EFFECTS OF REPETITIVE RETINOTOPIC STIMULUS ON VISUAL CORTEX fMRI SIGNAL

by

Andaç Hamamcı

B.S., Physics, Bogazici University, 2002

Submitted to the Institute of Biomedical Engineering
in partial fulfillment of the requirements
for the degree of
Master of Science
in
Biomedical Engineering

Boğaziçi University
April 2006

ACKNOWLEDGMENTS

I would like to thank to Assoc. Prof. Dr. Cengizhan Öztürk, for his supervising, support, encouragement and motivation. I am thankful to Prof. Dr. İlhami Kovanlıkaya for his support, patience and discusing the topic in a medical perspective.

I am greatly thankful to Uzay Emrah Emir for his introduction to the topic, great support and motivation during my thesis progress. It would not be possible to complete this work without his great effort.

I would like to thank to volunteers -my friends- for their patience and motivation. I very much appreciate Umut Sarı for his efforts during writing the thesis.

Finally, I am deeply greatful to my mother, father and Dilek for their recomendations on thesis outline, encouragement and love.

ABSTRACT

EFFECTS OF REPETITIVE RETINOTOPIC STIMULUS ON VISUAL CORTEX fMRI SIGNAL

The aim of this study is to investigate the effects of the repetitive stimulus on the retinotopic functional magnetic resonance imaging results obtained. Screen-projector system is constructed to show visual stimulus to subjects during fMRI scans. Retinotopic stimulus are applied to 5 healty subjects, everyday during 3 weeks. fMRI scans of subjects are done one at the begining and once after each week. The data acquired is analysed to obtain the retinotopic fMRI parameters. Visual areas, BOLD signal changes and cortical magnification in primary visual cortex are determined for each week. The results obtained are quite stable. Observed changes are discussed which might be investigated in further studies.

Keywords: Visual Cortex, fMRI, retinotopy, adaptation.

ÖZET

TEKRARLANAN RETİNOTOPİK UYARTILARIN GÖRSEL KORTEKS FONKSİYONEL MANYETİK REZONANS SİNYALİNE ETKİLERİ

Bu çalışmanın amacı, tekrarlanan retinotopik uyartıların, retinotopik fonksiyonel manyetik rezonans görüntüleme sonuçlarına etkilerinin araştırılmasıdır. fMRG çekimleri sırasında deneklere görsel uyartıları göstermek amacıyla ekran-projektör düzeneği kuruldu. Retinotopik uyartılar 3 hafta boyunca hergün 5 saglıklı deneğe uygulandı. fMRG çekimleri başlangıçta ve her haftanın sonunda birer kez alındı. Elde edilen veriler analiz edilerek retinotopik fMRG parametreleri elde edildi. Her hafta için görsel alanlar, BOLD sinyal değişimleri ve birincil görsel korteks kortikal magnifikasyonları belirlendi. Elde edilen sonuçlar oldukça kararlıydı. İleriki çalışmalarda araştırılabilecek, gözlemlenen değişimler tartışıldı.

Anahtar Sözcükler: Görsel Korteks, fMRG, Retinotopi, Adaptasyon.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

∇r	Gradient of r
$\sum$	Summation
T	Tesla
s	Second
Hz	Hertz
mm	Millimeter

LIST OF ABBREVIATIONS

BOLD	Blood Oxygenation Level Dependent
CSF	Cerebrospinal Fluid
fMRI	Functional Magnetic Resonance Imaging
GE-EPI	Gradient Echo-Echo Planar Imaging
GRE	Gradient Echo
LH	Left Hemisphere
MRI	Magnetic Resonance Imaging
MT	Middle Temporal Area
PC	Personal Computer
RF	Radio Frequency
RH	Right Hemisphere
SNR	Signal to Noise Ratio
T1	Spin-Lattice Relaxation Time
T2	Spin-Spin Relaxation Time
V1	Primary Visual Cortex
V2	Secondary Visual Cortex
VA	Ventroanterior Area
VFR	Visual Field Ratio
VFS	Visual Field Sign

1. INTRODUCTION AND MOTIVATION

Areas responsible of visual processing are located at the visual cortex which is at the occipital lobe of the brain. Visual cortex is divided into several areas which have distinct functionalities. Using the anatomical information is not sufficient to determine this areas because of the high variability. These areas can be determined by using the distinctive functional properties. One of these distinctive properties is their retinotopic organization. That is, the neighbourhood relations of the visual field are preserved for the neurons of the visual cortex.

Retinotopic organization of the visual cortex has been investigated by invasive experiments on animals since a long time. However, most of these methods can not be applied while investigating the normal operation of intact human brain. With the advent of the functional magnetic resonance imaging, several areas of the visual cortex have been defined and characterized using the retinotopic organization. The results obtained by using retinotopic fMRI are consistent with that obtained by electrophysiological studies on primates [1, 2].

One of the most important application areas of retinotopic functional imaging is to study the plastic changes of the visual cortex in response to peripheral manipulations or experience. Determining the ability of cortical plasticity is important mainly because of two reasons: (1) It is related to learning, (2) it indicates the recovery rate of a misfunctionality caused by a cortical damage. In these studies, the results of retinotopic fMRI done frequently on the same subject are compared. However, as far as we know, the effects of the retinotopic stimulus itself on the measured parameters has not been studied. The effects of the repetitive retinotopic stimulus is investigated in this thesis.

There are six chapters following this in the thesis. In the next background chapter, information and a literature survey about visual cortex organization, plasticity of visual cortex, functional adaptation due to repetition and determination of visual

areas are given. Experimental set-up is described briefly in chapter 3. In chapter 4, analysis methods used to construct the cortical surfaces and retinotopic parameters are given in detail. While, analyse method is not standard in the literature, alternative ways to analyse retinotopic fMRI data are discussed.Results obtained are presented in chapter 5, where visual field sign maps and map of bold response changes are tabulated. Bold signal changes in primary visual cortex (V1) are given both in plots and tables. The cortical magnification with respect to eccentricity angle is plotted. The results are discussed in chapter 6. In chapter 7, further work to improve the retinotopy procedure is discussed.

2. BACKGROUND

2.1 Organization of the Visual Cortex

Areas of visual cortex have been defined using a variety of criteria, including anatomical connections, the stimulus selectivity of single-unit responses, architectural properties, and retinotopic organization [3]. Human primary visual cortex area (V1) represents an entire hemifield in calcarine cortex of each hemisphere which is located at the posterior pole of the brain in the occipital lobe. The dorsal and ventral boundaries of V1 are separated by roughly 4 cm and fall on the cortical surface at the upper and lower lips of the calcarine. Studies of patients with localized cortical damage showed that the receptive fields of neurons within area V1 are retinotopically organized (Figure 2.1). From posterior to anterior cortex, the visual field representation shifts from the center (fovea) to the periphery. This dimension of retinotopic organization is commonly referred to as eccentricity. A second dimension of retinotopic organization, the angular dimension, is represented on a path that traverses from the lower to the upper lip of the calcarine sulcus. Along this path, the visual field representation shifts from the upper vertical meridian through the horizontal meridian to the lower vertical meridian [3]. Visual cortex areas can be better visualised on flattened representation. Area V1 is bordered by two cortical regions, each roughly 1 cm, that form area V2. Dorsal V2 (V2d) represents the lower quarter of the visual field and ventral V2 (V2v) represents the upper visual field. The strip V2v (V2d) is bounded by another distinct cortical strip called V3v (V3d). V3v is sometimes refered as VP [2]. Like their neighbours, V3d and V3v each represent one quarter of the visual field, and each spans roughly 1 cm of width and 6-8 cm of length [3]. In macaque monkey, dorsal and ventral field representations of V3 have some functional differences. Altough it is not clear that these seperate quarter field representations are distinct, general approach is to maintain separate labels for each of the quarter field representations: V2d, V3d, V2v, and V3v [3, 2]. An additional retinotopically organized area, V3A, is adjacent to V3d. And another retinotopically organized ventral region is labeled as V4v which is also named as VA (ventroanterior

area) [2]. If the whole map of human visual cortical areas is flattened as a sheet around the occipital pole, another visual area MT (middle temporal area) which is sometimes refered as V5 in literature, is located deeper into the medial bank [4].

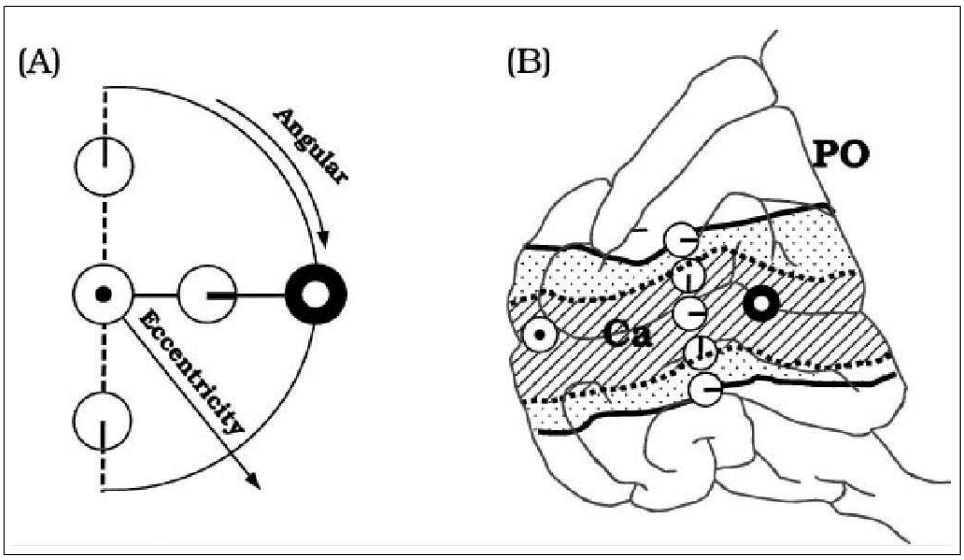

Figure 2.1 Representation of visual field (A) on visual cortex (B) [3]

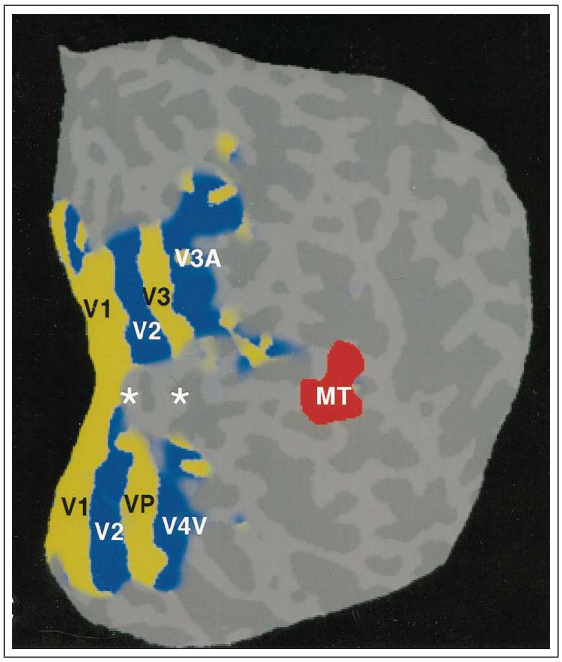

Figure 2.2 Flattened representation of visual cortex areas [5]

2.2 Plasticity of the Sensory Maps

Topographic sensory maps are not static in adults, but in fact undergo plastic changes in response to both peripheral manipulations and experience throughout life. High plasticity for children is well known: (1) Removed hemisphere functions can be done by other side, (2) children can learn and forget a known language but adults can not adapt that much [6].

Plasticity is the modifiability of receptive field properties. Receptive field for visual system is initially defined as "the region of retina which must be illuminated to obtain a response in any given fiber" [7]. Determining the ability of cortical plasticity is important mainly because of two reasons: (1) It is related to learning. (2) It indicates the recovery rate of a misfunctionality caused by a cortical damage.

The field of cortical synaptic plasticity and cortical map plasticity have been implicitly linked by the hypothesis that synaptic plasticity underlies cortical map reorganization [8]. The imaging studies on adult cortex show that altough the large scale branching patterns of dendrites and axons were stable, 50 % of the spines, which show experience-dependent structural plasticity during development, changes in one month [9]. Most of the studies investigating the cortical map plasticity are based on reconstructing changes in distributed responses following enduring changes of inputs. This can be induced by peripheral denervation, by central lesions or by repetitive training. It has been shown in various studies that removal of the normal input to part of the adult primary visual cortex results in map reorganization. Kaas et al., lesioned a 5 degree-10 degree area o one retina. Weeks after, the area responsive to the lesioned area acquired new receptive fields corresponding to area surrounding retina around the lession [10]. In another study by Chino et. al., 2 months after producing a focal lesion in one retina, V1 topography was relatively unaltered, however after a few hours enucleation of the unlesioned eye, the silenced area developed responses to retinal locations surrounding the focal lesion [11]. This is an important result showing that both retinas has to be lesioned to induce plasicity. Schmid et al reported altered topographical maps in contrast to Chino. Novel responses were comperatively weaker and underwent

rapid habituation [12]. Altough the microelectrode studies done on adults have shown that plasticity in adults is high, more recent studies that uses fMRI show that the plasticity of adult cortex is very little during 7.5 months after a binocular lesion [13]. How higher areas react to a hole in V1 is also an interesting topic because visual experience depends on those areas too [6].

Several characteristics of perceptual learning suggest the involvement of early stages in sensory processing, perhaps as early a stage as primary sensory cortex [8]. The result of the study of Crist et.al. by using monkies trained with two-line bisection task showed that the cortical magnification factor and receptive field sizes were remained constant while the response rate was slightly reduced [14]. Sugita et al. reported V1 neurons in monkeys can develop novel receptive fields to the ipsilateral hemifield after monkies have worn reversing spectacles for several months [15]. Similarly, training dependent expansion of the cortical representation was shown for somatosensory [16] and auditory systems on monkies [17].

Most of these studies show the potential of modifiability of cortical properties due to the repetition of a specific stimulus. It is important to have a prior information on how much the measured parameters are changed due to repetition in retinotopic fMRI experiments while using this tool in plasticity research area, which is studied in this thesis.

2.3 Adaptation

Neural activity is usually reduced due to repetitive stimulation [18]. This effect has been reported at various spatial scales (individual cortical neurons to hemodynamic changes) and location, and various temporal scales (longevity from milliseconds to days). Stimulus-specific reduction in neural activity can be named as adaptation, mnemonic filtering, repetition suppression, decremental responses or neural priming. It is an important research area mainly because of two reasons: (1) For inferring the nature of representations across different stages of a processing stream. (2) Might be

the neural correlate of priming, that is, the improved processing of a repeated stimulus according to some behavioral measure (e.g. greater accuracy in identifying the stimulus, or faster response to make a decision about it).

The neural mechanisms underlying the adaptation might be firing-rate adaptation, synaptic depression, long term depression and long-term potentiation. There are multiple potential neural causes of the adaptation like the decrease in the firing amplitude (fatigue model) [19, 20], decrease in the number of neurons that respond (sharpening model) [21, 22, 23] or shortening of the neural activity duration and/or latency (facilitation model) [24, 25].

One of the methods that is widely used by researchers to characterize the functional properties of neural populations at subvoxel resolutions using MRI is fMRI-adaptation method [18]. fMRI responses tend to decrease monotonically with the number of repetitions, often reaching a plateau after six to eight repetitions. First, one measures the basic repetition suppression effect induced by repetitions of identical stimuli. After repetition, subjects are presented with a stimulus that is varied along one dimension. If the underlying neural representation is insensitive to the change in the stimulus then the fMR signal will be reduced similar to the reduction produced by repetitions of identical stimuli.

fMRI-adaptation has been widely used by researchers to examine selectivity in object-selective cortex to object transformations, object format, perceived shape, contour completion and face representation; to probe higher-level conceptual representations using object pictures and words; and to examine sensitivity to specific types of visual information, such as orientation tuning, color and sensitivity to motion information. Some studies have argued that adaptation occurs across a wide expanse of object responsive cortex, and not only in regions within that cortex that show maximal responses to particular stimuli. Vuilleumier and colleagues used an event related paradigm and found differential sensitivity to object rotation across hemispheres: the left hemisphere displayed invariance to object rotation, the right did not [26]. James et al. found higher invariance to object rotation in the ventral than the dorsal stream

[27]. The different levels of rotation invariance found across studies could depend on the objects used, the degree of rotation and the type of paradigm used.

When measuring the effects of repetitive stimulation on retinotopic fMRI experiments, quantifying the hemodynamic changes due to adaptation plays a crucial role not only because it might affect the accuracy of the measurement but also to determine the potential usage of the retinotopic fMRI in fMRI adaptation research area. Altough the longevity of the adaptation is usually short, it can be still valid in days scale [28].

2.4 Analysis of Borders of Visual Cortex

The positions of the functionally specialized visual areas are only loosely linked to cortical anatomy. Several of these areas are retinotopic, that is, their neurons respond to stimulation of limited receptive fields whose centers are organized to form a continuous mapping between the cortical surface and the visual field. Because of the retinotopic organization of primary visual cortex (V1) and several nearby areas (V2, V3, etc.), it is possible to create stimuli that control the location of the neural activity [1, 2, 29, 30, 3]. A flickering ring that expands slowly from the center of the visual field to the periphery creates a traveling wave of neural activity that spreads from posterior to anterior cortex, along the eccentricity dimension. The activity spans several adjacent visual areas, including V2 and V3, that share a similar retinotopic organization with respect to eccentricity. Similarly, along the angular dimension path that traverses from the lower to the upper lip of the calcarine sulcus, the visual field representation shifts from the upper vertical meridian through the horizontal meridian to the lower vertical meridian. A flickering wedge that rotates slowly about fixation creates a traveling wave of neural activity that spreads from the lower to the upper lip of the calcarine. Because of the different angular representations in V1 and V2, the traveling wave reverses direction at the border. Similarly, visual area V3 surrounds V2 and the traveling wave reverses direction again at the V2/V3 border. Hence, measurements using the rotating wedge stimulus reveal the boundaries between retinotopically organized visual areas [3].

Response to the retinotopic stimulus can be measured using various methods such as electrophysiology or fMRI. Retinotopic fMRI is based on bold contrast which measures the local hemodynamic change due to neural activation. Retinotopic fMRI studies use different approaches, which are based on acquiring fMRI data during phase encoded retinal stimulations, to calculate the visual cortex areas. Retinotopic fMRI procedures also require a surface representation of cortex. Some different approaches to calculate the visual areas are given below.

1. **Visual Field Sign** Adjacent visual areas often have mirror representation of each other. Using this information the areas of the visual cortex (V1, V2, V3...) can be differentiated. The visual field sign designates the orientation of the representation of the visual field on the cortical surface. It indicates if visual field representation of a small area of visual cortex is a non-mirror or mirror image representation of the retina. The phase gradients in retinal eccentricity and polar angle with respect to cortical x and y, ∇r and $\nabla\Theta$, are vector quantities. The visual field sign of the small portion can be determined using the clockwise angle, λ, between ∇r and $\nabla\Theta$. $\lambda < \pi/2$ indicates a non-mirror representation, while $\lambda >$ -$\pi/2$ indicates a mirror representation. Since it is a relative measure, it is insensitive to (a) orientation of areas on the cortex, (b) receptive field coordinate transformations (placement of the center of gaze, the vertical meridian, and so on) [1, 2].

2. **Visual Field Ratio** Another method to determine the visual field sign, is to calculate the ratio of an oriented area measured using the local representation of the visual field coordinates with respect to the same area measured using a locally isometric parametrization of the surface. This quantity is refered as the visual field ratio (VFR). The visual field sign is then the sign of the VFR, and the visual area borders correspond to contour lines of zero VFR. The size of the zone of small absolute VFR around visual area borders gives an immediate visual impression of the uncertainty of the position of the delineated borders, information that is absent from the VFS. Most of the spatial features of the VFR are present in the representation of the polar angle coordinate whose gradient reverses direction at

the borders between visual areas. In contrast, the eccentricity gradient is smooth across visual area borders [29].

3. **Volumetric Visual Field Calculation** Volumetric method that extracts retinotopically mapped visual areas in a completely automatic way. It does not require an explicit reconstruction of the cortical surface which greatly simplifies the analysis. Derivative vectors which identify the cortical surface normals are generated from the anatomical MRI images [31].

3. EXPERIMENTAL SETUP

All MRI data were acquired on the Philips Intera 3 T system at Yeditepe University Hospital. Visual stimulus was projected onto a home built back projection screen made of semi-transparent acrilic material on wooden holder by a Toshiba S10 projector connected to a PC [32]. No special modification was needed on projector, since it was located at the operator room. Volunteers were able to see the screen through the mirror attached to the head coil. The visual field angle was 7 degrees in horizontal and 5 degrees in vertical. Schematic of the setup is given in figure 3.1

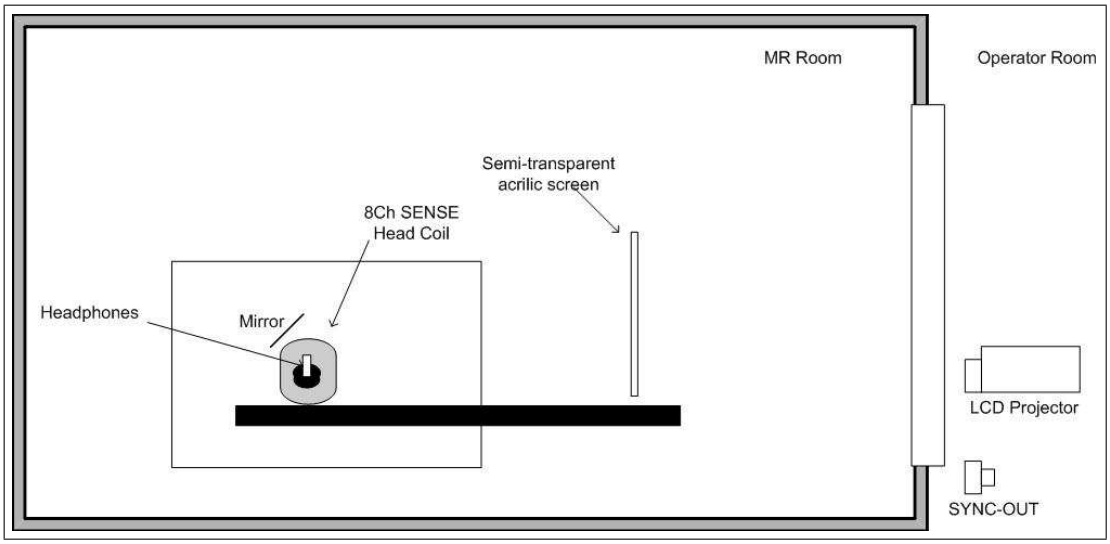

Figure 3.1 fMRI setup

Structural data were acquired with a MPRAGE equivelant high resolution T1 weighted 3D GRE sequence with 1 x 1 x 1.2 mm resolution, 240 mm field of view, 128 slices and 8 degrees flip angle (see appendix A1 for detailed imaging parameters). The body coil was used for RF excitation and a 8 channel SENSE Head Coil for signal detection. Slices were oriented in oblique axial AC-PC aligned plan which can be seen in figure 3.2. Head motion was constrained using small sandbags to the right and left of the subject's head.

