaneously for many vertices at the same time, which makes the procedure quite fast. (Unfortunately, this appears to only work in MATLAB R2011b and higher. On older versions you must do it at a snails-pace one vertex at a time which takes ages…). As of SamSrf 8, several alternatives for the standard fine fitting are available (see next section for details).

The granularity and dimensions of this search space for the coarse fit you can define when you specify your pRF model. The more predictions this coarse fit contains, the longer it will take to generate predictions and how much disc space and memory the search space requires. There is probably a sweet spot between having a well-sampled search space and unnecessarily wasting too much time and memory on the coarse fit. Obviously, a pRF model with more free parameters also requires more resources and will put limits on how accurately you can sample the search space. **Importantly, you only really need to generate this search space once, provided you collect the**

same amount of data in the same sequence for each subject. If the stimulation paradigm differed between subjects, then unfortunately you also need individual apertures and thus an individual search space.

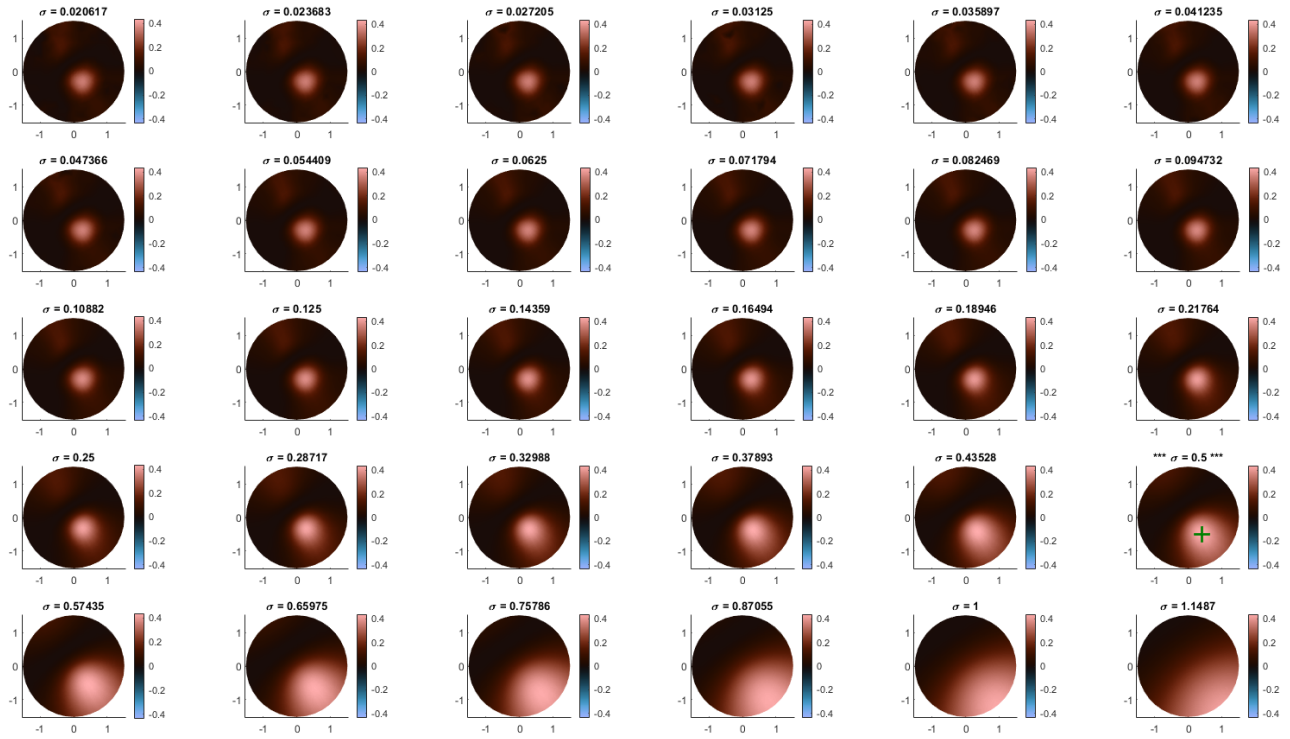
For both the coarse and fine fit, the best fitting model is determined by the Pearson correlation between the observed and predicted time series. It is quantified by the coefficient of determination (the "goodness of fit"), which is the square of the correlation coefficient. Once the parameter fitting is complete, SamSrf subsequently runs a regression analysis to fit a *beta* parameter to estimate the response amplitude, and a baseline (intercept) parameter to estimate the baseline. Since we typically z-standardize the observed time series, the baseline will typically be close to zero but it can vary. More importantly, the beta can be positive or negative. Negative betas are often a sign of some artifactual data, such as regions outside the mapping stimulus or blood vessel artifacts.