Functional MRI data is acquired by a GE-EPI sequence which provides BOLD

contrast with 2 s TR, 30 ms TE, 90 degrees flip angle, 230 mm FOV, 30 slices, 3.2 mm thickness and 64 x 64 matrix (see appendix A2 for detailed imaging parameters). The slice scan of the sequence is in interleaved increasing order (1,3,5...,2,4,6...). In each functional session, high resolution T1 images with the surface generation protocol are acquired in the same plan with fMRI in order to register the functional scans to the constructed surface.

Five volunteers are trained by showing the stimulus everyday using a PC monitor which provides nearly the same visual field angle as the fMRI set-up. fMRI scans are acquired at the end of each week during 3 weeks and once at the begining (4 functional sessions per subject).

An expanding ring shaped checkerboard is used to measure eccentricity and a counter clockwise rotating quarter disk is used to measure polar angle. The snapshots of these stimulus are given in figure 3.3. When the ring reaches maximum eccentricity, it restarts from the center of the eccentricity. The duration was 512 s and the period was 32 s for each stimulus. The flickering rate of the checkerboard was 4 Hz which is near the optimal temporal frequency for driving cortical neurons [2]. Individual squares comprising the stimulus were scaled by the human cortical magnification factor [33]. An eccentricity stimulus expanding at a constant speed is used in order to calculate the cortical magnification factor easily. Since the timing of the stimulus was important, a custom developed special software that is based on the OpenGL library was used to render graphics to the screen. Subjects are asked to keep looking at the fixation dot

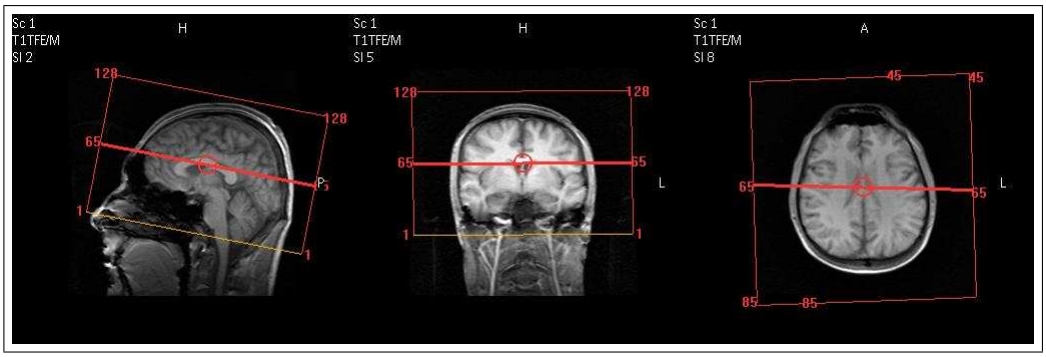

Figure 3.2 Oblique axial MRI plan for both structural and functional scans

at the center of the screen. The basic stimulus frequency was low enough so that there was no whole-cycle phase ambiguity [2].

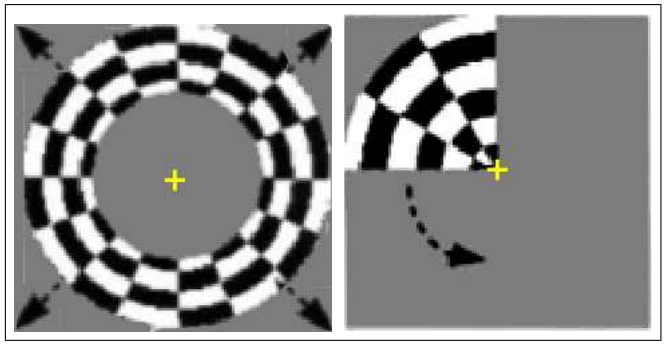

Figure 3.3 Expanding ring stimulus to encode eccentricity (left) and rotating quarter disc stimulus to encode polar angle (right)

4. ANALYSIS

fMRI retinotopic mapping differs at least in two respects from a more "traditional" three-dimensional amplitude based functional analysis: First, the analysis of retinotopy requires the interpretation of functional data in their local spatial context of the sheetlike, highly folded cortical gray matter. This context is not obvious in the three-dimensional Cartesian space in which the data are acquired. It is usually provided by an explicit model of the individual cortical surface used in a surface-based analysis of the functional data. Second, due to the Fourier-type paradigm commonly used for fMRI retinotopic mapping, the main parameter of interest for the functional analysis is the delay (phase) of the observed response, not its amplitude. The processing of this information differs conceptually from an analysis based on the response amplitude alone [29].

4.1 Cortical Surfaces

The cerebral cortex, which makes up the largest part of the human brain, has the topology of a 2D sheet and a highly folded geometry. Most of the features that distinguish the cortical areas can only be measured relative to the local orientation of the cortical surface. A partial list of these includes laminar features (e.g., cortical thickness), as well as retinotopic, tonotopic, and somatotopic maps. There is no way reported to measure these quantities in an unlabeled 3D data set. The two-dimensional nature of the maps as well as their topographic arrangement strongly suggest that a two-dimensional surface-based metric is more appropriate for analyzing their functional properties than the more typically used volume-based metrics [34]. The functional neighbourhood relations are transformed to 2D cartesian space by flattening the cortical surface, which makes calculations easier as seen in figure 4.1.

The highly folded nature of the cortical surface also makes it difficult to view

functional activity in a meaningful way. The typical means of visualization of this type of data is the projection of functional activation onto a set of orthogonal slices. This procedure is problematic as regions of activity which are close to each other in the volume may be relatively far apart in terms of the distance measured along the cortical surface. In addition, the naturally two-dimensional organization of cortical maps is largely obscured by the imposition of an external coordinate system in the form of orthogonal slices [34]. The cortical 'unfolding' procedure rearranges the topographical features (such as cortical area boundaries, columns and interlaminar axonal projections) so that they are all contained within the single plane available on a printed page or video screen. Different ways to display the functional data are shown in figure 4.2. Although the human gray matter is folded tortuously in vivo, unfolding procedures do not necessarily entail any distortion. With a single longitudinal cut, a cylinder or cone can be unfolded without any distortion at all, although a sphere will be distorted by unfolding. The relevant question is whether the human cortex (or portions thereof) is more like a cylinder, a cone or a sphere. Measurements from optimally flattened human and monkey brains yield values for residual distortion (angular and areal) of about 15% [4].

Another application of cortical surface approach is for registration of multisubject data. In order to relate and compare anatomical features or functional activations

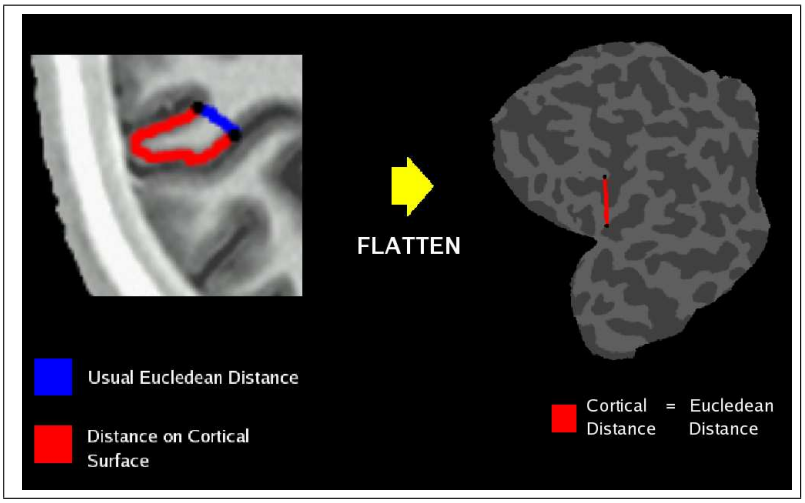

Figure 4.1 Cortical Distances are transformed to eucledean distances by flattening

across subjects, it is necessary to establish a mapping that specifies a unique correspondence between each location in one brain and the corresponding location in another that is, to bring the two brains into register. Most comparisons of data across subjects in the human brain have relied on the 3D normalization approach described by Talairach and Tournoux, however a spherical surface-based coordinate system that is adapted to the folding pattern of each individual subject, allows much higher localization accuracy of structural and functional features of the human brain [35].

FreeSurfer software was used in this project, which was developed in Massachusetts General Hospital, to generate flattened cortical surfaces. This software and related documents can be downloaded from the web. Schematic display of the main steps of the procedure is given in figure 4.3. Details of the algorithms used for each step are explained in the following subsections. Further explanation of the procedure can also be found in related references [34] and [36].

4.1.1 Talairach Registration

The transformation matrix which takes image coordinates into Talairach coordinates is calculated for use in subsequent processing stages. The transformation parameters is computed by using gradient descent at multiple scales to maximize the

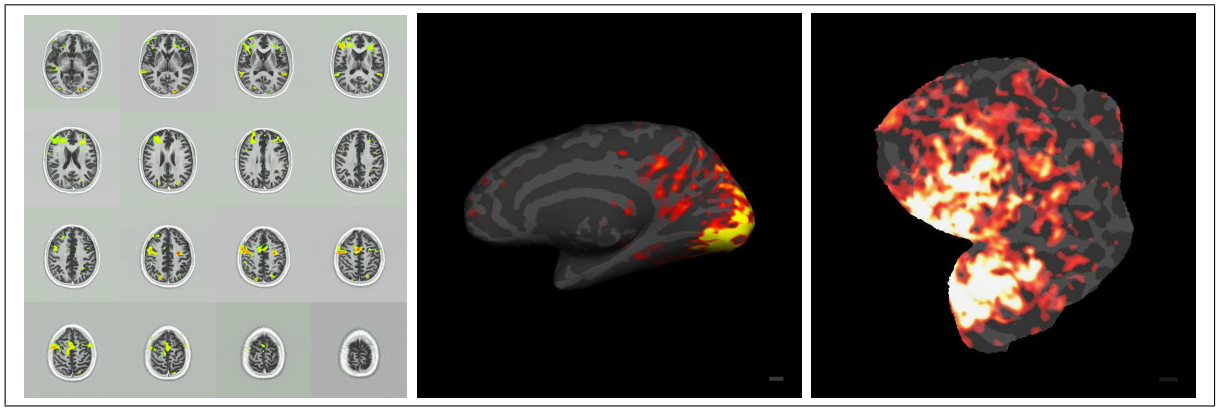

Figure 4.2 Different visualizations of functional data. Mosaic (left), Inflated surface (center), Flattened surface (right)

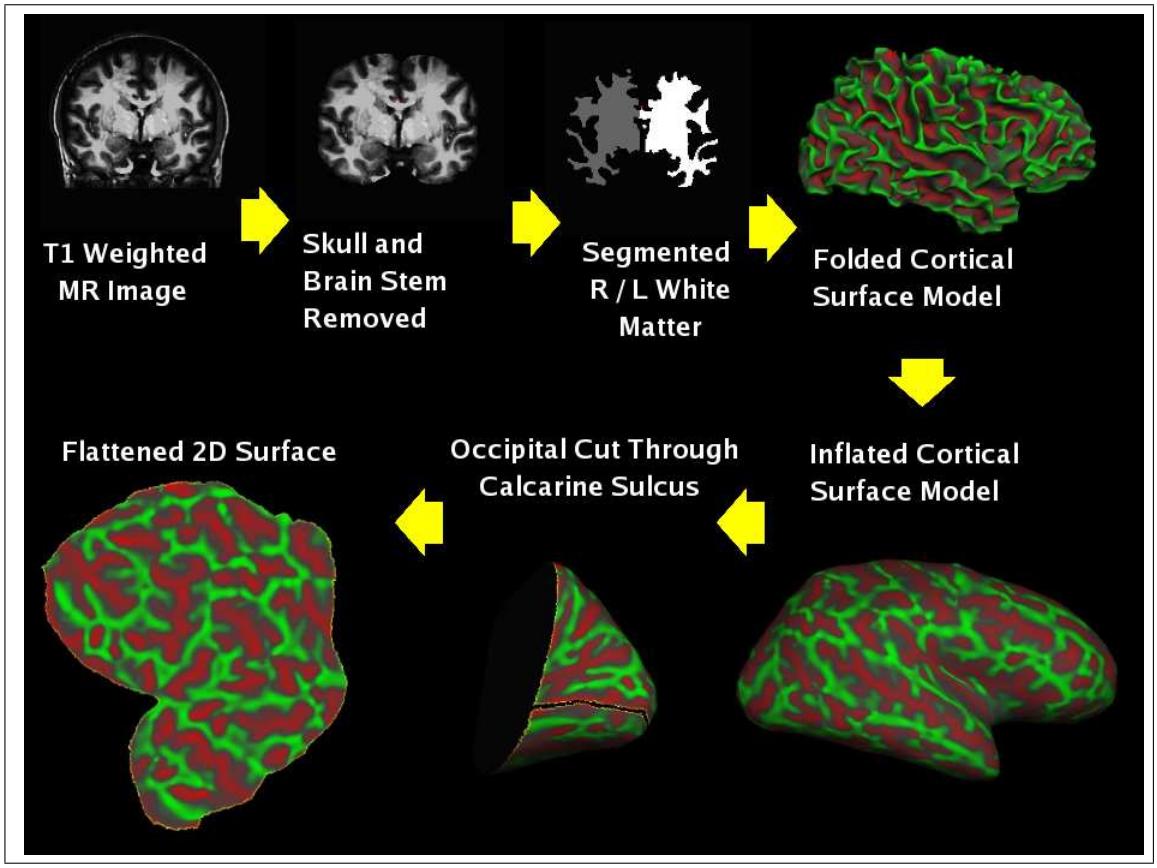

Figure 4.3 Main steps of the cortical surface generation procedure

correlation between the individual volume and an average volume composed of a large number of previously aligned brains.

4.1.2 Intensity Normalization

High resolution T1-weighted images generated by an MR scanner are typically corrupted by magnetic susceptibility artifacts and RF-field inhomogeneities. This causes variations such as different intensities for different spatial locations of the same tissue type and should be corrected for a better segmentation. The procedure relies on the assumption that in any given slice parallel to the xy plane in the magnet coordinate system (perpendicular to the long axis of the bore) the highest intensity tissue type of any significance will be white matter. This assumption is used to center the mean white matter intensity in every xy slice at a desired value. The remaining bias field is then corrected by automatically detecting a set of "control points" which are determined to be in white matter. The intensity value of these points represents a sampling of the remaining bias field. A Voronoi partitioning algorithm is then used to compute the bias field at noncontrol points, that is, non-control points are assigned the value of the closest control point.

4.1.3 Skull-Stripping

This procedure involves deforming a tessellated ellipsoidal template into the shape of the inner surface of the skull. The deformation process is driven by two kinds of "forces": (1) an MRI-based force, designed to drive the template outward from the brain, and (2) a curvature reducing force, enforcing a smoothness constraint on the deformed template. The deformed template is then used to strip the skull from the 3D MRI volume, by removing all voxels outside the tessellated surface.

4.1.4 White Matter Labeling

The cortical surface is known to be smooth, with finite curvature everywhere, resulting in a locally planar structure where cortical gray matter borders other tissue types such as white matter or CSF. The segmentation procedure employs this information by detecting the plane-of-least-variance and using intensity information in this plane as a basis for classifying regions in which intensity information alone is insufficient for accurate tissue classification. This approach prevents thin white matter strands from being blurred with surrounding gray matter or CSF and integrates filtering and model-based knowledge as part of the labeling process. The segmentation process is a two-step procedure. First, a preliminary classification is performed based solely on intensity information. Next, this volume is examined and regions which contain more than one tissue type are marked for further processing. The planar orientation which minimizes within-plane intensity variance is then computed for each of these "border" voxels. The labels generated by using intensity information are changed if the voxel's intensity is ambiguous, some of its nearest neighbors have different labels or a significant number of voxels in the plane of least intensity variance have differing labels.

4.1.5 Cutting Planes

To generate representations of each cortical hemisphere, two cutting planes which prevent connectivity across them are automatically established. The first is a sagittal cut along the corpus callosum which serves to separate the two hemispheres, while the second is horizontal through the pons, removing subcortical structures, allowing us to generate two topologically closed surfaces. Both cutting planes are established using an initial seed point specified by the Talairach coordinate of the relevant structures. Next, the in-plane connected components are computed in a region around each seed point, and the slice which contains the minimal cross-sectional area is identified as the location for the desired cutting plane. Finally, the bounding box of the connected component in this slice is used to specify the coordinates of the entire cutting plane,

with the sagittal cutting plane extended to the bottom of the slice.

4.1.6 Connected Components

A connected components procedure is accomplished to generate two connected solid masses representing the white matter structure of each hemisphere. Starting with seed points in the white matter of each hemisphere, the segmented data is partitioned into six-connected components, discarding all components with no connectivity to the seed points. Next, the complement of the set into connected components, representing all regions classified as nonwhite. Isolated components of this partitioning represent regions interior to the white matter components are then filled. After each phase of the connected components procedure, an iterative 3x3x3 order statistic filter is applied, filling (or unfilling) a voxel if at least 66% of its neighboring voxels are filled (or unfilled). This is done to eliminate invaginations or protuberances.

4.1.7 Surface Tessellation, Refinement, and Deformation

A surface tessellation is constructed for each hemisphere by using two triangles to represent each face separating voxels classified as a white matter of a given hemisphere from differently classified voxels. The surface repositioning used is similar to the "shrink-wrapping" technique. Since the resulting tessellations represent the boundary of the mass of connected voxels for each hemisphere, they must also have the same topology as the surface of the corresponding volume. However, since the voxel faces are necessarily orthogonal to one of the cardinal axes, the resulting tessellation tend to be jagged at the single-voxel scale. To alleviate this effect, initial tessellation is smoothed using a deformable surface algorithm guided by local MRI intensity values, resulting in two smoothly tessellated cortical hemispheres. The minimization of an energy functional based on the tessellated white-matter surface results in a final surface that is both smoother than the initial tessellation and more accurate. The use of the deformable surface procedure necessitates the implementation of an algorithm

to prevent self-intersections from occurring in the surface. Each point in a 4x4x4 mm volume contains a list of faces which intersect it. After each step, the faces attached to the vertex are examined for self intersection. If self intersection is detected, the movement delta is cropped to a point where the self-intersection no longer takes place. Since the connectivity is explicitly maintained, and self intersection is prevented, the topology of the surface cannot change during this deformation process.

4.1.8 Surface Inflation

The high degree of folding of the cortical surface makes it desirable to inflate the reconstructed surface for visualization purposes. This renders the interior of sulci visible, as well as making the surface-based distance between regions more apparent to visual inspection. The purpose of the surface inflation is thus to provide a representation of the cortical hemisphere that retains much of the shape and metric properties of the original surface, but allows the visualization of functional activity occurring within sulci. This is achieved by minimizing an energy functional consists of a spring force which smooths the surface, and the metric-preservation term, which constrains the evolving surface to retain as much of the original metric properties as possible. The inflation process is driven by the average convexity or concavity of a region. That is, points which lie in concave regions move outwards over time, while points in convex regions move inwards. Thus, integrating the normal movement of a point during inflation provides a measure of average convexity or concavity at that point. The average convexity is useful for quantifying the folding pattern of a surface, as it captures large-scale geometric features, while being insensitive to the noise in the form of small folds that typically occur on the banks of a sulcus and relatively stable across individuals.

4.1.9 Flattening

In order to flatten a cortical hemisphere with minimal distortion, a number of cuts on the medial aspect of the original surface is made, one in a region around

the corpus callosum to remove all midbrain structures, one down the fundus of the calcarine sulcus, a set of equally spaced radial cuts, as well as a sagittally oriented cut around the temporal pole. This pattern of cuts removes most of the intrinsic curvature of the surface, allowing it to be flattened with only minor distortion, while preserving the topological structure of the lateral aspect of the surface. The optimal choice of cuts depends on which parts of the surface one is most interested in preserving for a particular application. For example, for analysis of visual data, a planar cut is made that detaches the posterior part of the brain including all of the occipital lobe, and parts of the parietal and temporal lobes. Then a cut down the fundus of the calcarine sulcus is introduced, which separates the upper and lower visual fields of the retinotopic areas and removes most of the intrinsic curvature of the remaining surface. The resulting surface is projected onto a plane whose normal is given by the average surface normal of the cut surface. After the projection has been accomplished, a consistent orientation is given to the flattened surface by setting the normal vector field to z-direction and the surface is allowed to unfold by minimizing the energy functional, using angularly spaced randomly sampled distances in a 0.8 cm radius of each vertex as the neighborhood.

4.2 Frequency Analysis of fMRI Time Series

When the stimulus is seen through a small aperture in the visual field, such as the receptive field of a neuron, the visual field alternates between the flickering checkerboard and the neutral gray field. The alternation of the contrast pattern and uniform field causes a waxing and waning of the neural response. If the stimulus moves at a constant velocity from periphery to fovea, the alternation frequency of the response will be the same for all points in the visual field. An example of fMRI signal obtained from an arbitrary pixel is plotted in time domain in figure 4.4 and in frequency domain in figure 4.5. However, the temporal phase of the response will differ. Neurons with receptive fields in the periphery will respond earlier than neurons with receptive fields near the fovea. Hence, the phase of the response defines the receptive field position along the dimension of stimulus. Retinotopic mapping methods that rely on this phase signal to map the visual field responses are called phase-encoded [3].

4.2.1 Phase Delay Calculation

Motion correction of functional scans is done using AFNI's 3dvolreg. Acquired functional volumes using both stimulus are registered to the volume that is closest to the stuctural scan.

The estimation of the response phase and amplitude is done in three dimensions on a voxel-per-voxel basis. The response to the periodic stimulus is close to sinusoidal, with only a small amount of energy present in higher harmonics. However, the amplitude of the signal present at those harmonics varies greatly between individual voxels and is generally very low. The analysis employed here is based on the fundamental frequency only. The phase of the modulation varies systematically with the position in the cortex. Both, signal amplitude and phase are calculated using the complex valued Fourier transform at the stimulation frequency :

$$F_{v_0}(x_j) = \sum_{k=1}^{N} f(x_j, t_k) e^{-i2\pi v_0 t_k} \tag{4.1}$$

where F_{v_0} is the volume of complex fourier components at stimulus frequency, x_j are the voxel positions, N is the number of time points for a voxel, f is the 4D functional data, t_k is the k^{th} point in time axis. The sign of the exponent depends on the stimulus direction. [30, 29] Alternatively cross-correlation method is used in some studies to determine the phase delay [33]. In this approach, several sinosoidal heomodynamic response models are constructed for different phases. By cross-correlating each of these heomodynamic response models with the time series at a voxel, the maximum correlated phase is assumed to be the phase for that voxel. This cross-correlation analysis is equivelant to fourier transform based delay analysis that is used in this study [30].

While a 2D sequence is used in functional scans, acquisition time for sequential slices is delayed by $2\pi(TR/T_{stimulus})(1/N_{slices})$. The phase angles for the pixels in a slice were then corrected to take into account of delays resulting from the interleaved order in which the slices were collected (1, 3, 5 ... 2, 4, 6...).