Algorithms for fine fitting

By default, the fine fit uses an optimisation procedure known as the Nelder-Mead simplex search algorithm, which is implemented in MATLAB's Optimization Toolbox under the name *fminsearch*. This creates a little polytope around the seed parameters with k+1 vertices, where k is the number of free parameters (so for a 1D tuning curve with peak and tuning width as parameters, this would be a triangle). It then moves around and shrinks this polytope as required to home in on the local minimum. You can read more about this here:

https://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method

This algorithm is very effective, but it can be slow and it also has some problems. It may also be overkill for most pRF applications, at least standard retinotopic mapping purposes. SamSrf includes a function called samsrf_gsr2 which allows you to visualise the correlations for each grid position in the search space. Here is an example for a standard 2D Gaussian pRF. Each plot shows the correlations for the x0 and y0 positions in the search space for a given pRF size (Sigma):



What this shows is that there is a clearly defined cluster of high correlations (pink blob in the bottom right quadrant). The peak correlation is pretty consistent between the various pRF sizes. Only for very large pRFs the cluster becomes blurry and smeared peripherally. The maximum correlation across the whole search space is indicated by the green cross (rightmost column, second to bottom row). This shows that there is actually a very smooth gradient in terms of best fits and very little ambiguity about the parameters. Algorithms like the Nelder-Mead method are really designed to deal with much more complex problems. So for many purposes we can optimise the parameters using much simpler algorithms, which are available in SamSrf:

1. *Slow Coarse Fit:* Instead of only taking the best correlating prediction from the search space it averages the parameters of a top percentile you can define (*Model.Coarse_Fit_Percentile*). This can yield pretty good pRF estimates while being considerably faster than the fine fit. But it is slower than the standard coarse fit. There is probably an optimum for what search space granularity is required to obtain sufficiently precise estimates. Simulations can help here. Also keep in mind that this method is necessarily constrained by the limits of your search space. If you expect pRFs to fall outside the stimulus range your search space should extend considerably beyond that and you may need to include x0,y0 = 0 (a typical polar search grid would not contain that).

2. *Adjustable parameter tolerance:* You can also adjust the parameter tolerance of the Nelder-Mead algorithm. If the tolerance is higher than default (1e-4) this trades accuracy for speed. This was the very problem we ran into with SamSrf 6 because its tolerance was too lenient (1e-2) for our purposes. But for many applications this may be completely acceptable & you could also try intermediate tolerance levels. Simulations could help here, too.

3. *Hooke-Jeeves algorithm:* You can also use Hooke-Jeeves pattern-search as an alternative to the Nelder-Mead algorithm. This algorithm works by stepping away from the seed parameters along each dimension. If a better fit is found, the current parameter is moved there. If not, the step size is shrunk by half. See here for more details:

https://en.wikipedia.org/wiki/Pattern_search_(optimization)

This algorithm is comparably fast but not as precise and robust as the Nelder-Mead method. But as we have seen, for a lot of pRF data a simple approach may suffice. For speed, SamSrf only runs a small number of iterations for this fit. (Note: this only applies to pRF forward-modelling; when fitting reverse correlation pRFs/CFs the algorithm uses the default number of iterations). This method seems to perform well for standard 2D pRF models (especially compared to a lenient Nelder-Mead tolerance) but that will probably depend a lot on your situation. Again, this might be worth running some simulations.

Parallel processing

SamSrf relies strongly on parallel computing. This means that if you have the Parallel Computing Toolbox in MATLAB installed it will take advantage of multi-core computers and can in theory also be used on high-performance clusters. This will speed up several computationally expensive procedures, such as smoothing, back-projection, and of course the slow fine-fitting algorithm. SamSrf may be very slow without it. In SamSrf 8, we removed support for non-parallel processing completely, but if you want to use it without it you can in theory edit the functions accordingly.

Analyses available in SamSrf

Below follows a list of pRF methods currently available in SamSrf. Please also refer to the ModelHelp tool for more information about specifying models and running these analyses:

1. *Simple Gaussian*: This is the most basic pRF model that is most widely used in the literature. It models the pRF as a two-dimensional Gaussian. It is defined by its position (x_0 and y_0) and its size (sigma).
2. *Difference-of-Gaussians*: This procedure models centre-surround antagonistic receptive fields. In addition to position (x_0 and y_0), we have two sigmas (one for the centre, one for the surround) and a parameter for the ratio of the amplitudes (delta) of the two components. The second surrounding Gaussian is subtracted from the first, the centre Gaussian. This function produces a Mexican hat shape. **Importantly, you cannot directly translate the centre sigma into a classical sigma from the standard simple Gaussian pRF. Rather, you need to quantify the full-width at half-maximum of this function to determine the pRF size.**
3. *Multivariate Gaussian*: This is an oriented elliptical Gaussian so this has two pRF size parameters (sigma1 and sigma2, for the horizontal and vertical axis, respectively) and an orientation parameter (phi). **There is some ambiguity in this model because the model is rotationally symmetric. You need to account for this in your post-modelling analysis.**
4. *Gaussian Curve*: This procedure is very similar to the standard pRF approach except that it is fitting a one-dimensional tuning curve rather than a two-dimensional receptive field. The apertures in this case should contain bars traversing the stimulus space in only one direction. This direction can be either horizontal or vertical. The tuning curve is modelled as a Gaussian pRF that is fixed in the orthogonal dimension (so if the stimulus moves horizontally, the pRF's vertical position is fixed at 0. **The assumption here is that the stimulus space is linear. So if your stimulus dimension is non-linear you must log-transform it first** (e.g. in tonotopy or spatial frequency).
5. *Polar Angle Curve*: This is the same as the Gaussian Curve model except that this assumes a circular stimulus space (e.g. orientation and direction) and therefore fixes the position of the Gaussian pRF to be at a constant distance from the origin.
6. *Reverse Correlation*: This procedure uses a model-free approach to estimate the pRF profile. It convolves the stimulus apertures with the HRF, then regresses the activity measured in each fMRI volume against these apertures, so that for each pixel in the aperture you see how much the stimulus predicts the measured response. This gives an image that reflects at which location the voxel showed the strongest responses. Because our stimuli are usually temporally ordered and predictable designs there will be a lot of correlation throughout the image (e.g. with a bar stimulus you will see a set of bars intersecting at the pRF location). Some filtering may therefore be required to minimise these artifacts whilst retaining the (presumably) true pRF. These pRF profiles are then saved and the x and y coordinates of the pRF peak and its spread are estimated from that. Up to this point the procedure is comparably fast. However, you can also fit 2D pRF models to those reverse correlation profiles. This requires you to define some additional parameters in the Model. You can also transform pRF profiles (rotating them into a common axis and/or centring them on their peak) and average them for further analysis. This procedure has many advantages over model-based pRF fits but also substantial disadvantages. So the choice between forward-model and reverse-correlation methods really depends on the purpose of your research.

7. *Connective Field Reverse Correlation*: This method is vaguely based on the ideas first presented by Haak et al. 2013 in *NeuroImage*. It estimates the vertex-wise functional connectivity (correlation between time courses) between all vertices in your region of interest, and each vertex in a pre-defined *seed region*. In a retinotopic paradigm, this seed region could for example be V1. This produces a correlation profile across the whole seed region. The connective field is the region where you observe the strongest correlation. We then use a template map of the seed region to assign a position to each connective field. This template could be the probabilistic retinotopic map predictions described by Benson et al. 2012 in *Current Biology* or some anatomical definition. Further, we can then fit a 2D pRF model to the correlation profile projected back into stimulus space to estimate pRFs.
8. *Connective Field Forward-Model:* Like for pRF models, you can fit a forward model for connective fields. This is similar to the method first described by Haak et al. 2013. This has some advantages over using reverse correlations, but it also comes with other caveats and depends on several assumptions. It also contains a few options to simplify the analysis, such as analysing only polar angle wedges or eccentricity bands.

Surface projection

One last thing you should know about how the analysis in SamSrf works is surface projection. FreeSurfer's automatic cortical reconstruction creates a 3D model of the boundary between grey and white matter. This is a mesh consisting of over 100,000 little points (vertices) that are connected to one another in an even greater number of tiny little triangles (faces). FreeSurfer also expands this surface until it hits the pial boundary between grey matter and cerebrospinal fluid. Thus, for each vertex on the grey-white mesh there is a corresponding vertex in the pial mesh. We can use this to find points that lie within the grey matter and locate which voxels those correspond to in our functional MRI scans. This is what we do when we project the fMRI data onto the cortical surface. All analyses in SamSrf take place on the surface (but functions for running volumetric analyses are available if needed. These have never been tested by us however, and you will lose a lot of functionality).

There are different options for conducting the surface projection. You can either do that in SamSrf using the included functions, or you can do it directly in FreeSurfer. The latter approach is faster and you can include volumetric or surface-based smoothing if desired and use a number of other features. We recommend using FreeSurfer because of its speed – however, we probably cannot answer any questions about this approach, since we didn't write that code.

FreeSurfer further inflates the cortical meshes to smooth out the cortical folding structure. This is helpful for displaying the data on the cortical sheet and delineating maps. Finally, it also expands these inflated meshes to a spherical mesh. This can be used to combine data from different individuals. It is also extremely useful for surface-based (geodesic) operations that we might want to do later (e.g. smoothing, field sign calculation, cortical magnification, etc.).