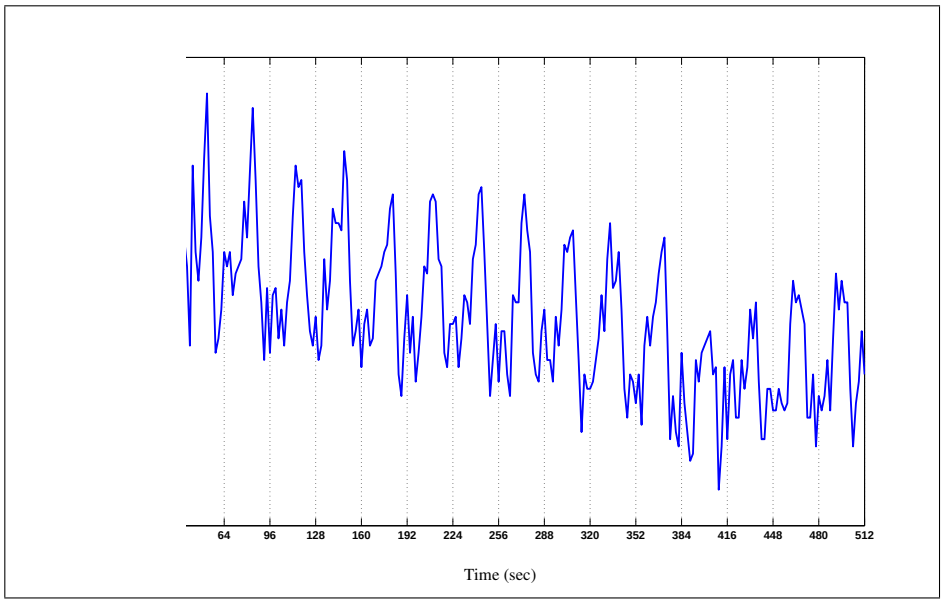

Figure 4.4 Signal vs time plot of an arbitrary pixel

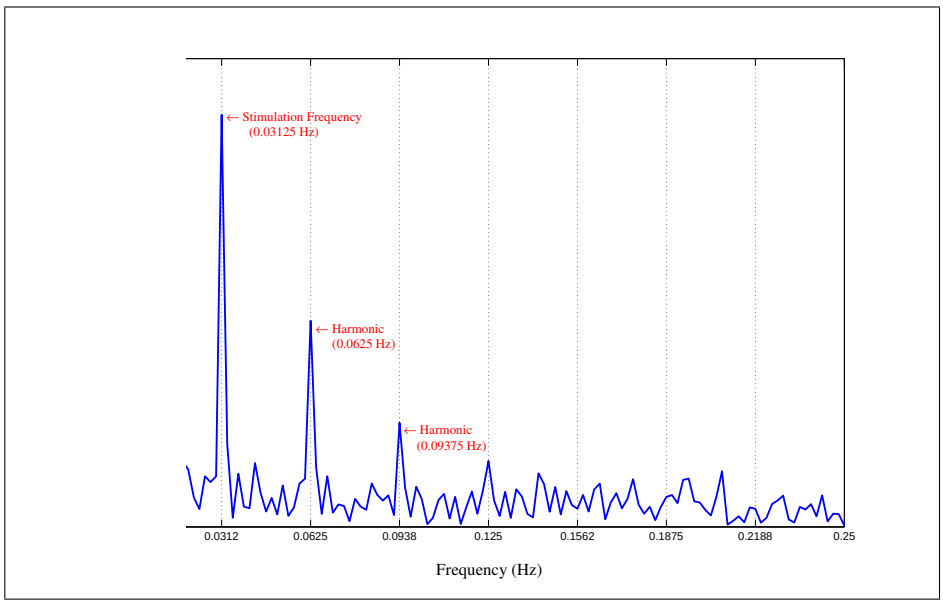

Figure 4.5 Amplitude vs frequency plot of the series in Figure 4.4

As Engel gives the derivation in the related article [30], the correlation coefficient is the amplitude of the response at the stimulus frequency divided by the square root of the timeseries power (eq. 4.2).

$$C = A(f_0) / \left(\sum_{f=1}^{N/2} A(f)^2 \right)^{1/2} \tag{4.2}$$

where f_0 is the stimulus frequency [37].

This value is used as phase significancy. The fourier amplitude at stimulus frequency is replaced by this value at each pixel for further processing. The expected value in the absence of stimuli is $1/(N_{frequencies})^{1/2}$.

A bandwidth selected around the fundamental frequency is used in some studies instead of the whole spectrum [13].

$$C = A(f_0) / \left(\sum_{f=f_0-\Delta f/2}^{f_0+\Delta f/2} A(f)^2 \right)^{1/2} \tag{4.3}$$

Another approach is quantifying the phase significancy by calculating a SNR of the response amplitude. The standard deviation of the phase error is the inverse of the SNR of the response amplitude at the stimulation frequency [29].

The BOLD change percentages are calculated seperately by using the ratio between the stimulus amplitude and the baseline. The amplitude is quantified as the amplitude at the stimulus frequency and the baseline as the DC amplitude.

$$BOLD = (A(f_{stimulus})/A(f_0)) \times 100 \tag{4.4}$$

4.2.2 Painting the Values on Cortical Surface

A surface-based representation of the cortical response from the functional data acquired in three-dimensional Cartesian space is needed. Two issues need to be ad-

dressed. The first concerns the problems inherent in the reduction of dimensionality, assigning volume-based data to the surface model. This step is necessarily nonhomeomorphic. In the presence of misalignment (local and/or global) between functional and structural data, the assignment may induce large errors in the two-dimensional representation of the data. These errors are best exemplified by the case of assigning functional data to the wrong bank of the calcarine sulcus. The issue of alignment needs to be addressed at the moment of data acquisition or by appropriate correction of distorsions prior to assignment of data to the surface model. The second issue is the potential mismatch between the data at the individual voxels and the original cortical response, due to noise and the distance between voxel centers and surface elements. Sources of this mismatch need to be identified and their respective contributions estimated and taken into account in the context of the assignment of phase information [29].

While the aim of the study is to determine the changes of the functional data acquired in different weeks, registering the functional data to the generated surface of subject properly is one of the most critical processes. Because of this reason, structural T1 scans with the protocol used for surface generation, are acquired at the same plan before each session's functional scans. FSL software developed by Oxford University is used for registration. While the surface data is converted to the coronal by freesurfer, A transformation matrix is created which converts the axial T1 data to the coronal. This transformation is applied to the functional data and registered to the surface data using FSL's FLIRT function using 9 degrees of freedom, correlation based algorithm. The obtained transformation matrix is combined with axial to coronal transformation to obtain the total transformation. Finally, FSL registration matrix is converted to 4 by 4 affine transformation matrix of freesurfer.

The real and imaginary parts of the complex data which has the phase of the response delay and amplitude of the coherence and bold signal changes are painted seperately. The values corresponding to each vertex of the midgray surface are sampled and assigned by using the Paint function of Freesurfer. By using a surface nearer the gray/white matter border than the pial surface, it is possible to accurately assign

activations to the correct bank of a sulcus despite the fact that the individual functional voxels were comparable in size to the thickness of the cortex [2].

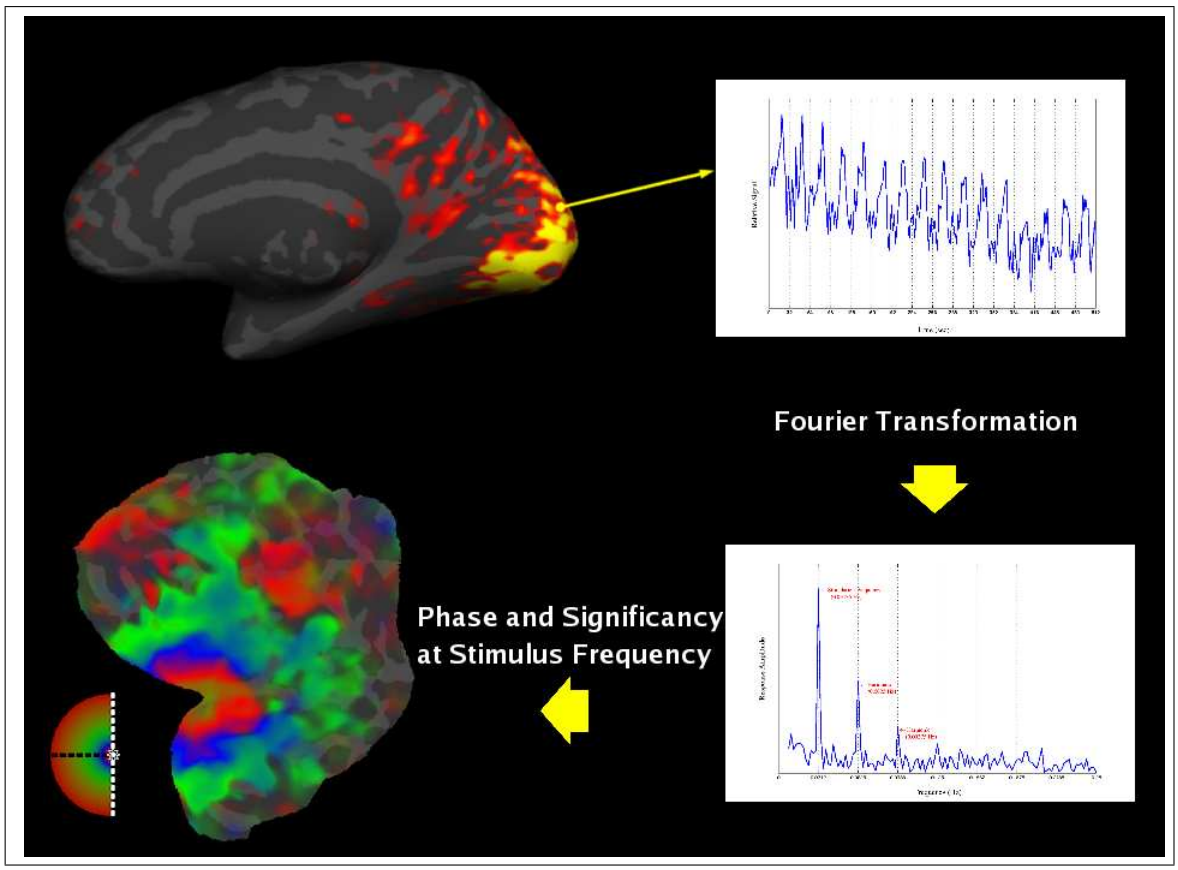

Figure 4.6 Mapping the phases to flattened cortical surface

4.3 Determination of the Visual Cortex Areas

The phase data which is painted on the cortical surface should be smoothed first. Doing smoothing on cortical surface is more logical than smoothing the original 3D data, because the distances and the neighbouring relations are defined in functional organization coordinates. Two problems should be noted while doing this. First, due to the $\pi/-\pi$ warping, smoothing at this border should be handled carefully. And secondly, while averaging the phases the significancy for the phases should be used for weighting. These problems are solved by smoothing the complex values which have both phase and significancy information instead of the phase values only. Also by

this way, the warping problems are removed. To do this a list of neighbour vertices (vertices that share the same faces) are generated for each vertex on the surface. Then the average complex value of all neighbours and the center vertex are assigned to the center vertex. This procedure is repeated 50 times to obtain a smooth representation. This procedure is quite stable thus the result is mostly independent of the number of steps after a value. However the way of quantification of phase significancies affects the obtained results.

The phase gradients on cortical surface are needed to be calculated. While the gradient parallel to the cortical surface is interested, flattened cortical model is used to calculate gradients. A regular grid with the size of the flattened representation is created. Then, the phases of the complex data are interpolated to this grid by using cubic interpolation. Calculating the gradient is straight forward on this regularly sampled 2D data. The phase gradients in retinal eccentricity and polar angle with respect to cortical x and y, ∇r and $\nabla\Theta$, are vector quantities who have units deg/mm.

The warping problem again occurs while calculating the phase gradients. Due to the circular nature of phase, the gradient vectors at the $\pi/-\pi$ boundary will be very high and probably in a false direction. This problem is solved by phase shifting. The phase maps are calculated also after shifting the phases an amount of π. So, the false gradients at $\pi/-\pi$ boundary are shifted to 0. By comparing these two maps and accepting the lower gradient for each pixel a final map is calculated which is filtered of these faults. An example of original, shifted and corrected map is given in figure 4.7.

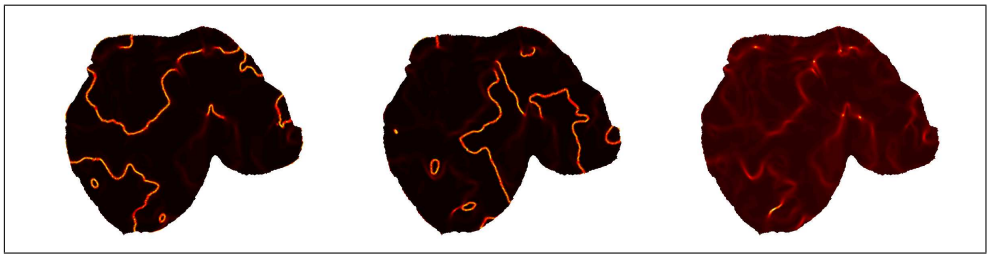

Figure 4.7 Phase gradient unwarping

After calculating the phase gradients for polar and eccentricity stimulus, The visual field sign method is used to determine the areas of visual cortex. The field sign

of a point is the sign of the cross product of eccentricity and polar phase gradients.(eq. 4.8)

$$VFS = sign(\nabla r \times \nabla \Theta) = sign(\nabla r_x \nabla \Theta_y - \nabla r_y \nabla \Theta_x) \tag{4.5}$$

And finally, the obtained VFS data on regular grid is resampled at vertex coordinates of the surface and displayed on surfaces. The schematic of VFS calculation is shown in figure 4.9. In order to do all this processes, a matlab toolbox is developed which reads the surfaces and data in freesurfer format does smoothing on surface and regular gridding (see appendix B). The matlab code used for calculating the VFS is also given in the appendix. An example of obtained VFS results visualized in different ways is shown in figure 4.9.

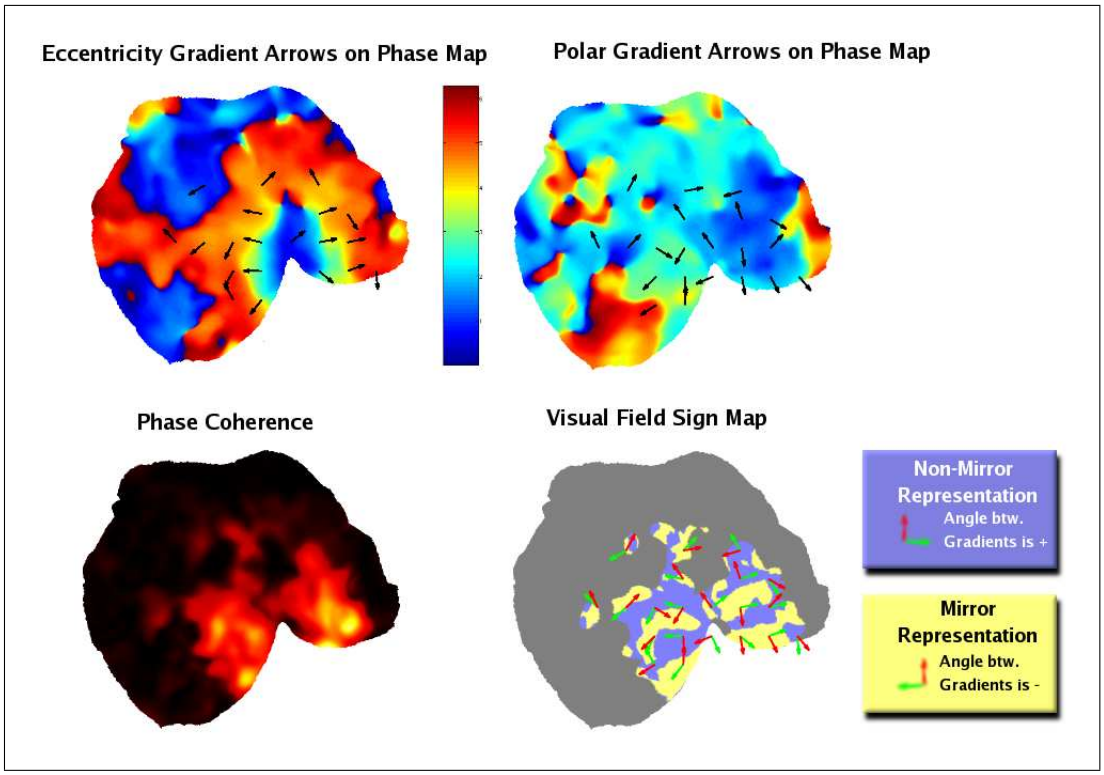

Figure 4.8 Determination of visual field signs using phase gradients

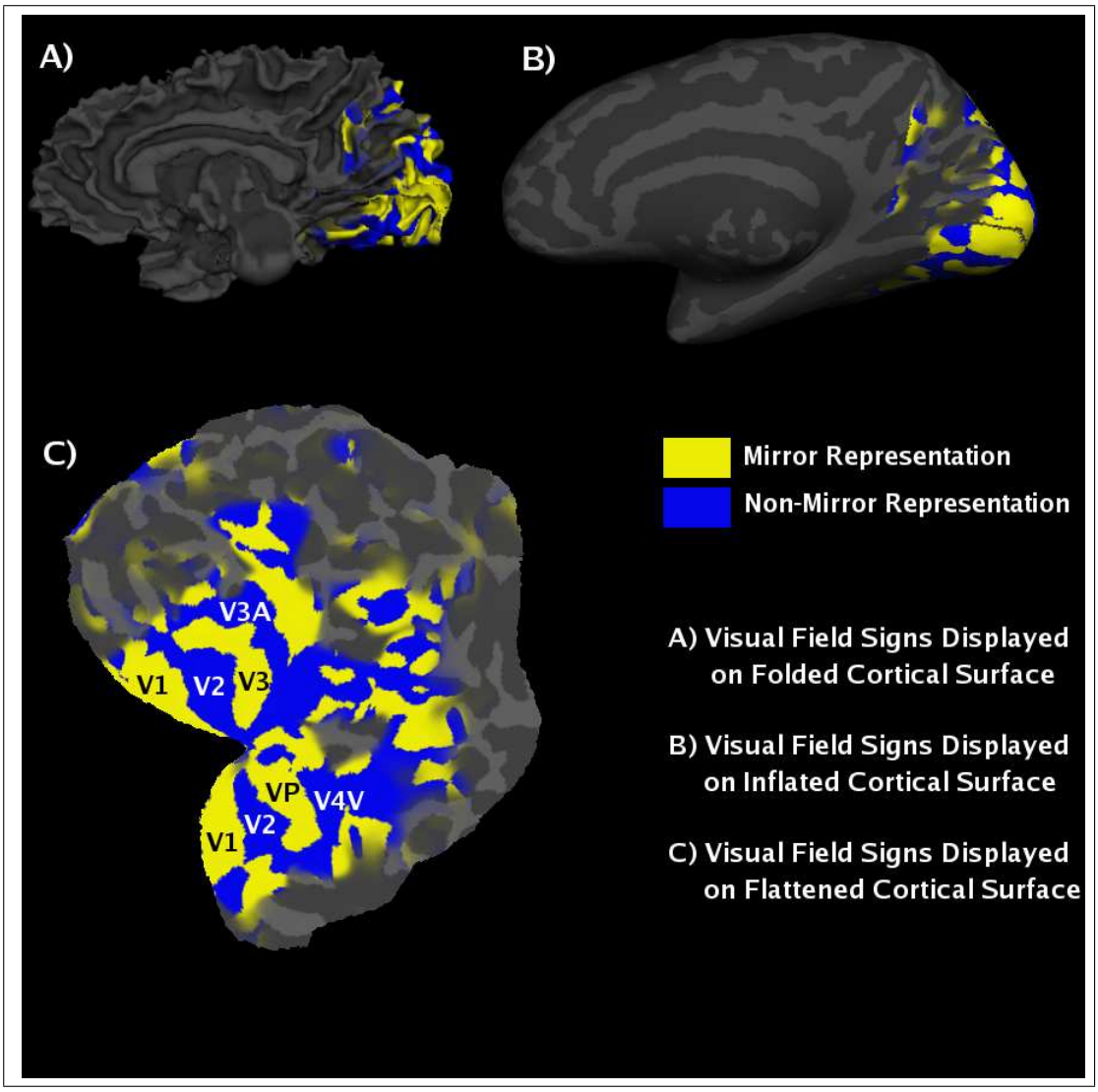

Figure 4.9 Different visualizations of visual field signs

4.4 Internal Structure: Cortical Magnification Factor

The internal structure of a visual cortical map can be characterized using the linear cortical magnification factor, M(r) mm of cortex per deg of visual angle as a function of eccentricity, r [2]. Foveated objects at zero eccentricity activate a much larger area of primary visual cortex than the same size object located several degrees off the line of sight. Estimating cortical magnification as a function of eccentricity is a useful means to describe the overall visual-cortical architecture of the human primary visual cortex. The human cortical magnification can be estimated using electrophysiological, psychophysics, cortical stimulation, and functional magnetic resonance imaging [38].

The falloff in linear cortical magnification factor, M, (mm of cortex per deg of visual angle) with increasing eccentricity has often been described using an equation of the form:

$$M(r) = A[r + B]^{-C} \tag{4.6}$$

where r is the eccentricity, and A, B, and C are constants. Sereno et al. [2] have calculated M using these formula and find the results for human upper visual field V1, V2, VP and V4 as :

$$\begin{aligned} M_{V1}(r) &= 20.05[r + 0.08]^{-1.26} \\ M_{V2}(r) &= 25.19[r + 0.09]^{-1.53} \\ M_{VP}(r) &= 18.28[r + 0.24]^{-1.75} \\ M_{V4v}(r) &= 18.17[r + 0.24]^{-1.55} \end{aligned} \tag{4.7}$$

Engel et al [30] developed a simple formula relating position within area V1 (millimeters from the point representing 10^o) to position within the visual field (degree of visual angle):

$$lnE = 0.63D + 2.303 \tag{4.8}$$

where E is eccentricity measured in degree from fixation, and D is cortical distance (in millimeters) from the cortical position representing 10^o of eccentricity. Cortical locations anterior to that point are assigned positive values, and cortical locations posterior to that point are assigned negative values. This representation agrees well with estimates obtained using completely different means in monkeys and humans [3].

In our study, cortical magnification is quantified by calculating the inverse of the average eccentricity gradient amplitudes for each .5 degrees of eccentricity.

5. RESULTS

5.1 Visual Area Maps

In this section, visual cortex areas of 5 subjects mapped on flattened surface are given for left and right hemispheres during 3 weeks. Also the curvatures mapped on generated surfaces are given as structural reference.

In figure 5.1 flat cortical surfaces of occipital lobe obtained using the procedure explained in section 4.1 are given. These surfaces belong to FA, VB, UE, AK and SO in the order from top to bottom. Right hemispheres of the subjects are on the left and the left hemispheres are on the right of the page. Red-green color coding indicates the amount of displacement for each point during inflation, so sulcus are painted with red and gyrus are painted with green color.

Visual field sign maps determined for four scans of five subjects are given in figures from 5.2 to 5.6. Yellow areas are mirror and blue areas are non-mirror representations of the visual field, so adjacent visual areas have different colors. Right hemisphere of the subject is at the left column and the left hemisphere is at the right column. Rows are ordered in scan order from top to bottom. Thus, the topmost row is the result without any training, the second row is the result after first week and so on. Anatomical location of the visual areas can be determined by using figure 5.1.

Visual areas are quite stable for four scans. This verifies the robustness of the retinotopic fMRI method. However they are not precise enough to give a conclusion about the slight effects of training on measured visual area borders. Despite of the stability of visual areas, the size of the areas that have signal above the treshold have high variation. In week #2 of AK (figure 5.5) size of the painted area is much higher and in week #4 of the SO it is much less than the other scans. Altough, there is a general trend of decrease for UE, AK and SO, this is not true for FA and VB.

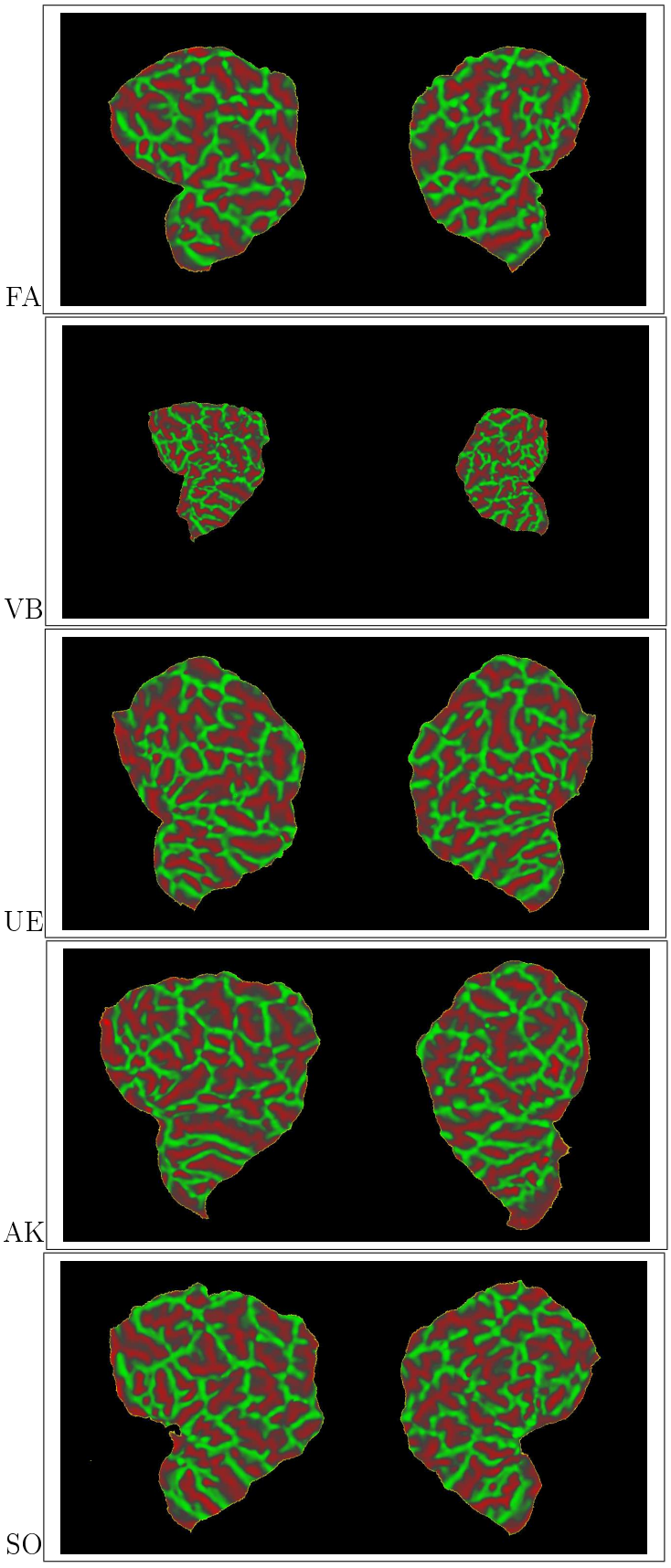

Figure 5.1 Flat surfaces for FA, VB, UE, AK, SO (RH on left, LH on right)

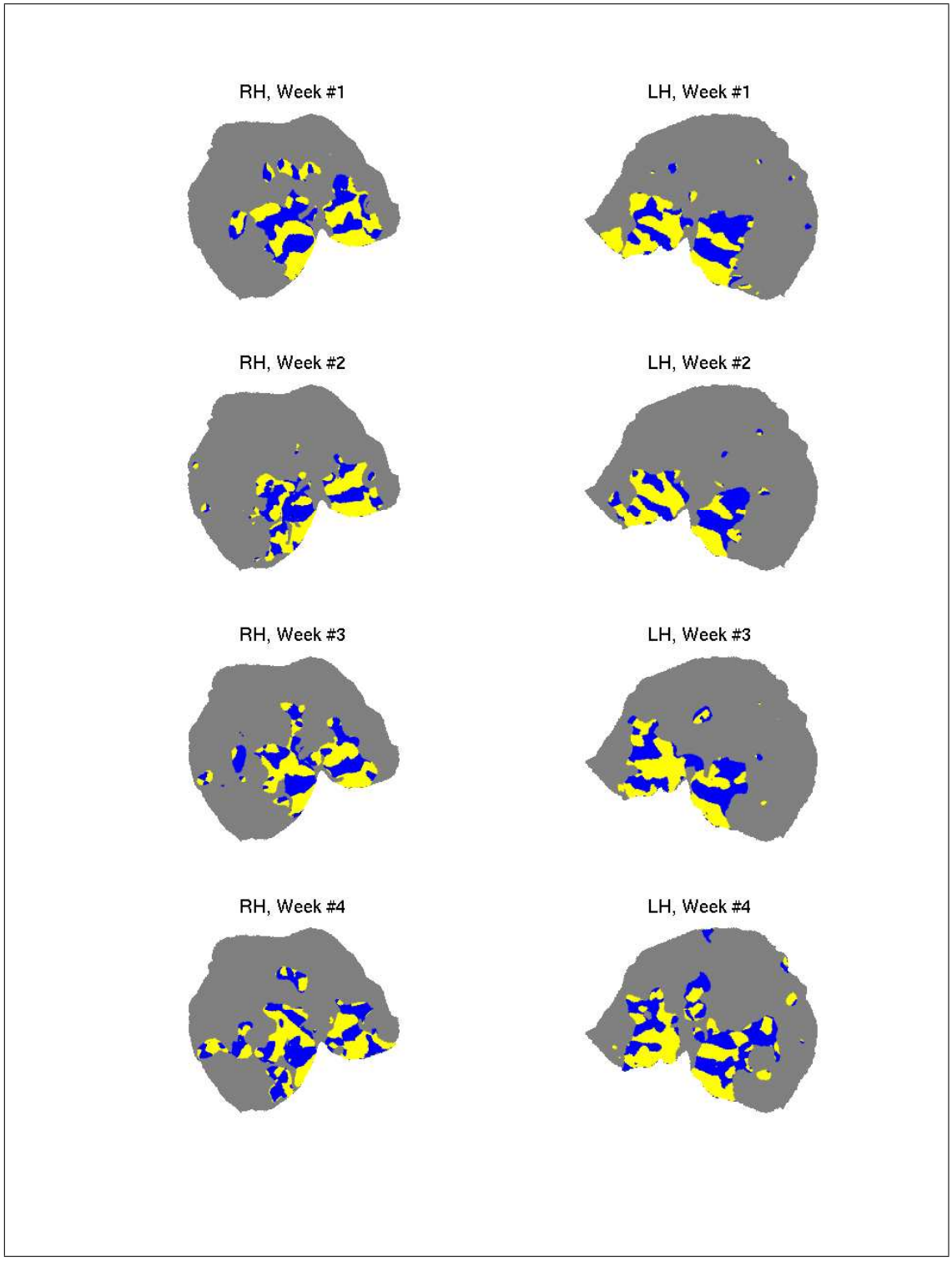

Figure 5.2 Visual field sign maps for subject FA

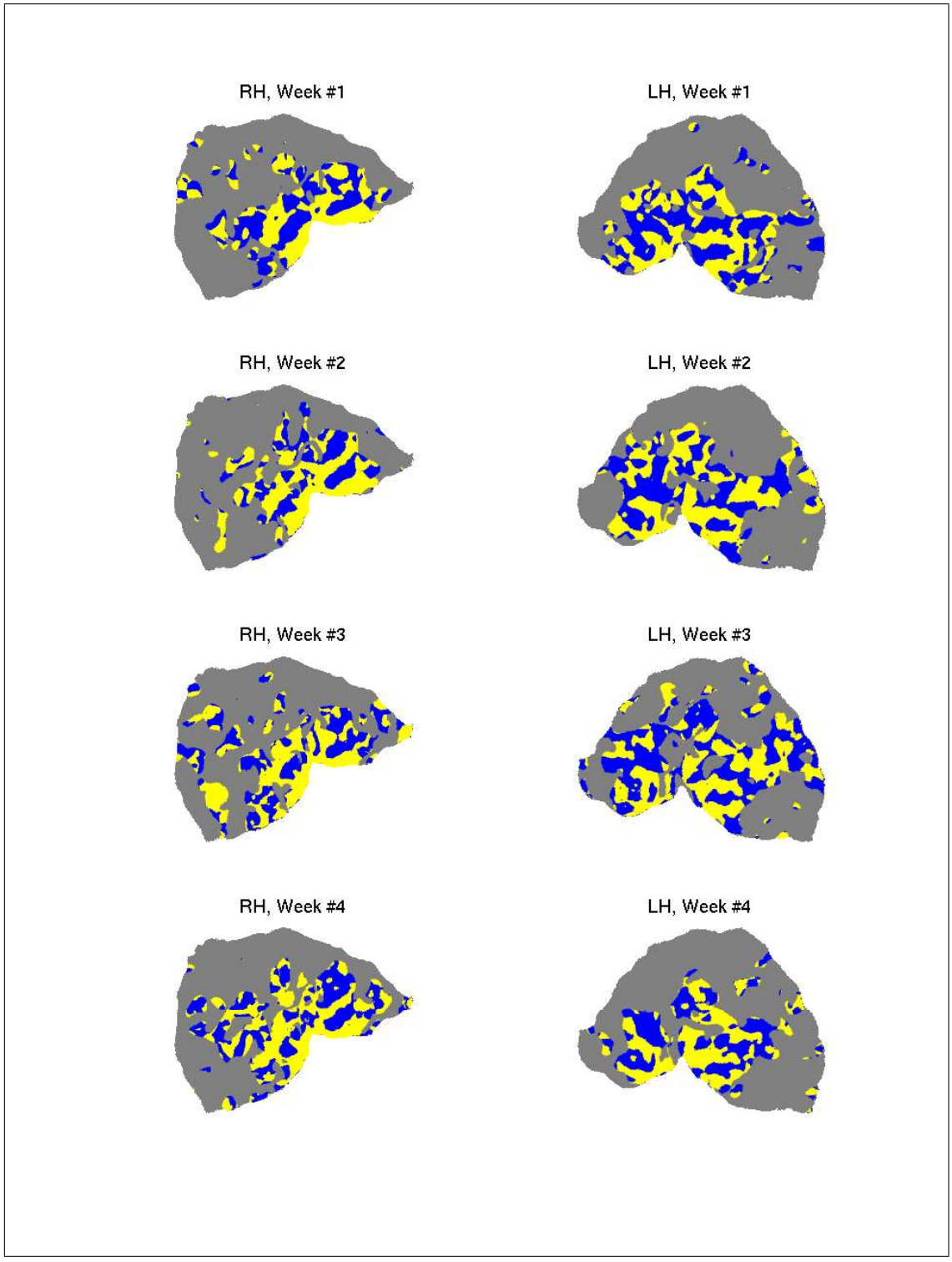

Figure 5.3 Visual field sign maps for subject VB

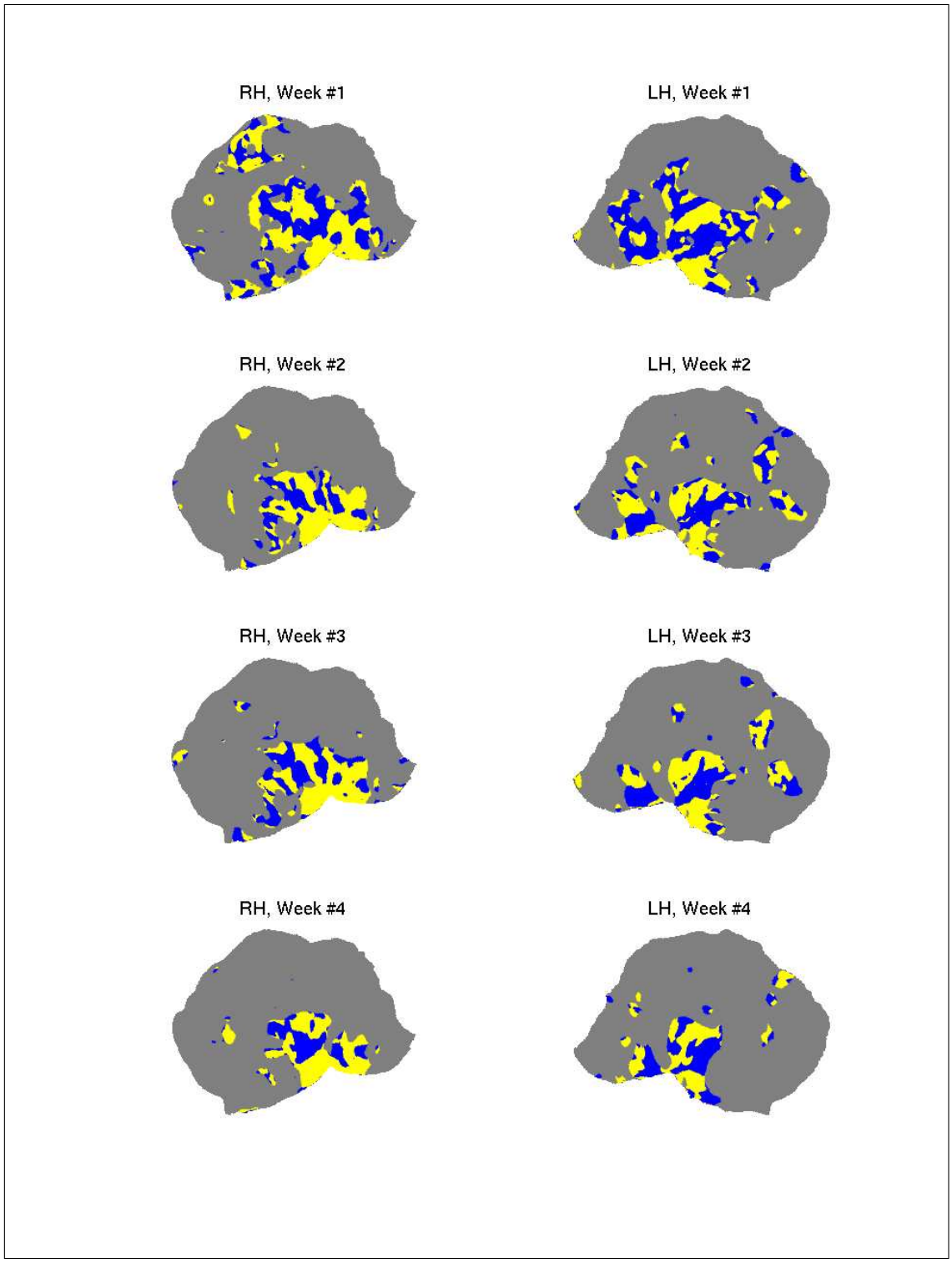

Figure 5.4 Visual field sign maps for subject UE

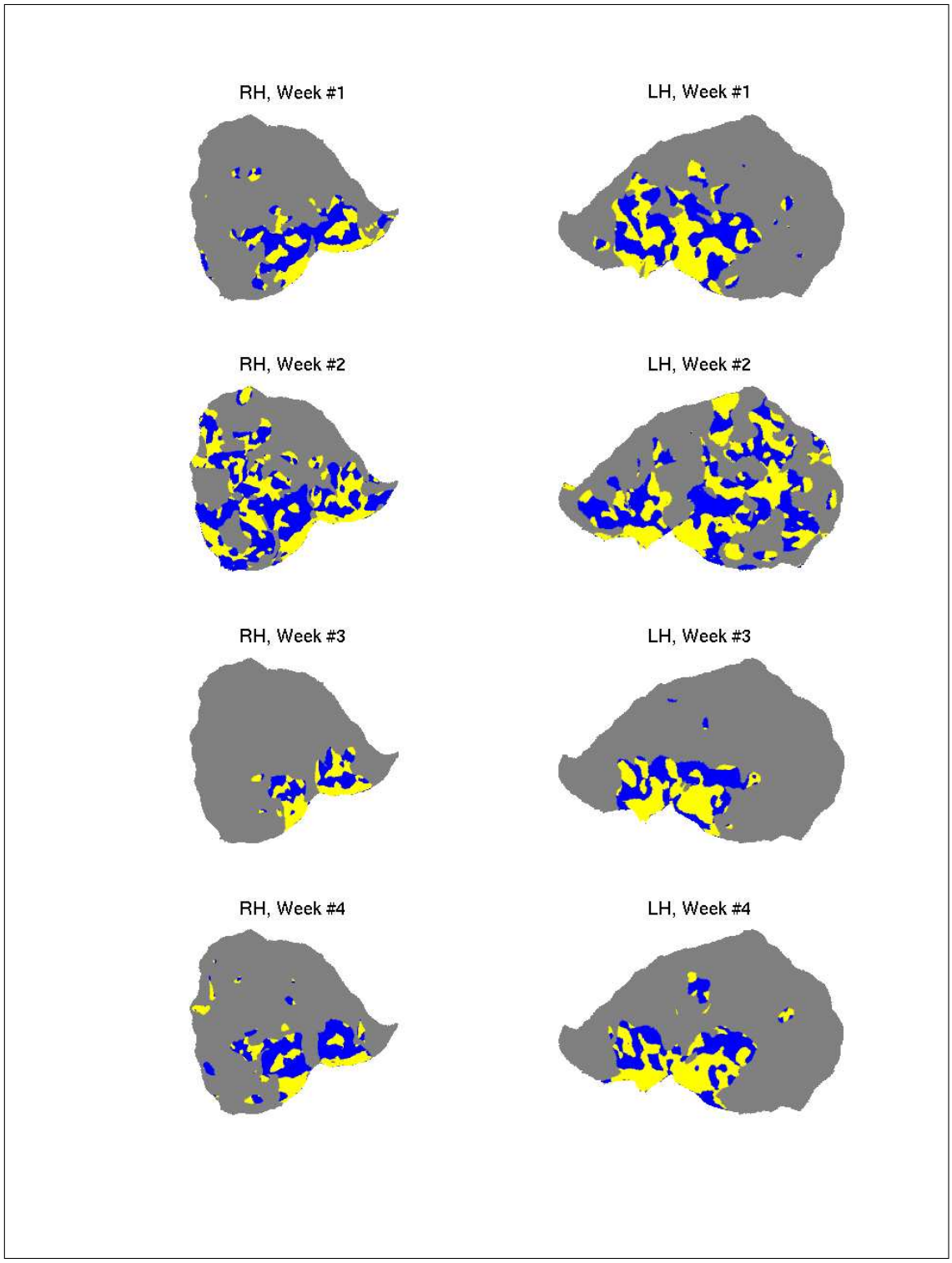

Figure 5.5 Visual field sign maps for subject AK

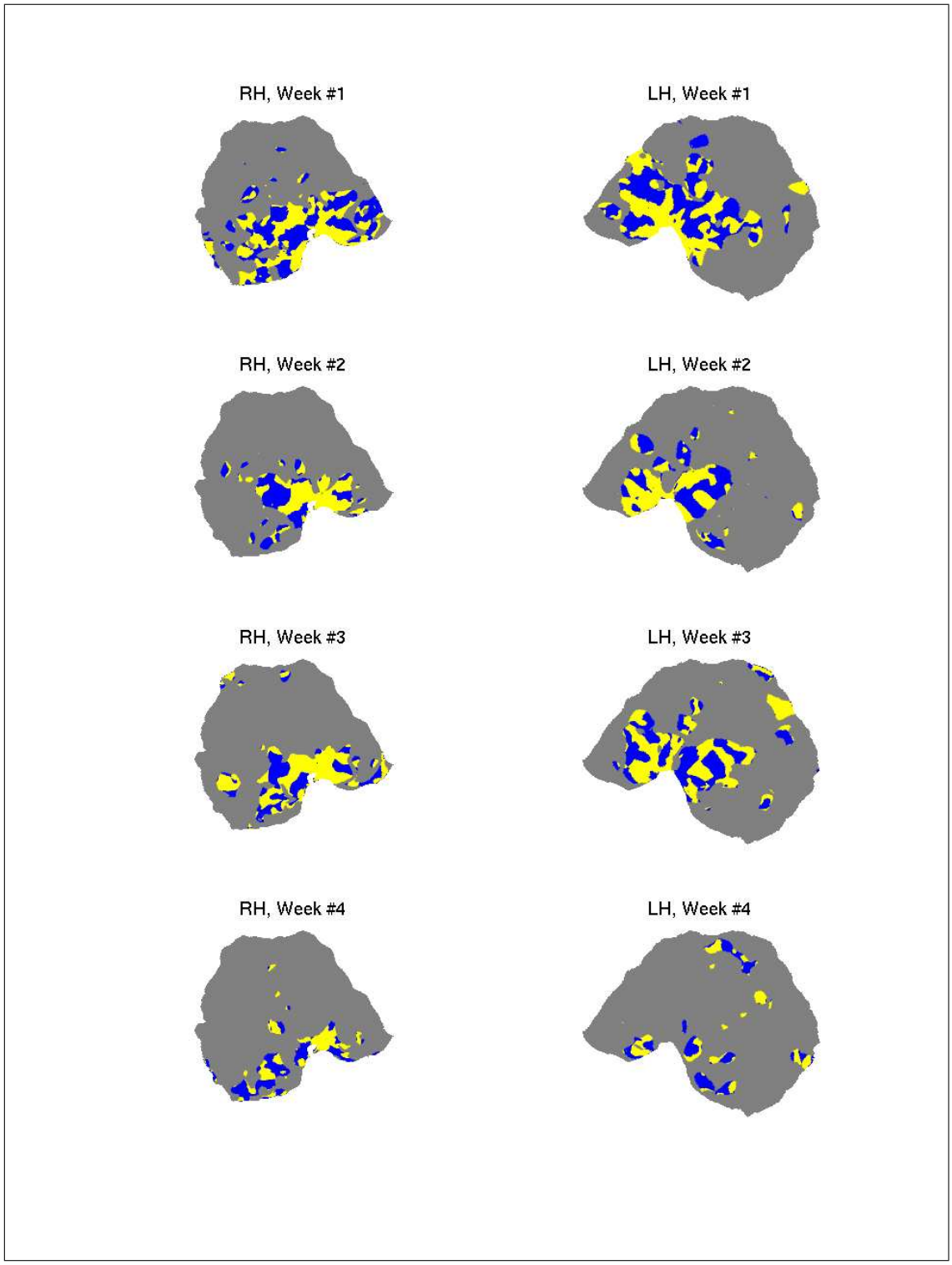

Figure 5.6 Visual field sign maps for subject SO

5.2 BOLD Response

In this section, BOLD response changes due to expanding and rotating stimulus for both hemispheres of subjects are given seperately. The change of average BOLD response measured from the manually drawn ROI of V1 area is plotted versus four scans for right hemisphere in figure 5.7 and for left hemisphere in figure 5.8. Measured values are also tabulated in tables 5.1 and 5.2. Without being significant, there exists a general trend of signal decrease for left hemispheres.

Distribution of the BOLD response changes are shown on flat occipital surfaces in figures from 5.9 to 5.13 for visual inspection. The signal at left hemispheres is higher than right hemispheres which might be due to the left dominancy of the subjects.

Section 5.22 is the same of 5.2.1 for rotating stimulus. Higher signal at left hemispheres can also be observed for rotating stimulus.

5.2.1 Expanding Stimulus BOLD Signal Changes

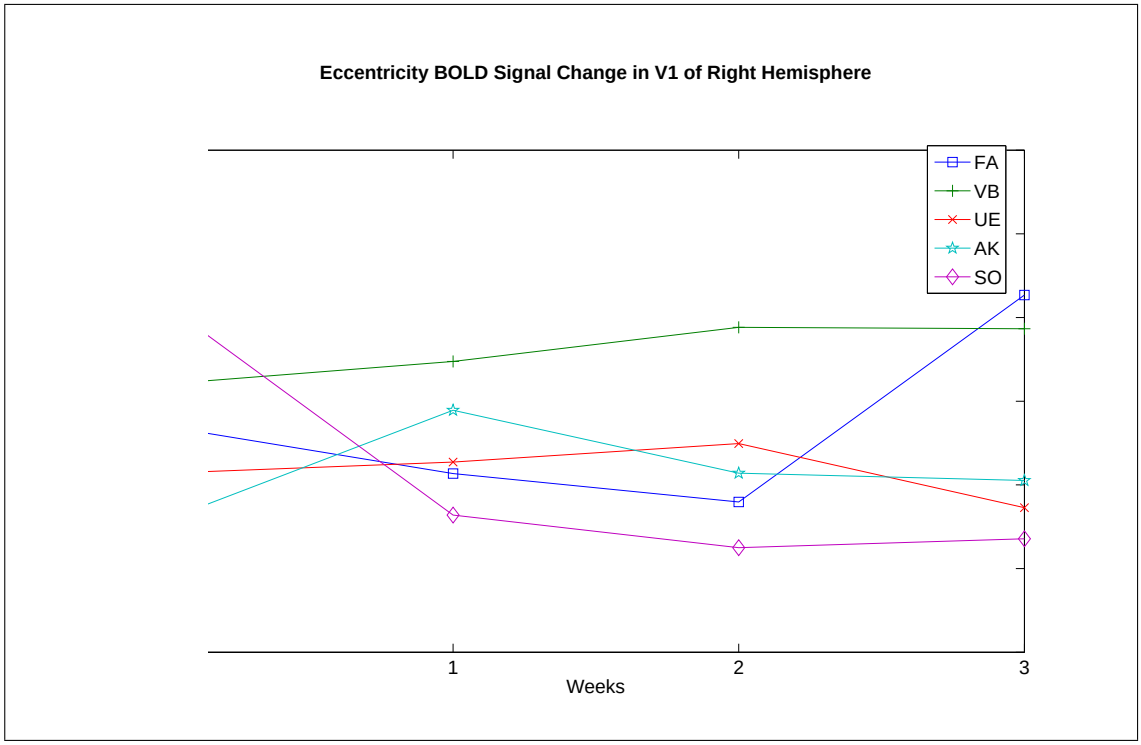

Figure 5.7 BOLD signal change percentage of right hemisphere V1 area due to expanding stimulus

Table 5.1
BOLD signal change percentage of right hemisphere V1 area due to expanding stimulus

	Week #0	Week #1	Week #2	Week #3
Subject FA	0.4694	0.4135	0.3795	0.6269
Subject VB	0.5207	0.5475	0.5883	0.5866
Subject UE	0.4141	0.4273	0.4494	0.3728
Subject AK	0.3587	0.4891	0.4140	0.4052
Subject SO	0.6138	0.3639	0.3249	0.3356

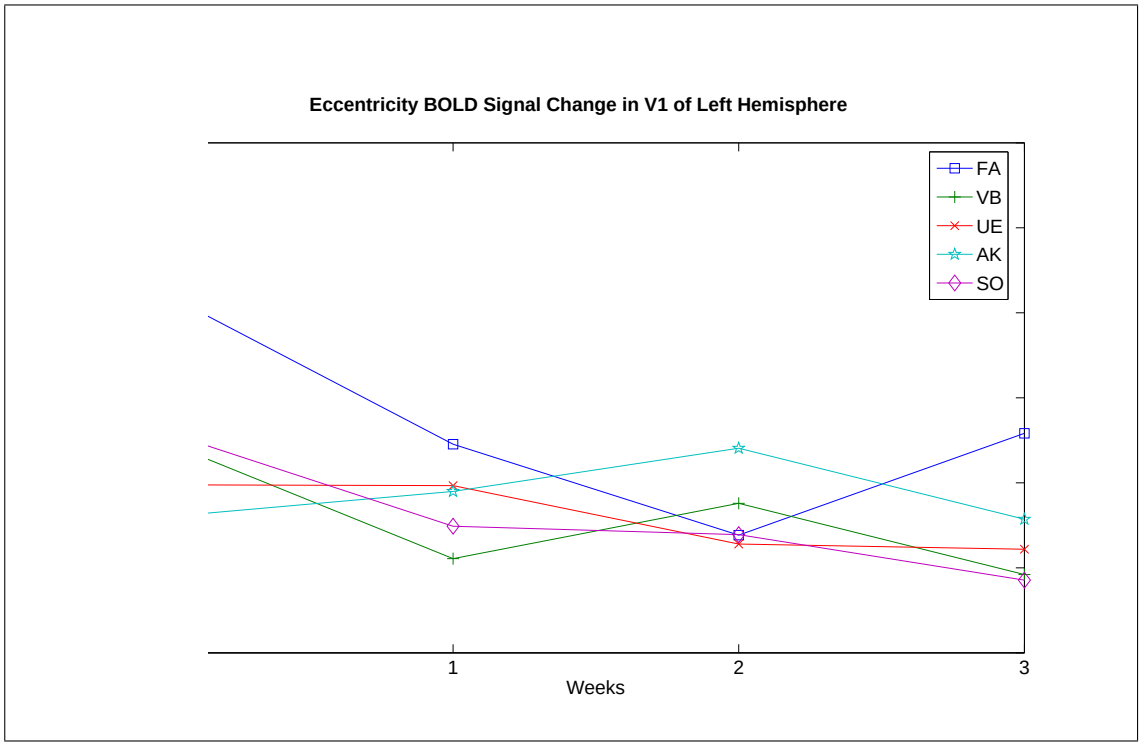

Figure 5.8 BOLD signal change percentage of left hemisphere V1 area due to expanding stimulus

Table 5.2
BOLD signal change percentage of left hemisphere V1 area due to expanding stimulus

	Week #0	Week #1	Week #2	Week #3
Subject FA	0.6208	0.4454	0.3385	0.4582
Subject VB	0.4469	0.3108	0.3758	0.2921
Subject UE	0.3976	0.3967	0.3280	0.3219
Subject AK	0.3602	0.3899	0.4407	0.3572
Subject SO	0.4589	0.3489	0.3390	0.2855

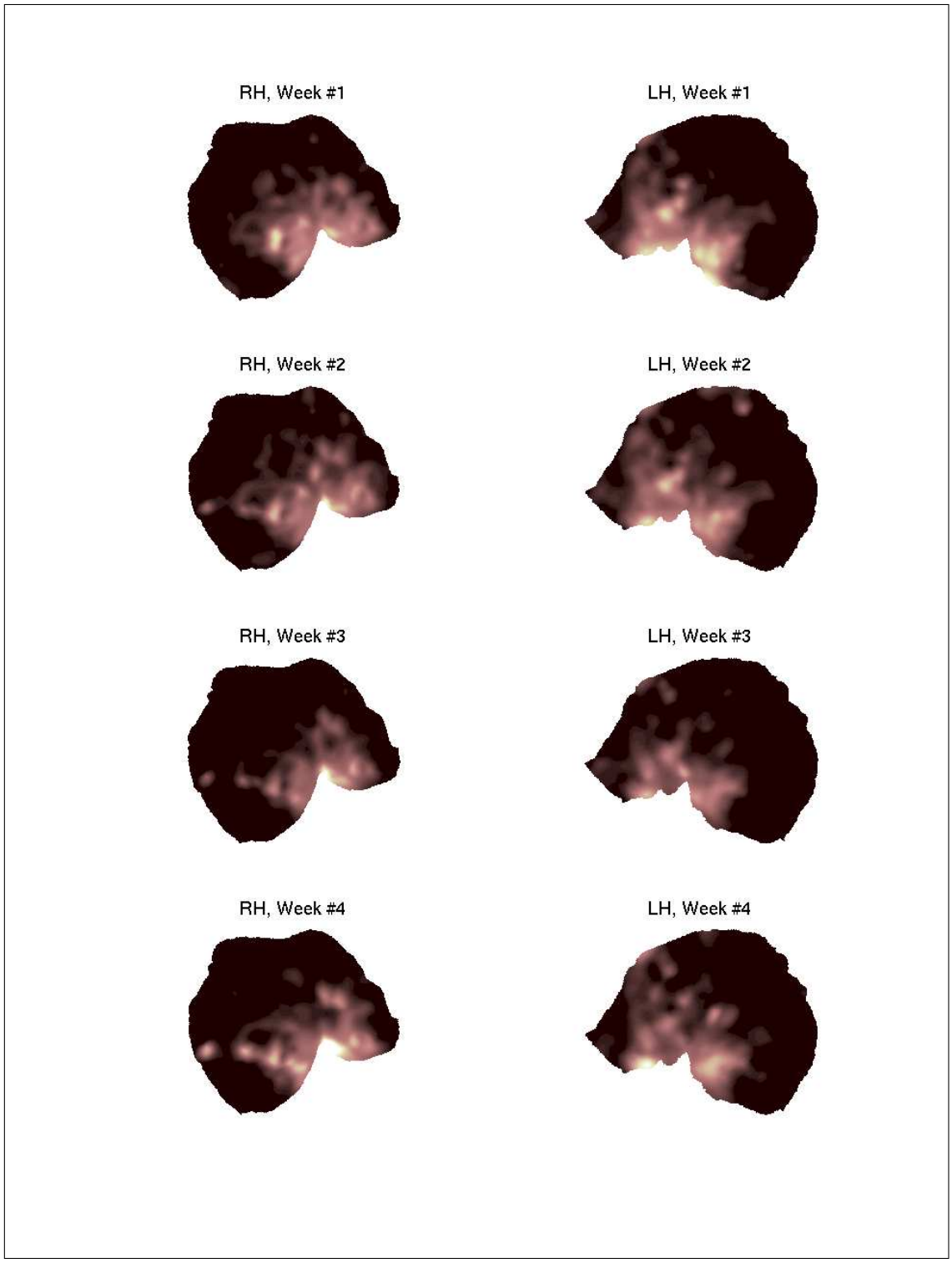

Figure 5.9 % BOLD change due to eccentricity stimulus for subject FA

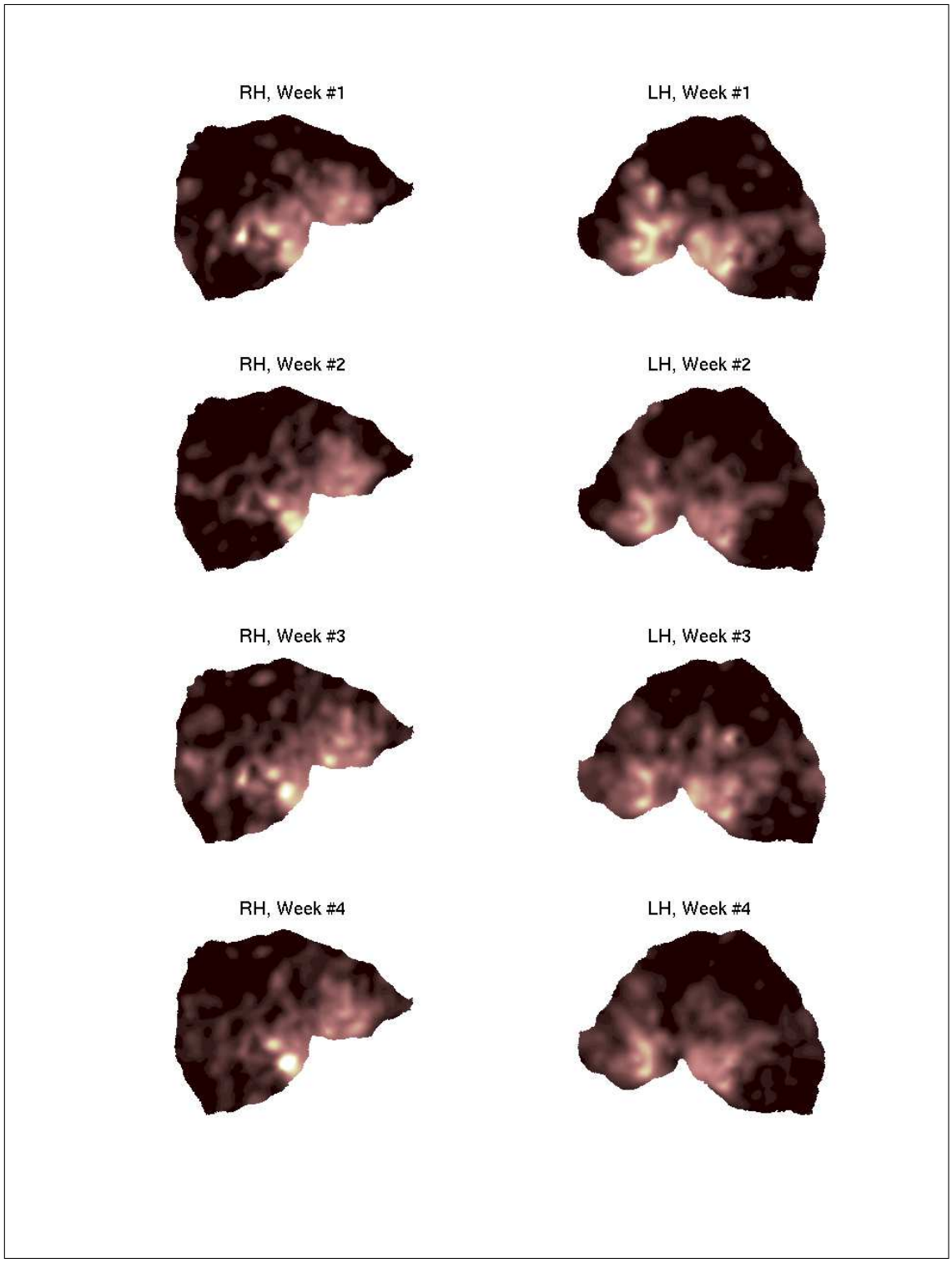

Figure 5.10 % BOLD change due to eccentricity stimulus for subject VB

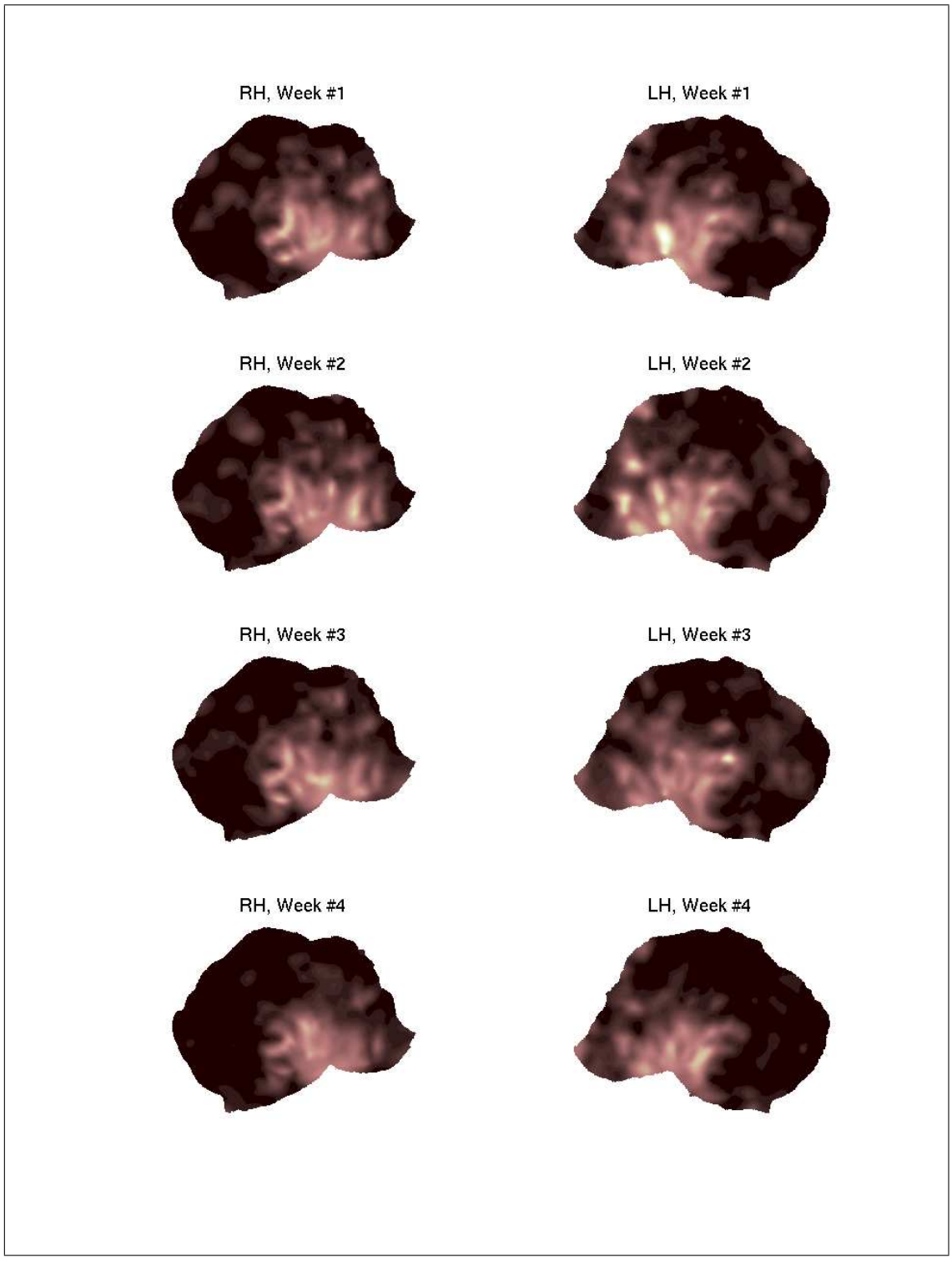

Figure 5.11 % BOLD change due to eccentricity stimulus for subject UE

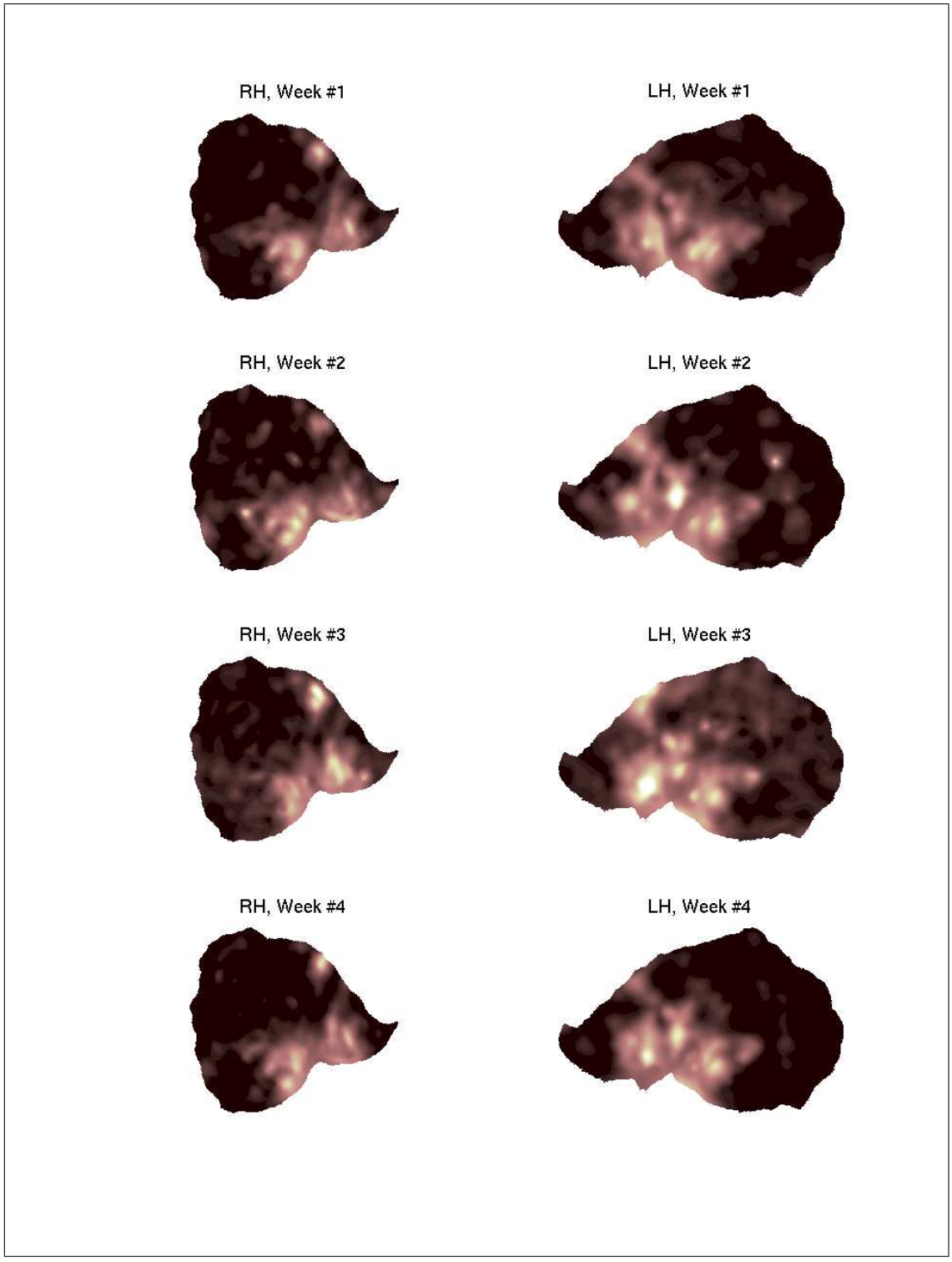

Figure 5.12 % BOLD change due to eccentricity stimulus for subject AK

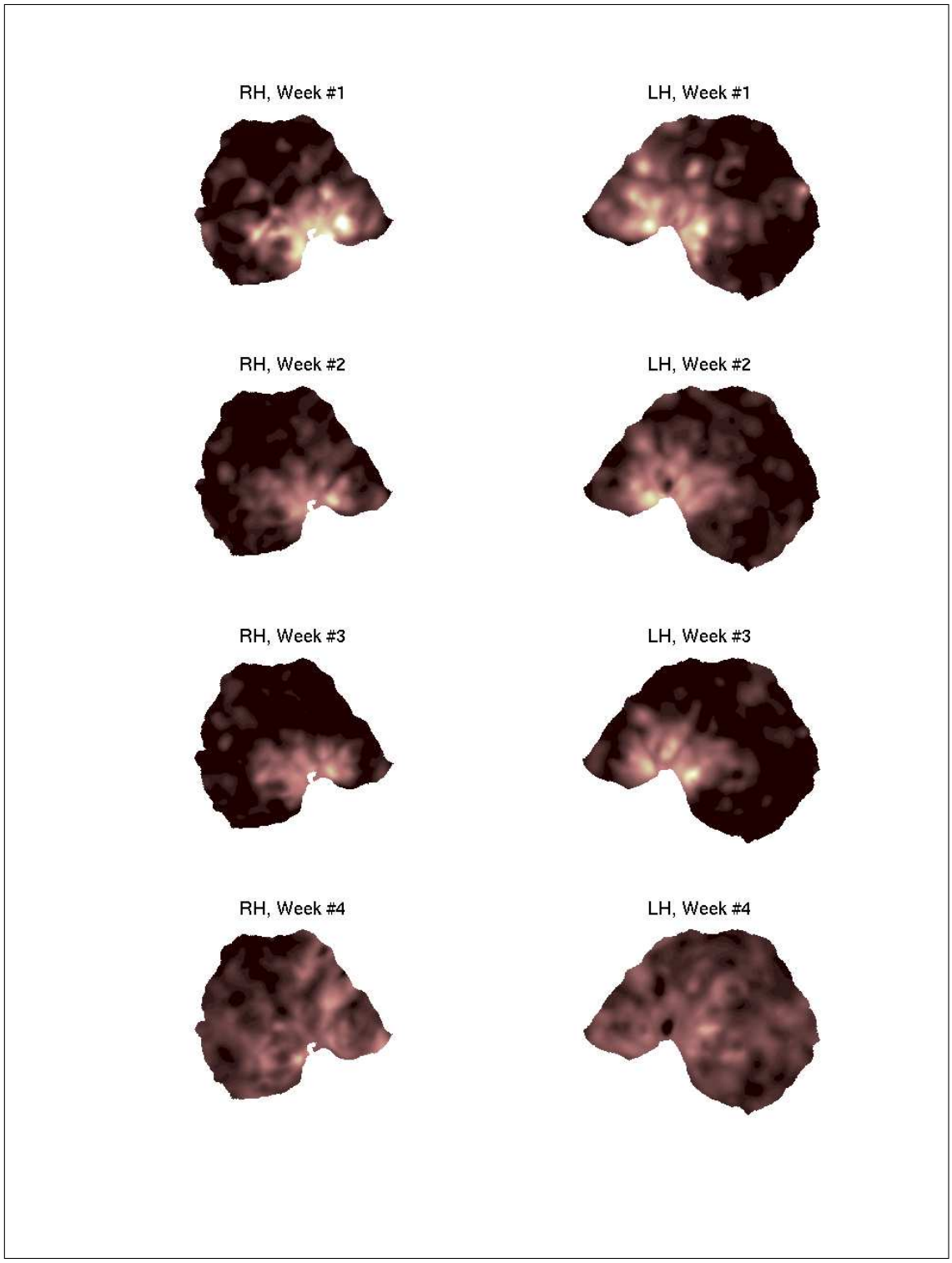

Figure 5.13 % BOLD change due to eccentricity stimulus for subject SO

5.2.2 Rotating Stimulus BOLD Signal Changes

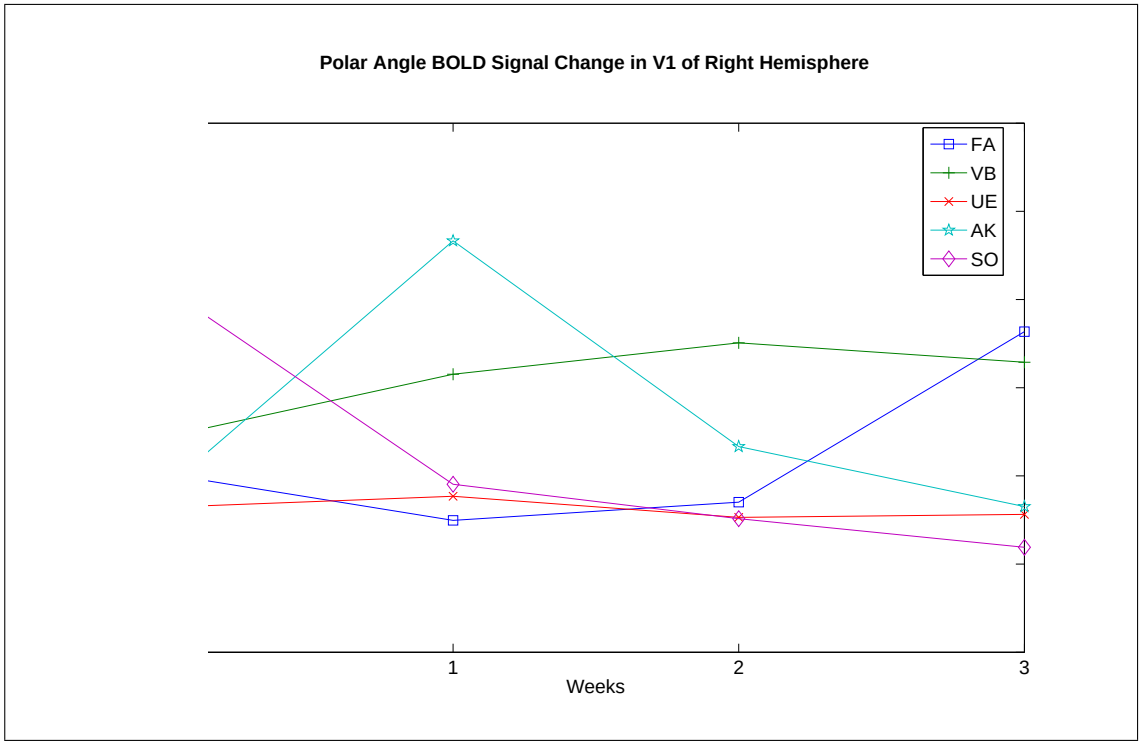

Figure 5.14 BOLD signal change percentage of right hemisphere V1 area due to rotating stimulus

Table 5.3
BOLD signal change percentage of right hemisphere V1 area due to rotating stimulus

	Week #0	Week #1	Week #2	Week #3
Subject FA	0.4018	0.3495	0.3702	0.5635
Subject VB	0.4448	0.5152	0.5509	0.5289
Subject UE	0.3645	0.3769	0.3529	0.3563
Subject AK	0.3877	0.6665	0.4333	0.3652
Subject SO	0.6113	0.3905	0.3515	0.3191

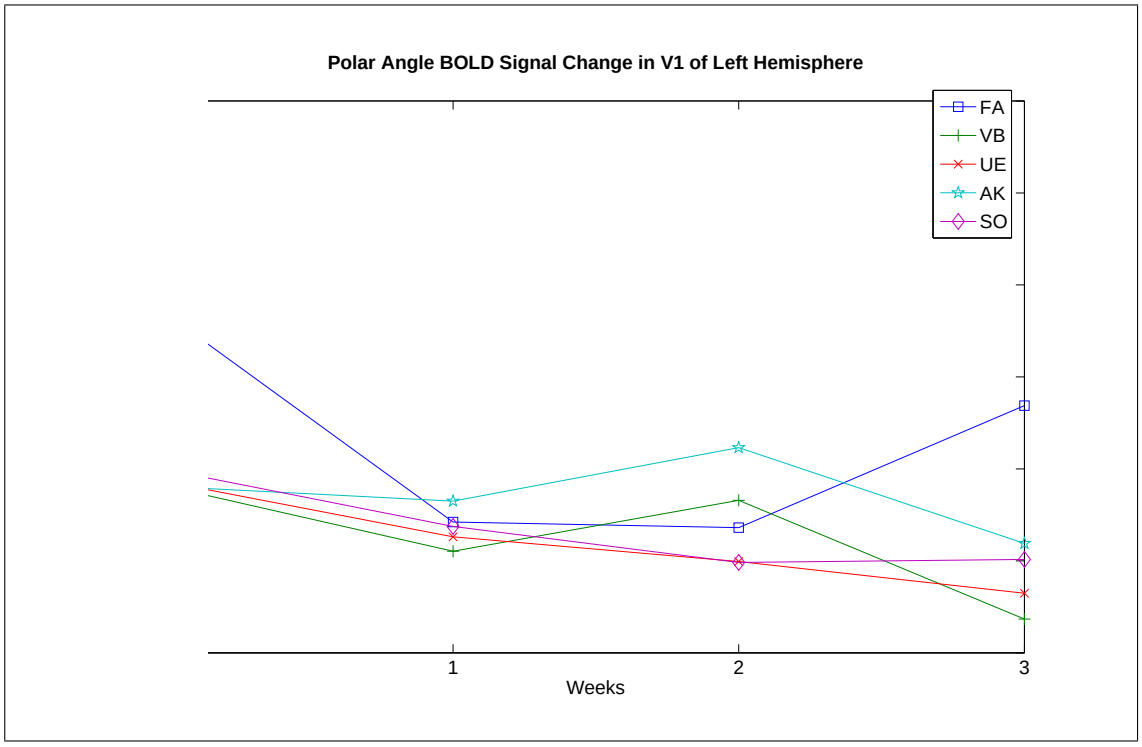

Figure 5.15 BOLD signal change percentage of left hemisphere V1 area due to rotating stimulus

Table 5.4
BOLD signal change percentage of left hemisphere V1 area due to rotating stimulus

	Week #0	Week #1	Week #2	Week #3
Subject FA	0.5677	0.3422	0.3361	0.4687
Subject VB	0.3809	0.3105	0.3658	0.2367
Subject UE	0.3857	0.3260	0.2992	0.2648
Subject AK	0.3808	0.3649	0.4231	0.3189
Subject SO	0.3987	0.3374	0.2982	0.3015

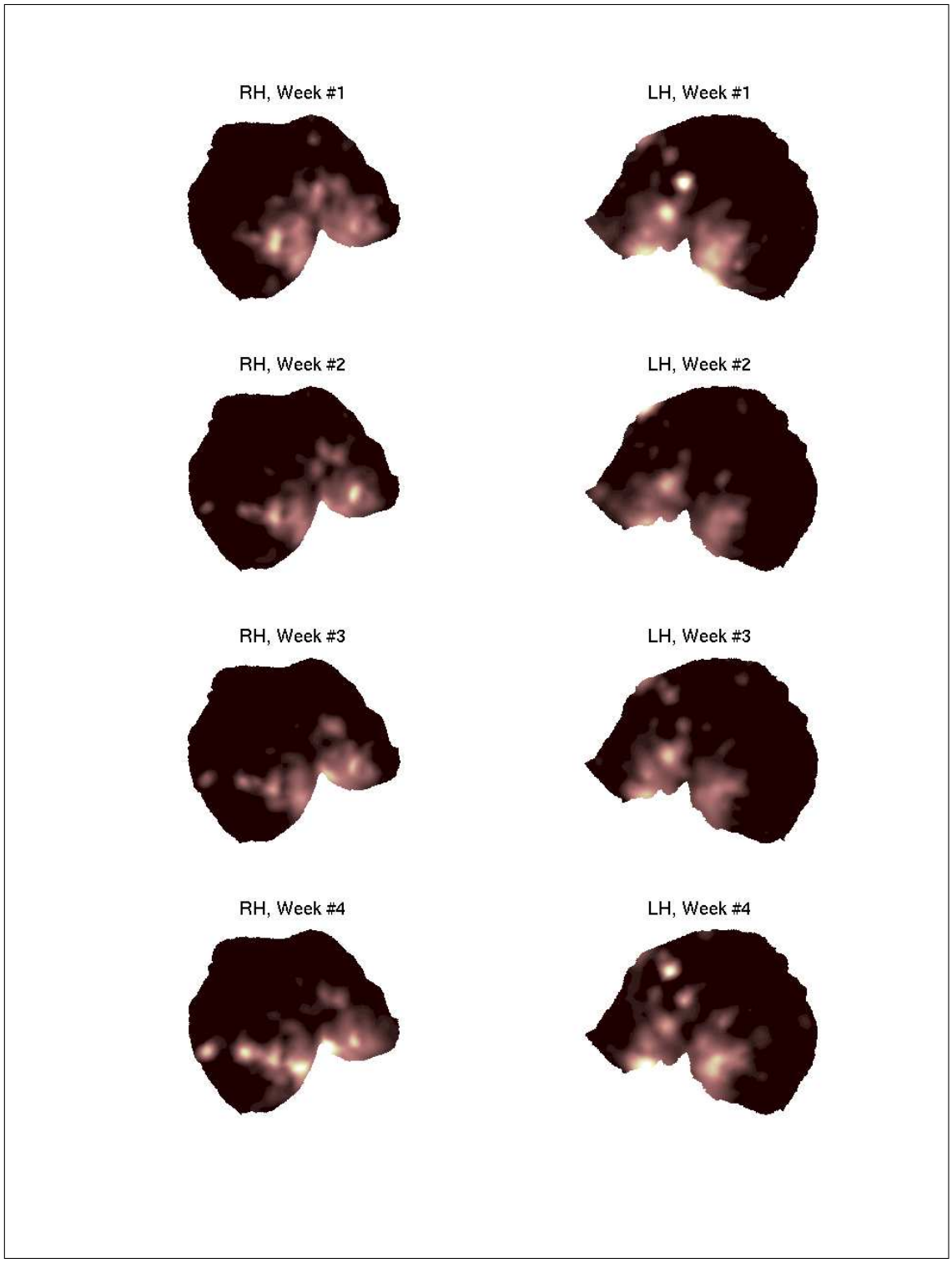

Figure 5.16 % BOLD change due to polar angle stimulus for subject FA

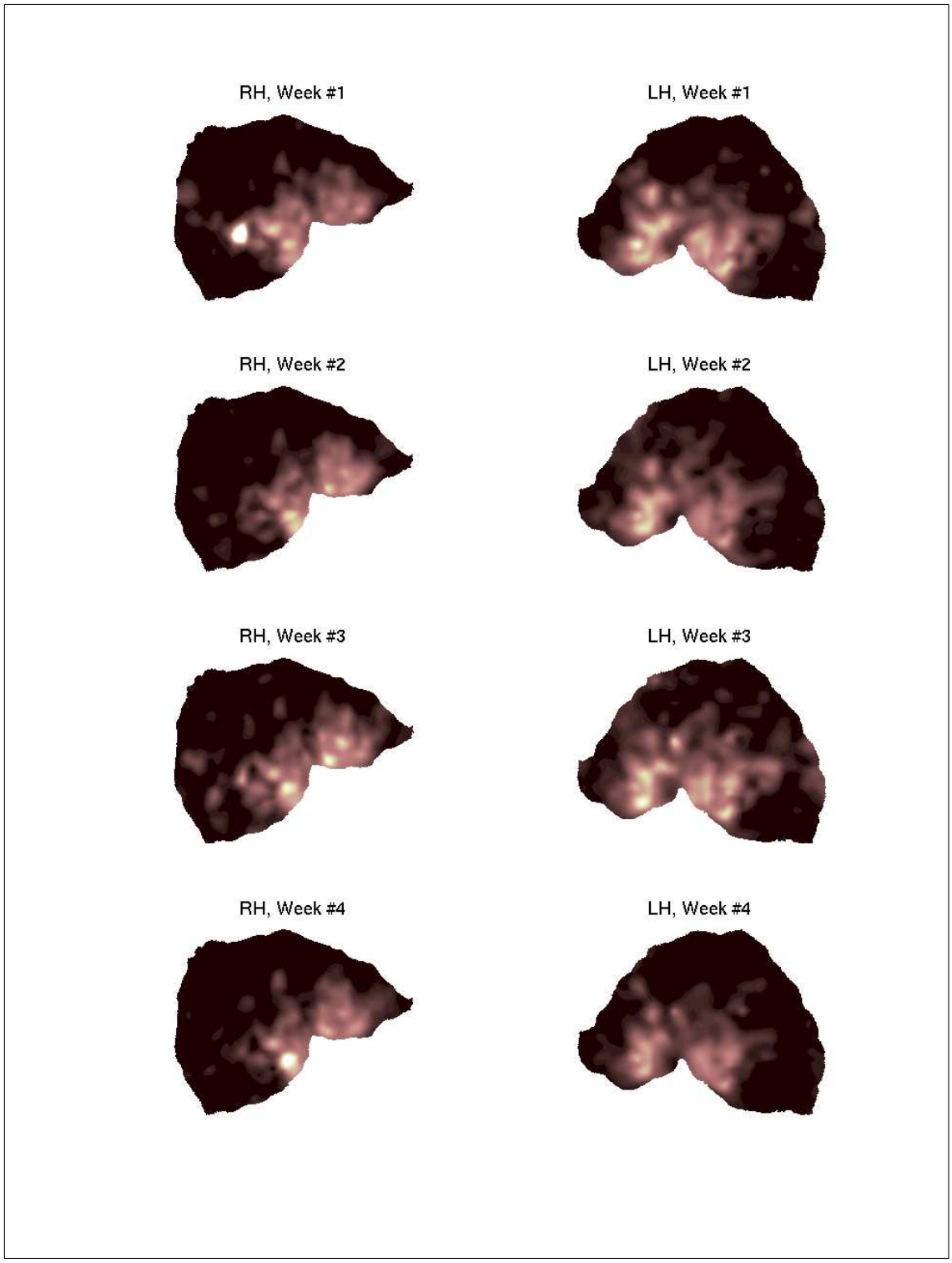

Figure 5.17 % BOLD change due to polar angle stimulus for subject VB

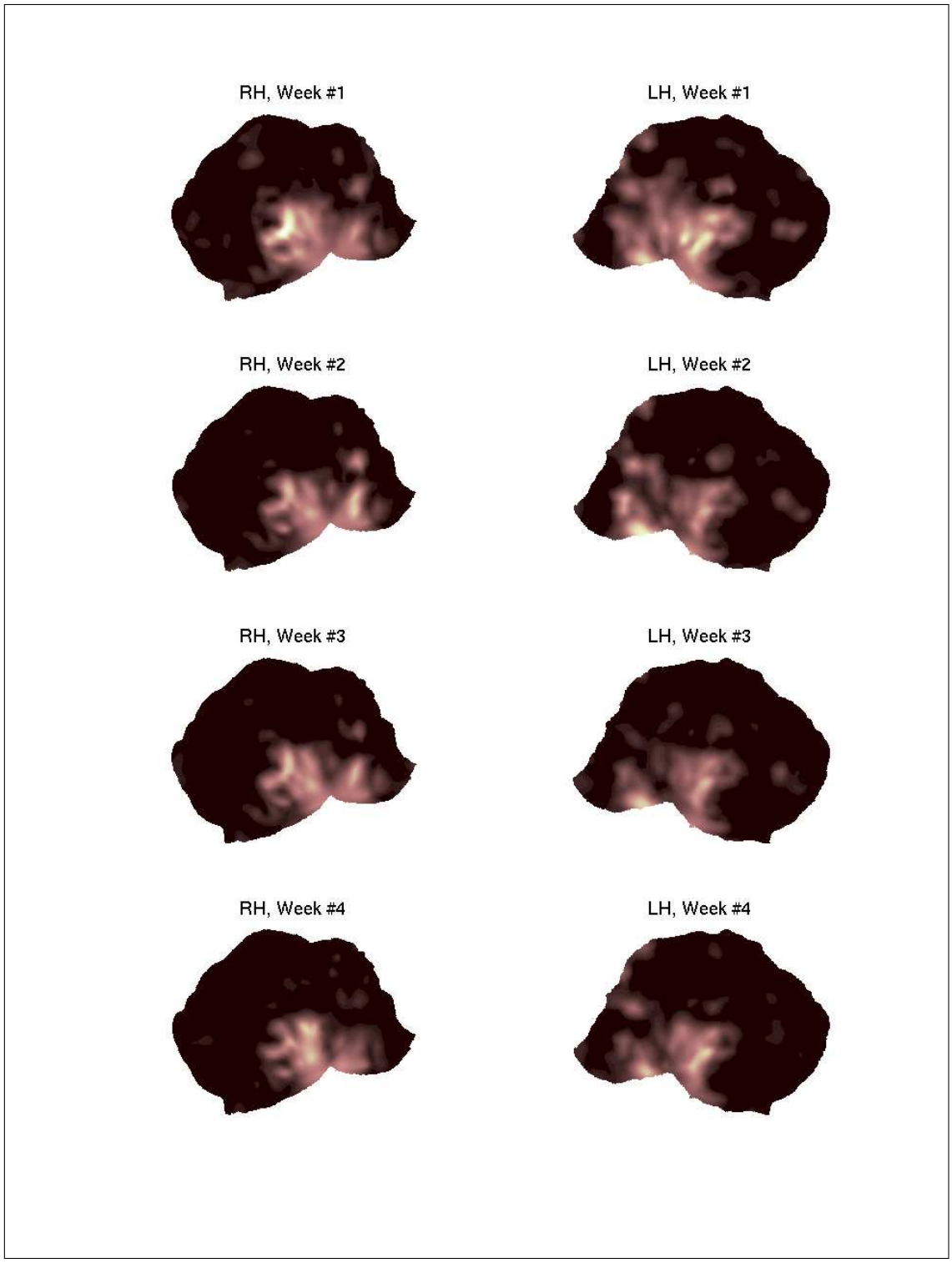

Figure 5.18 % BOLD change due to polar angle stimulus for subject UE

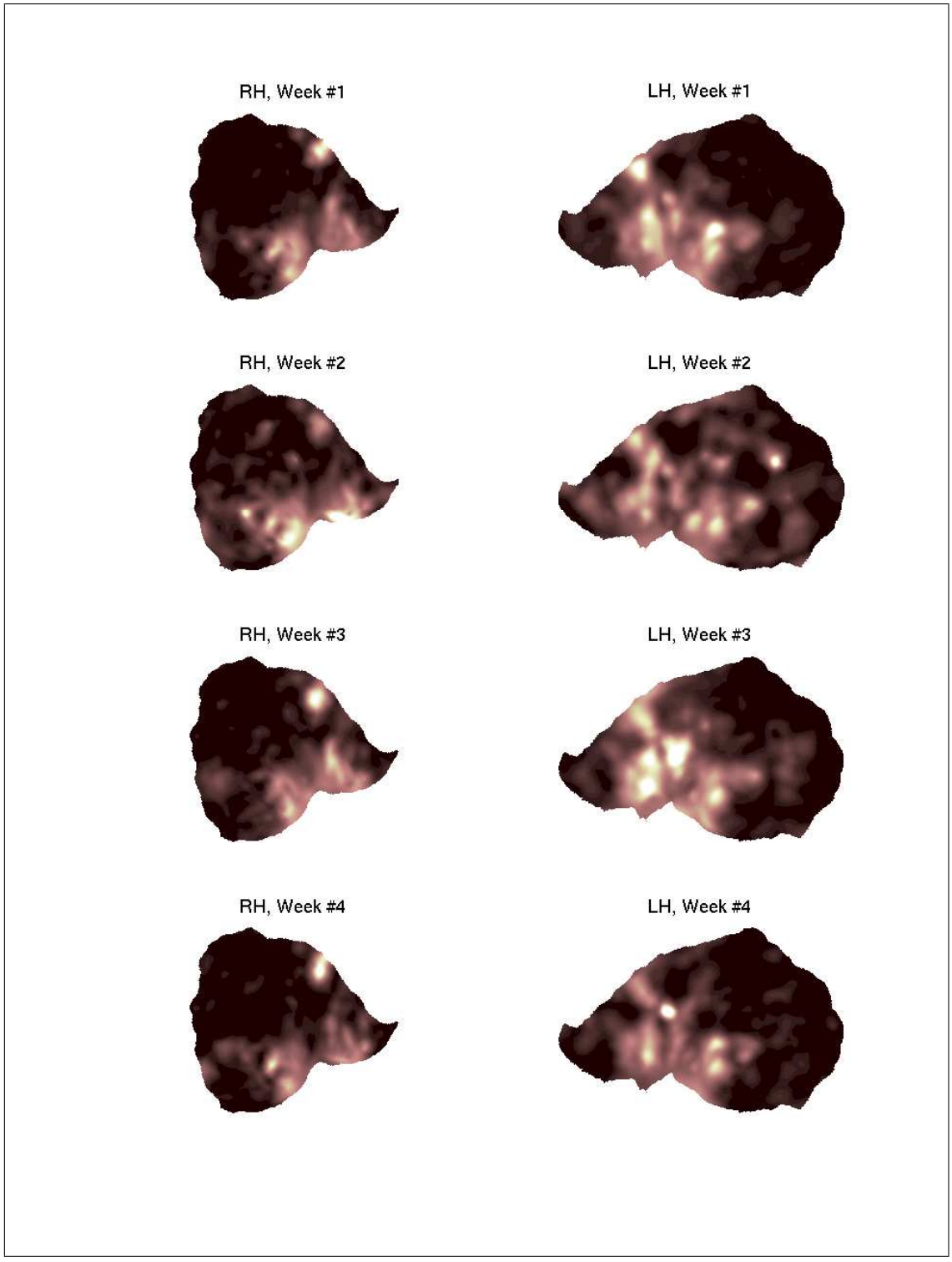

Figure 5.19 % BOLD change due to polar angle stimulus for subject AK

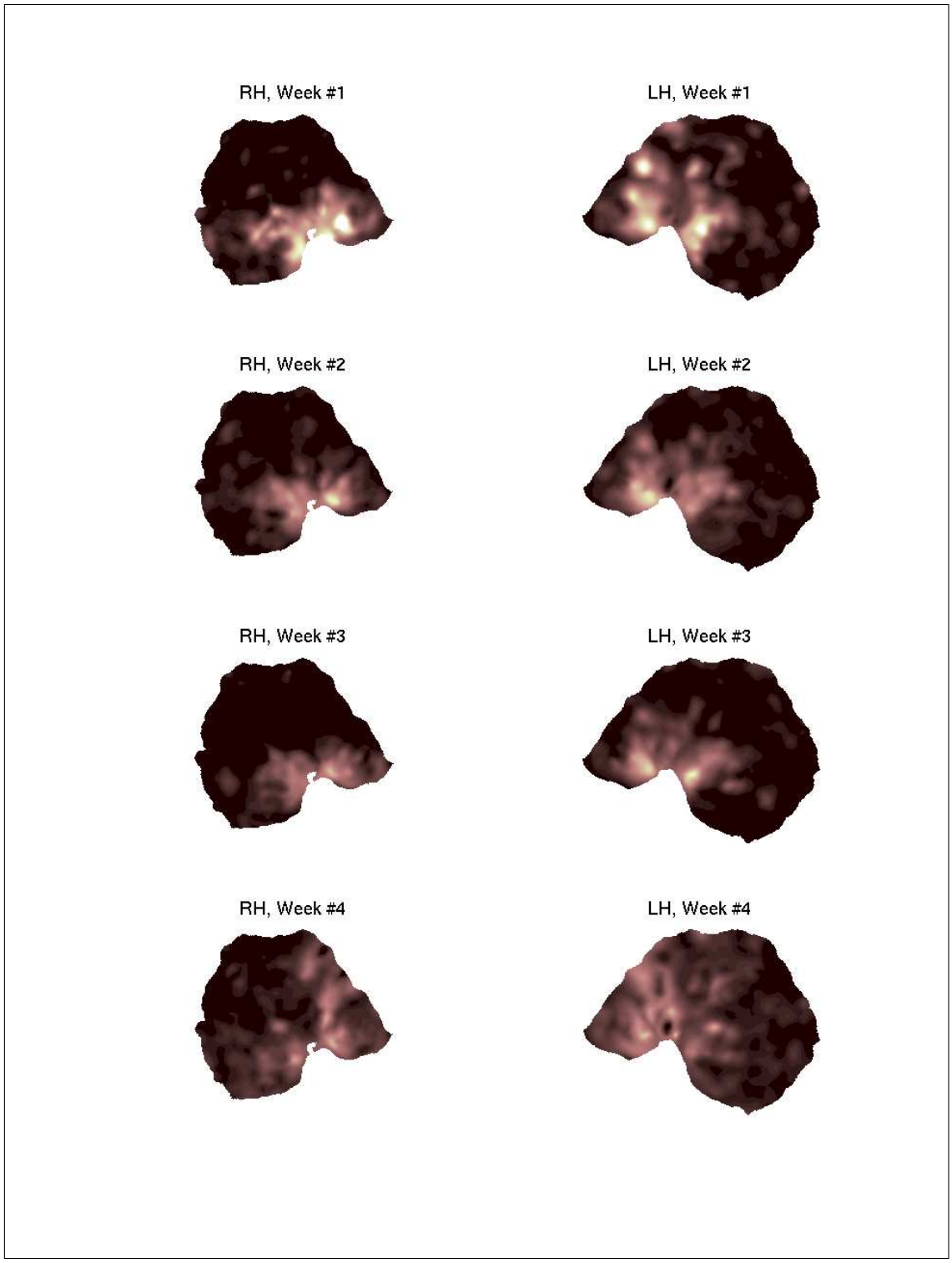

Figure 5.20 % BOLD change due to polar angle stimulus for subject SO

5.3 Cortical Magnification Changes

Cortical magnifications are plotted for both hemispheres of five subjects in figures from 5.21 to 5.30. The degree of eccentricity center is on the x-axis and the relative thickness per degree is on the y-axis. Different series are belong to different scans where 1 indicates the scan before training, 2 indicates the scan after the first week and so on.

Cortical magnification results are also stable which verifies the robustness of the retinotopic fMRI method. However, results are not precise enough to determine the changes due to training. Plots also suffer from the insufficient minimum eccentricity of our experimental setup.

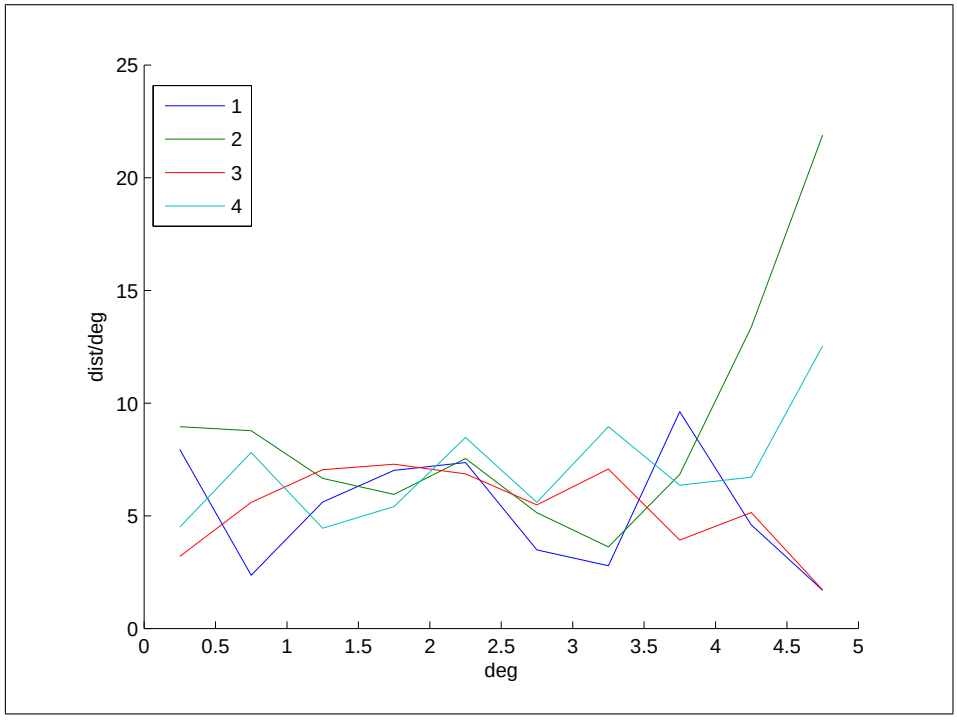

Figure 5.21 Cortical Magnification in V1 of right hemisphere for subject FA

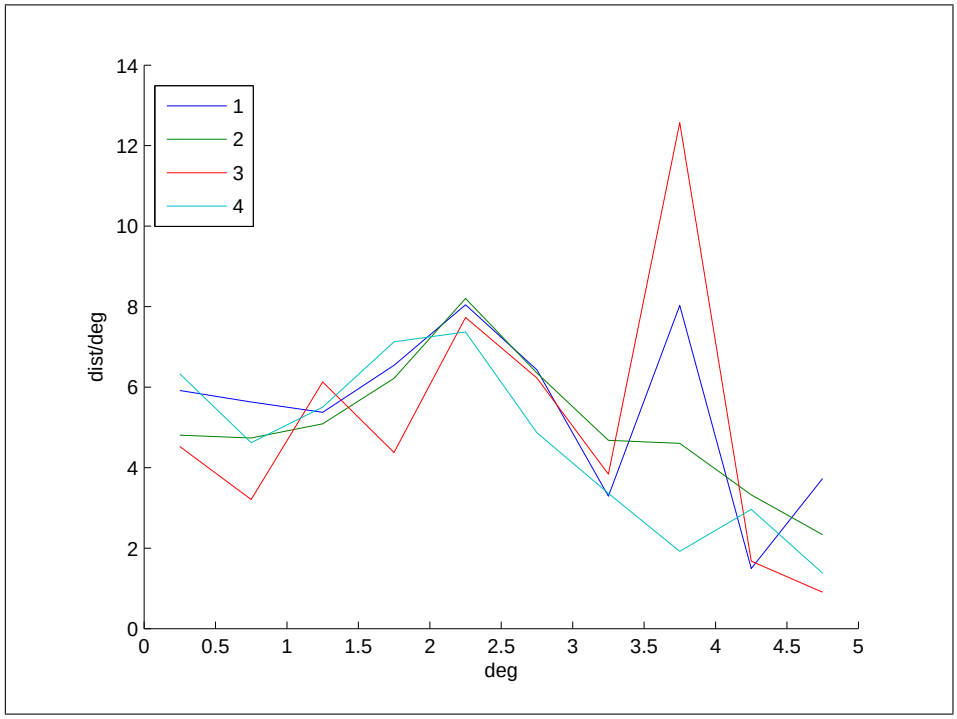

Figure 5.22 Cortical Magnification in V1 of left hemisphere for subject FA

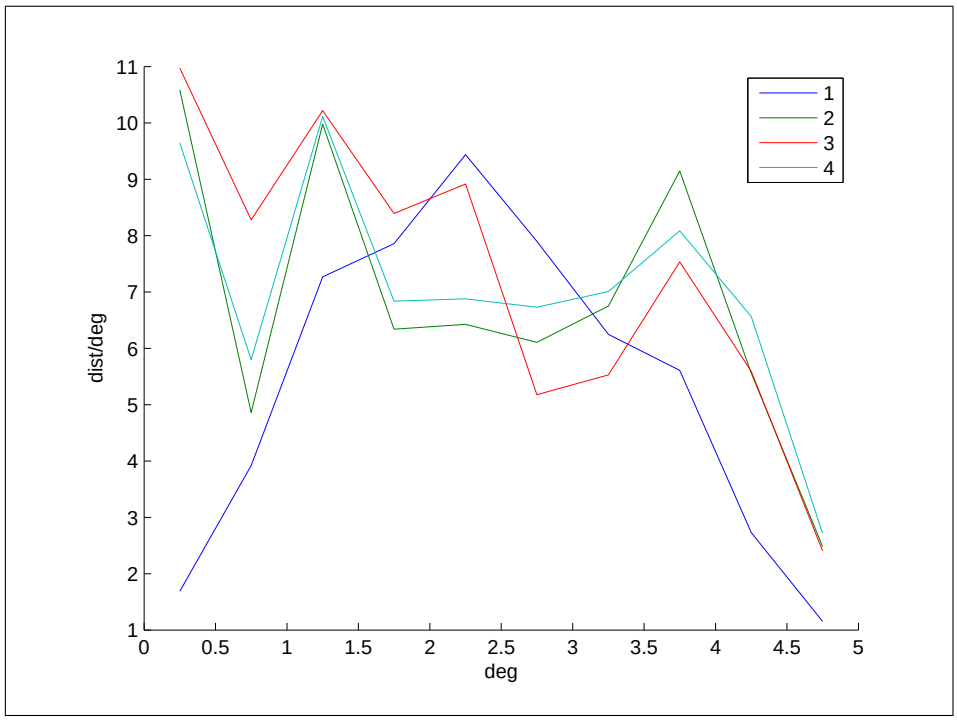

Figure 5.23 Cortical Magnification in V1 of right hemisphere for subject VB

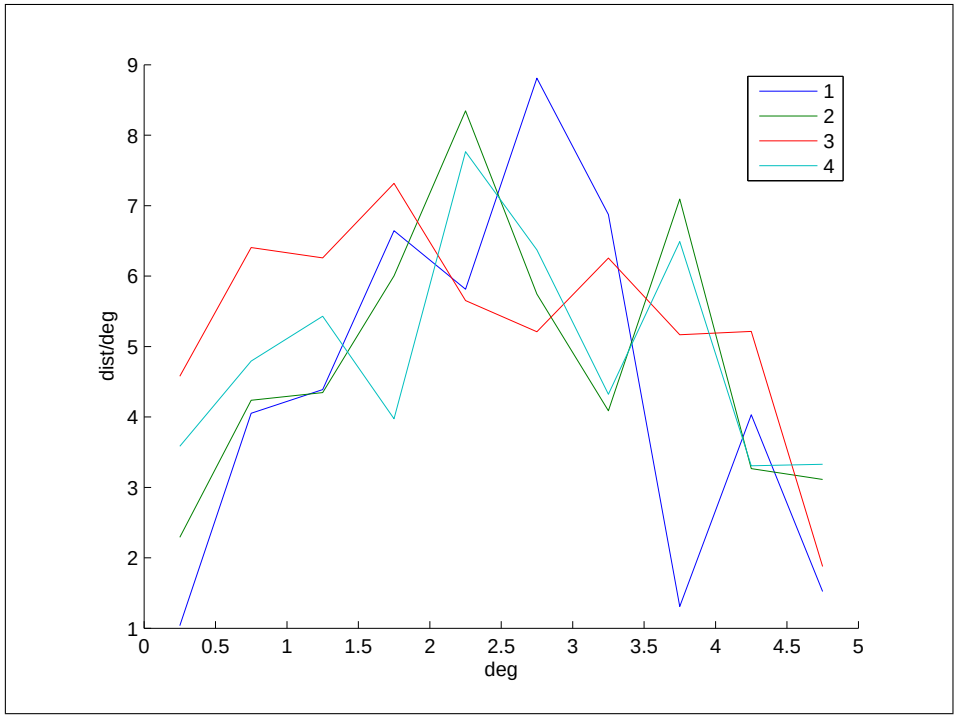

Figure 5.24 Cortical Magnification in V1 of left hemisphere for subject VB

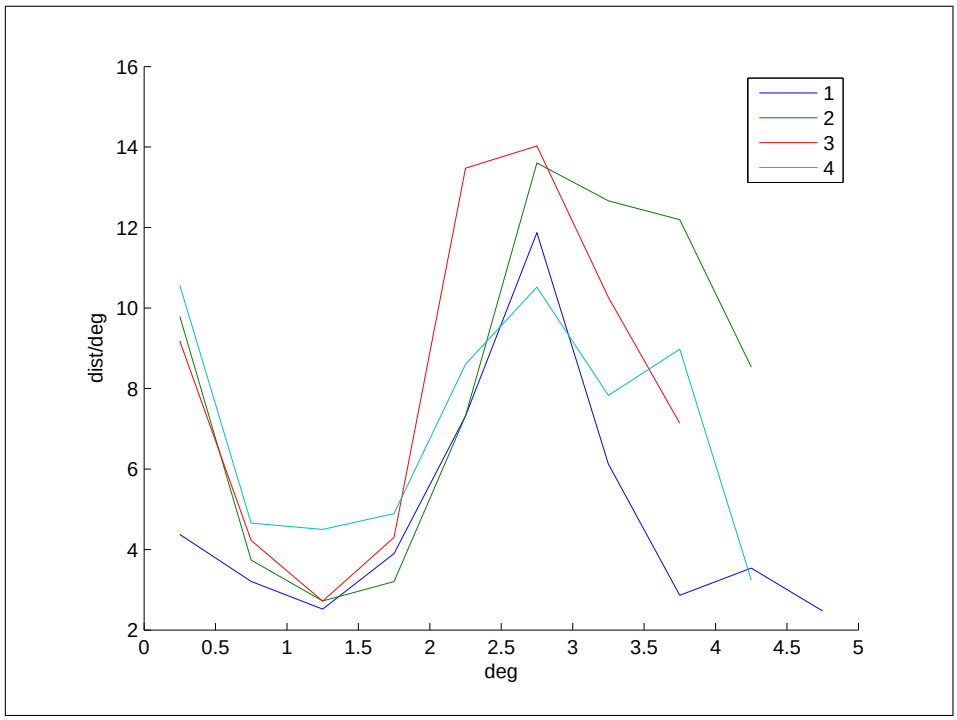

Figure 5.25 Cortical Magnification in V1 of right hemisphere for subject UE

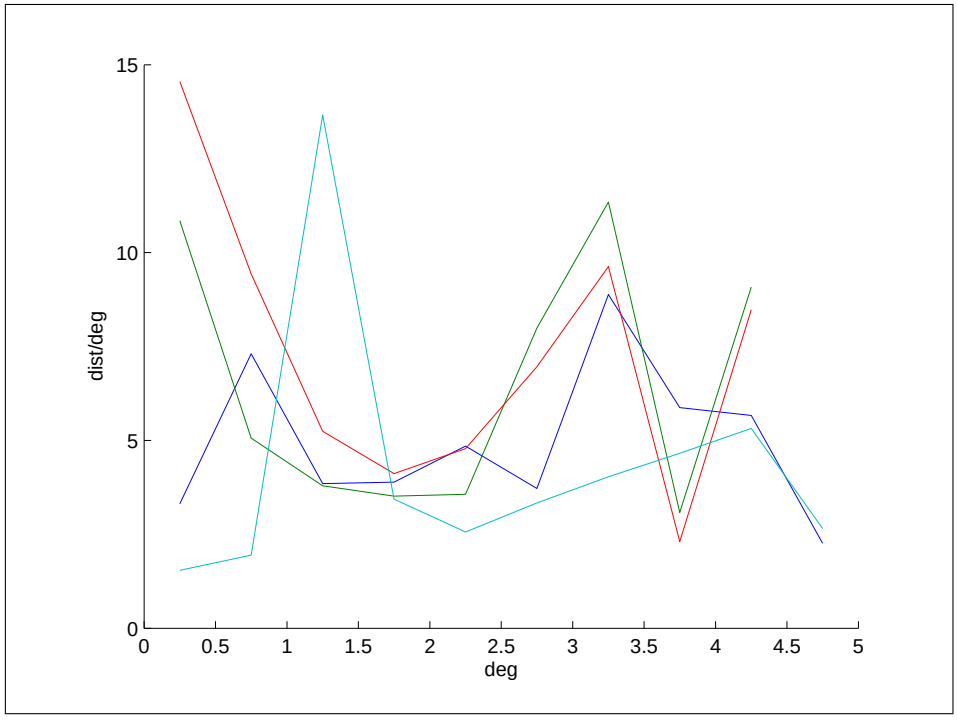

Figure 5.26 Cortical Magnification in V1 of left hemisphere for subject UE

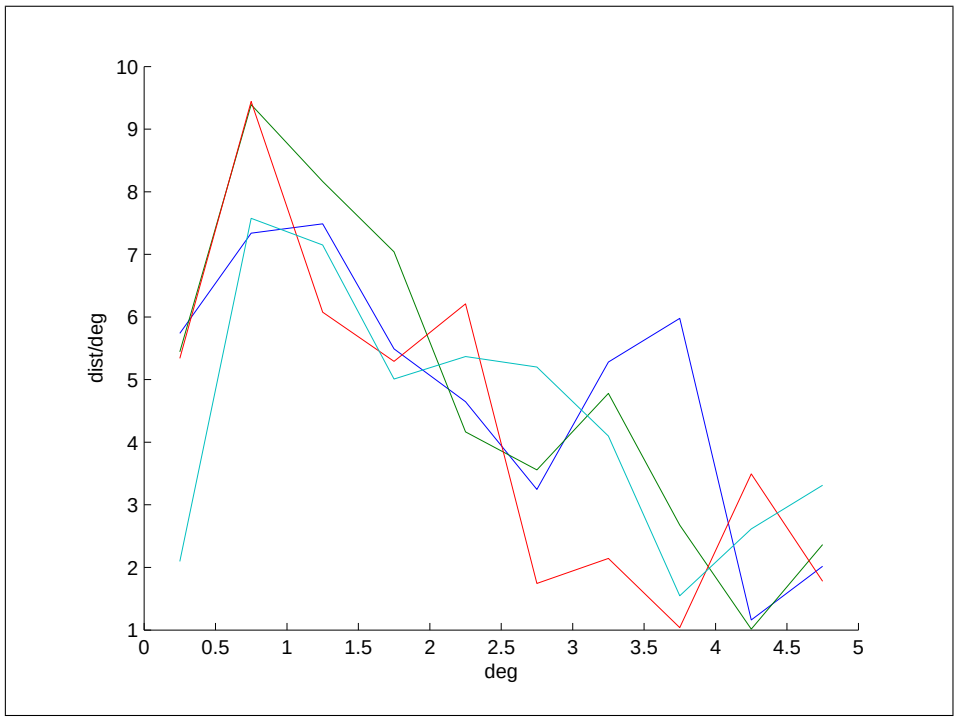

Figure 5.27 Cortical Magnification in V1 of right hemisphere for subject AK

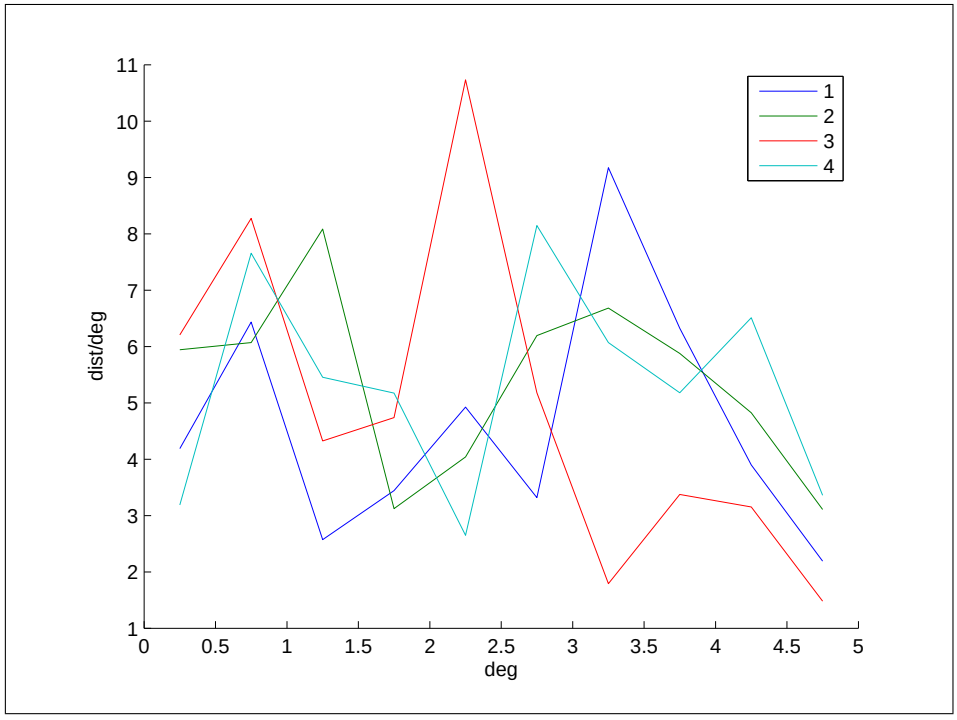

Figure 5.28 Cortical Magnification in V1 of left hemisphere for subject AK

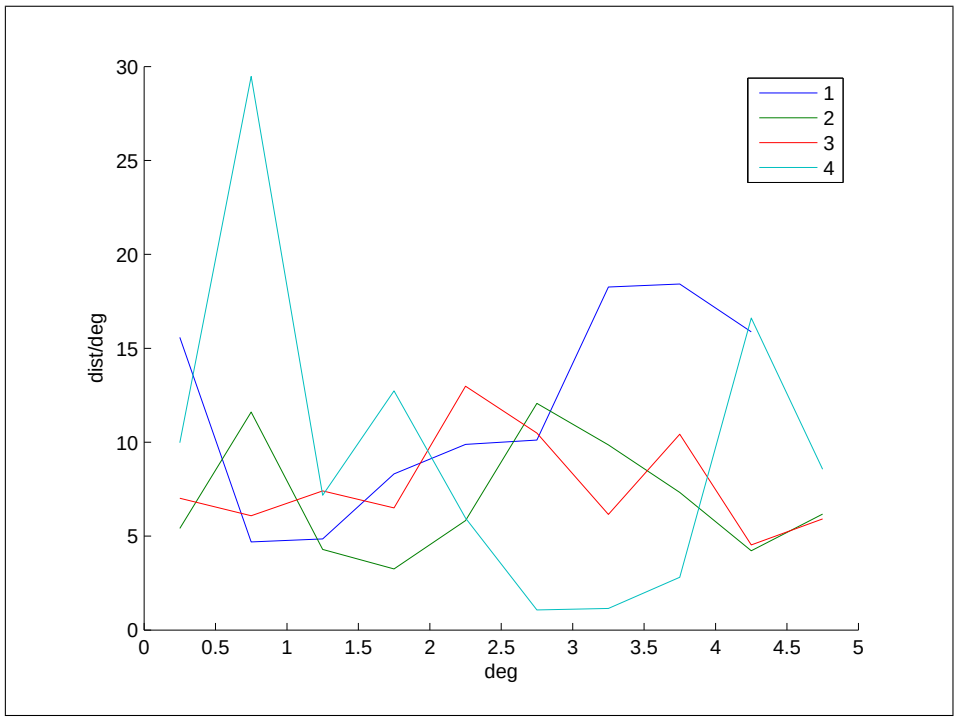

Figure 5.29 Cortical Magnification in V1 of right hemisphere for subject SO

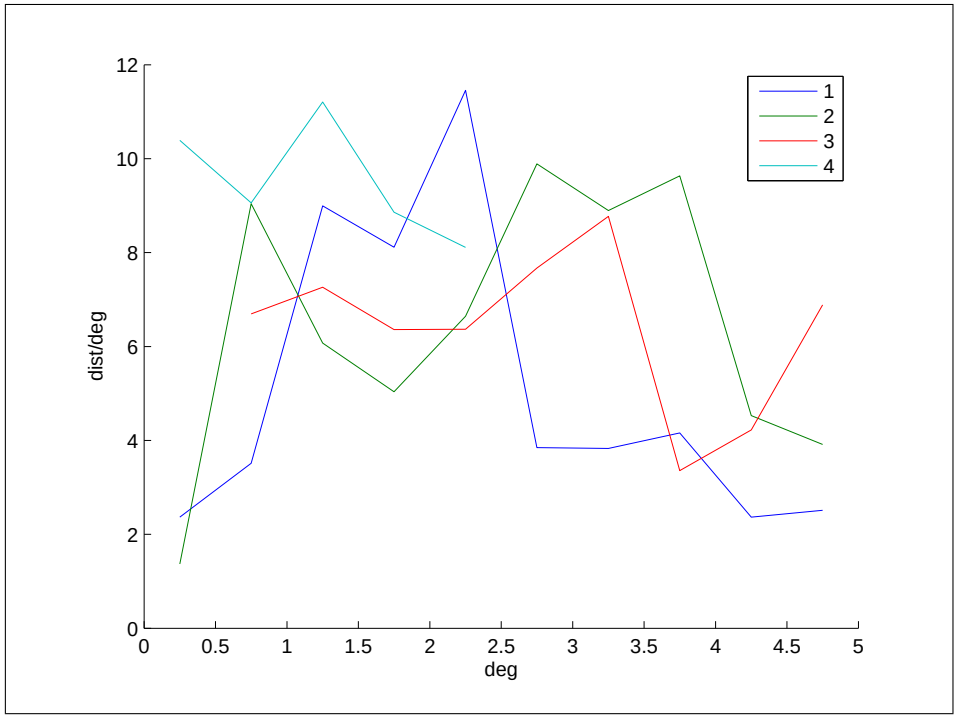

Figure 5.30 Cortical Magnification in V1 of left hemisphere for subject SO

6. DISCUSSION

Visual field sign maps, bold signal change maps, signal change in primary visual cortex and cortical magnification is determined for 5 subjects per week who were trained during 3 weeks.

The visual field sign maps are too noisy to measure and compare the areas weekly. Areas that can be determined are reduced weekly for subjects FA, UE, AK and SO. However, they are quite stable for all four week's scans. This indicates that the registration methods used are robust. Since most of the process is done on the flattened surface, surface generation and flattening affects the obtained maps. Surfaces are generated once for each subject. So the distortion at the surface flattening might cause the fail of determination of the visual areas for all weeks for subject SO and UE.

Without being significant, a decreasing trend in the bold signal change for left hemisphere of all subjects is observed for both stimulus. Weekly results for both stimulus are similar. The reduction of the visual field sign areas can be explained by the decrease in the bold amplitudes. However, it is not easy to say that this reduction is caused by adaptation since factors such as attention and expectation is not controlled during the scans. This is usually done by receiving button press feedbacks from the subject. The bold change at the left hemisphere is more than right hemisphere for subjects VB, UE, AK and SO. This might indicate the dominancy of the left hemisphere.

Cortical magnifications are observed to be stable during 3 weeks. This also indicates the repeatability of the methodology. The maximum eccentricity of the stimulus (5 degrees in current study) might be increased for better quantification of the cortical magnification. The relation between the observed response phase and the position of stimulation in the visual field also depends on the hemodynamic delay. Since the hemodynamic delay may vary as a function of the position on the cortex, it needs to

be measured and corrected for locally [29] by averaging the responses to two stimuli moving in the opposite directions.

Altough a significant change did not observed due to repetitive retinotopic stimulus, there are some findings consistent with the literature. The repeatability of the retinotopic experiments is verified.

7. FUTURE WORK

The retinotopy procedure presented can be improved further by correcting the following points:

Factors, such as attention and expectation, might also contribute to differences in the BOLD signal. Thus it is important to control these factors across conditions which is usually done by receiving button press feedbacks from the subject.

The relation between the observed response phase and the position of stimulation in the visual field also depends on the hemodynamic delay. Since the hemodynamic delay may vary as a function of the position on the cortex, it needs to be measured and corrected for locally. This can be done by averaging the responses to two stimuli moving in opposite directions.

Calculating the visual field signs by fitting a plane to the vertices and neighbours locally, which is computationally much more time consuming, might give a better result than doing on flat surface because of the distortion due to flattening. Another way is to correct the result by the known flattening distortion [4].

Stimulus can be improved by using an eccentricity stimulus that the speed of expansion or contraction varies exponentially with eccentricity which produces a wave of activation on the cortical surface traveling approximately at constant speed. Angle of the rotating wedge might be optimized. The maximum ecccentricity (which is 5 degrees) might be increased.

APPENDIX A. MRI PROTOCOLS

A.1 STRUCTURAL (MPRAGE)

GEOMETRY

Coil selection *SENSE-Head-8*

element selection *SENSE*

connection *d*

FOV (mm) *240*

RFOV (%) *100*

Foldover suppression *no*

Matrix scan *240*

reconstruction *256*

Scan percentage (%) *100*

SENSE *yes*

P reduction (AP) *1*

S reduction (FH) *1*

body tuned *no*

Overcontiguous slices *no*

Stacks *1*

slices *128*

slice thickness *1.2*

foldover direction *AP*

fat shift direction *L*

Chunks *1*

Large table movement *no*

PlanAlign *no*

REST slabs *0*

CONTRAST

Scan mode *3D*

technique *FFE*

Contrast enhancement *T1*

Fast Imaging mode *TFE*

shot mode *multi-shot*

TFE factor *240*

startup echoes *default*

shot interval *user defined*

(ms) *3000*

profile order *linear*

turbo direction *Y*

Echoes *1*

partial echo *no*

shifted echo *no*

TE *shortest*

Flip angle (deg) *8*

TR *shortest*

Half Scan *no*

Water fat shift *user defined*

(pixels) *1.8*

Shim *auto*

SPIR *no*

SPAIR *no*

TFE prepulse *invert*

slice selection *no*

delay *shortest*

ProSet *no*

MTC *no*

T2prep *no*

diffusion mode *no*

SAR mode *high*

B1 mode *default*

PNS mode *low*

gradient mode *maximum*

SofTone mode *no*

MOTION

Cardiac synchronisation *no*

Respiratory compensation *no*

Navigator respiratory *no*

Flow compensation *no*

fMRI echo stabilisation *no*

NSA *1*

DYN/ANG

Angio *no*

Quantitative flow *no*

Manual start *no*

Dynamic study *no*

POST/PROC

Preparation phases *auto*

SENSE ref.scan *no*

MIP/MPR *no*

Autoview image *M*

Reference tissue *Grey matter*

Preset window cont. *soft*

Reconstruction mode *real time*

Save raw data *no*

Hardcopy protocol *no*

Ringing filtering *yes*

A.2 FUNCTIONAL (GE-EPI)

GEOMETRY

Coil selection *SENSE-Head-8*

element selection *SENSE*

connection *d*

Homogenity correction *none*

CLEAR *yes*

body tuned *no*

FOV (mm) *230*

RFOV (%) *100*

Foldover suppression *no*

Matrix scan *64*

reconstruction *64*

Scan percentage (%) *100*

SENSE *no*

Stacks *1*

type *parallel*

slices *30*

slice thickness *3.2*

slice gap *user defined*

gap (mm) *0*

slice orientation *transverse*

foldover direction *RL*

fat shift direction *L*

Minimum number. *1*

Slice scan order *interleaved*

Large table movement *no*

PlanAlign *no*

REST slabs *0*

CONTRAST

Scan mode *MS*

technique *FFE*

Contrast enhancement *no*

Fast Imaging mode *EPI*

shot mode *single shot*

Echoes *1*

partial echo *no*

shifted echo *no*

TE *user defined*

(ms) *30*

Flip angle (deg) *90*

TR *user defined*

(ms) *2000*

Half Scan *no*

Water fat shift *minimum*

Shim *auto*

SPIR *fat sup.*

frequency offset *default*Â

SPAIR *no*

ProSet *no*

MTC *no*

diffusion mode *no*

SAR mode *high*

B1 mode *default*

PNS mode *moderate*

gradient mode *maximum*

SofTone mode *no*

MOTION

Cardiac synchronisation *no*

Respiratory compensation *no*

Navigator respiratory *no*

Flow compensation *no*

Temporal slice spacing *equidistant*

fMRI echo stabilisation *no*

NSA *1*

DYN/ANG

Angio *no*

Quantitative flow *no*

Manual start *yes*

Dynamic study *individual*

dyn scans *256*

dyn scan times *shortest*

dummy scans *0*

immediate subtraction *no*

fast next scan *no*Â

synch.ext.device *no*

dyn.stabilization *yes*

Keyhole *no*Â

POST/PROC

Preparation phases *full*

SENSE ref.scan *no*

MIP/MPR *no*

Autoview image *M*

Reference tissue *Grey matter*

Preset window cont. *soft*

Reconstruction mode *real time*

Save raw data *no*

Hardcopy protocol *no*

APPENDIX B. FREESURFER MATLAB TOOLBOX

B.1 FSTBX_Read_Surface

```
function FSTBX_Surface = FSTBX_Read_Surface(Surface_FileName)
%
% Usage: FSTBX_Surface = FSTBX_Read_Surface(Surface_FileName)
%
% Input:
%   Surface_FileName        Name of the surface file.
%
% Output:
%   FSTBX_Surface           FSTBX Surface structure.
%       .Faces              Vertex indices of faces. (nFx3)
%       .Vertices           Coordinates of vertices. (nVx3)
%       .NumberOfVertices   Number of vertices. (1x1)
%       .IsFullSurface      Always true. (bool)
%
% Reads the vertex coordinates and face lists from a surface file. Vertex
% indices are unique ID's for surfaces, patches and w files given in RAS
% coordinates. Face list gives the vertice indices that define a face.
%
% Supported Extentions:
%       .graymid .inflated .orig .white .smoothwm .pial
%
% Example:
%       FullInflatedSurface = FSTBX_Read_Surface('rh.inflated');
%


% Read the surface file
```

```
[FSTBX_Surface.Vertices, FSTBX_Surface.Faces] = read_surf(Surface_FileName);

% Convert faces to matlab convention (Starting with 1)
FSTBX_Surface.Faces = int32(FSTBX_Surface.Faces + 1);

% Set the number of vertices
FSTBX_Surface.NumberOfVertices = int32(size(FSTBX_Surface.Vertices,1));

% Set .IsFullSurface true. This value is false for patches.
FSTBX_Surface.IsFullSurface = true;

% Check if it is a triangular surface
if (size(FSTBX_Surface.Faces,2) ~= 3)
    error('Only Triangular surfaces are supported');
end
```

B.2 FSTBX_Read_Patch

```
function FSTBX_Patch = FSTBX_Read_Patch(Patch_FileName, FSTBX_Surface)
%
% Usage: FSTBX_Patch = FSTBX_Read_Patch(Patch_FileName, FSTBX_Surface)
%
% Input:
%   Patch_FileName          Name of the patch file.
%   FSTBX_Surface           Full surface structure that the patch is
%                           created.
%
% Output:
%   FSTBX_Patch             FSTBX Patch structure.
%       .Faces                          Vertex numbers of faces. (nFx3)
%       .Vertices                       Coordinates of vertices. (nVx3)
```

```
%       .NumberOfVertices               Number of vertices. (1x1)
%       .IsFullSurface                  Always false. (bool)
%       .VertexIdx                      Index of patch vertices to surface
%                                       vertices. (nVx1)
%       .VertexIsOnTheBorder            True if the vertex is on the border
%                                       of the patch. (nVx1)
%       .NumberOfFullSurfaceVertices    nV of Full Surface which the patch
%                                       is derived from. (1x1)
%       .Lookup                         Inverse of VertexIdx. Index of
%                                       surface vertices to patch surfaces.
%                                       (nFullSurfV x 1)
%       .LookupMask                     True if the Full Surface vertex is
%                                       defined in patch. (nFullSurfV x 1)
%       .IsFlat                         True if this is a 2d Flat Patch.
%
% A patch is a subset of a full surface. Vertex coordinates are redefined in
% patch files, so the shape of the patch might be different of surface
% (like flat patches). Reads the vertex coordinates and vertex indicies
% from a patch file and calculates other parameters. VertexIdx gives the
% corresponding vertex index of full surface. It is also a surface with
% faces, vertices and needed information for data manuplation. So this can
% be used as a surface in some function like FSTBX_Paint,
% FSTBX_Read_Value etc.
%
% Supported Extentions:
%       .patch.flat .patch.3d
%
% Example:
%       FlatPatch = FSTBX_Read_Patch('rh.occip.patch.flat',FullSurface);
%


% Check if the input surface is a full surface
```

```
if (FSTBX_Surface.IsFullSurface == false)
    error('Input surface should be a full surface.');
end

% Read the patch file
[FSTBX_Patch.VertexIdx, FSTBX_Patch.Vertices] = read_patchfile(Patch_FileName);

% Set the on the border mask (True : Corresponding vertex is on the border)
FSTBX_Patch.VertexIsOnTheBorder = (sign(FSTBX_Patch.VertexIdx)==1);

% Set logic value if the patch is a 2D flat surface
FSTBX_Patch.IsFlat = false;
if ( ( min(FSTBX_Patch.Vertices(:,3)) == 0 ) &&
     ( max(FSTBX_Patch.Vertices(:,3)) == 0 ) )
    FSTBX_Patch.IsFlat = true;
end

% We don't need the sign of Vertex Indices since this information is in
% .VertexIsOnTheBorder Mask
FSTBX_Patch.VertexIdx = int32( abs(FSTBX_Patch.VertexIdx));

% Set the number of vertices
FSTBX_Patch.NumberOfVertices = int32(size(FSTBX_Patch.Vertices,1));

% Set the number of full surface vertices
FSTBX_Patch.NumberOfFullSurfaceVertices = FSTBX_Surface.NumberOfVertices;

% Create faces for the patch using the full surface faces
% Create a Mask of used faces of full surface
Mask1 = ismember(FSTBX_Surface.Faces(:,1),FSTBX_Patch.VertexIdx);
Mask2 = ismember(FSTBX_Surface.Faces(:,2),FSTBX_Patch.VertexIdx);
Mask3 = ismember(FSTBX_Surface.Faces(:,3),FSTBX_Patch.VertexIdx);
```

```
Mask = Mask1 & Mask2 & Mask3;

% Find the indices of used faces
[row,col] = find(Mask);

% Get the used faces
FSTBX_Patch.Faces = FSTBX_Surface.Faces(row(:),:);

% Create a lookup table (inverse of VertexIdx)
FSTBX_Patch.Lookup = int32(zeros(FSTBX_Surface.NumberOfVertices,1));
for (i=1:FSTBX_Patch.NumberOfVertices)
    FSTBX_Patch.Lookup(FSTBX_Patch.VertexIdx(i,1)) = i;
end

% True if the full surface vertices are used in patch
FSTBX_Patch.LookupMask = (FSTBX_Patch.Lookup~=0);

% Convert surface faces to patch faces
FSTBX_Patch.Faces(:,1) = FSTBX_Patch.Lookup(FSTBX_Patch.Faces(:,1),1);
FSTBX_Patch.Faces(:,2) = FSTBX_Patch.Lookup(FSTBX_Patch.Faces(:,2),1);
FSTBX_Patch.Faces(:,3) = FSTBX_Patch.Lookup(FSTBX_Patch.Faces(:,3),1);

% False since this is a part of a full surface
FSTBX_Patch.IsFullSurface = false;
```

B.3 FSTBX_Read_Value

```
function FSTBX_Value = FSTBX_Read_Value(Values_FileName,
                                        FSTBX_Surface,
                                        fstbx_ValueFormat)
%
```

```
% Usage: FSTBX_Value = FSTBX_Read_Value(Values_FileName,
%                                        FSTBX_Surface,
%                                        fstbx_ValueFormat)
%
% Input:
%   Values_FileName     Name of the value file. It can be one of the
%                       fstbx_ValueFormat type file.
%   FSTBX_Surface       Surface or any Patch that the values correspond to.
%                       Uses only the number of vertices in full surface
%                       which is also included by any derived patch. So
%                       giving a patch or surface is indifferent.
%   fstbx_ValueFormat   The string of Value file format.
%                           PerVertex, PV (.sulc .curv .thickness
%                                          .defect_status .defect_labels)
%                           W (.w)
%                           FS (.fs .fm)
%
% Output:
%   FSTBX_Value         FSTBX Value format
%       .Values             The values read. Same indices as full surface
%                           vertices. (nVFull x 1)
%       .ValueIsSetMask     True if the value for the vertex is available.
%                           Value is 0 for false fields. (nVFull x 1)
%       .ValueFormat        PerVertex, W, FS or Processed. (string)
%
% Supported Extentions:
%       .sulc .curv .thickness .defect_status .defect_labels .fs .fm .w
%
% Example:
%       Curvature = FSTBX_Read_Value('rh.curv',FullSurface,'PerVertex');
%
```

```
% Get the number of vertices in full surface
if (FSTBX_Surface.IsFullSurface)
    nFullVertices = FSTBX_Surface.NumberOfVertices;
else
    nFullVertices = FSTBX_Surface.NumberOfFullSurfaceVertices;
end

switch (fstbx_ValueFormat)
    case {'PerVertex','PV'}
        FSTBX_Value.Values = read_curv(Values_FileName);
        FSTBX_Value.ValueIsSetMask = true(nFullVertices,1);
        FSTBX_Value.ValueFormat = 'PerVertex';

    case 'W'
        [W_Values,Vertex] = read_wfile(Values_FileName);
        Vertex = int32(Vertex + 1);
        FSTBX_Value.Values = zeros(nFullVertices,1);
        FSTBX_Value.Values(Vertex(:),1) = W_Values(:,1);
        FSTBX_Value.ValueIsSetMask = false(nFullVertices,1);
        FSTBX_Value.ValueIsSetMask(Vertex(:),1) = true;
        FSTBX_Value.ValueFormat = 'W';

    case 'FS'
        FSTBX_Value.Values = read_fs(Values_FileName);
        FSTBX_Value.ValueIsSetMask = true(nFullVertices,1);
        FSTBX_Value.ValueFormat = 'FS';

    otherwise
        error('Value Format is unknown.');
end
```

B.4 FSTBX_Paint

```
function Handle = FSTBX_Paint(FSTBX_Surface, FSTBX_Value)
%
% Usage: Handle = FSTBX_Paint(FSTBX_Surface, FSTBX_Value)
%
% Input:
%   FSTBX_Surface Surface to paint. Might be full or patch.
%   FSTBX_Value Vertex data for coloring the surface.
%
% Output:
%   Handle Handle to the created figure object for format
% manupilations
%
% Example:
%       H = FSTBX_Paint(FullSurface, CurvValue);
%

% Check if it is a full surface. Otherwise determine the values
% corresponding to the patch vertices
if (FSTBX_Surface.IsFullSurface == false)
    Values = FSTBX_Value.Values(FSTBX_Surface.VertexIdx(:));
else
    Values = FSTBX_Value.Values;
end

% Draw the surface.
Handle  = figure('Color','white'), patch('Vertices',FSTBX_Surface.Vertices,
'Faces',FSTBX_Surface.Faces,'FaceVertexCData',Values,'FaceColor','interp',
'EdgeColor','none');

axis image;
```

```
axis off;
colormap Hot;
colorbar;
```

B.5 FSTBX_Value2MeshData

```
function FSTBX_MeshData = FSTBX_Value2MeshData(FSTBX_Value, FSTBX_Patch,
                                                StepSize, method)
%
% Usage: FSTBX_MeshData = FSTBX_Value2MeshData(FSTBX_Value, FSTBX_Patch,
%                                                StepSize, method)
%
% Input:
%   FSTBX_Value         Vertex Data.
%   FSTBX_Patch         2D flat patch.
%   StepSize Step size of the regular grid.
%   method String of interpolation method. Available
%                       options are: 'linear','cubic'. Check griddata
%                       function of matlab for further information
%
% Output:
%   FSTBX_MeshData FSTBX Mesh Data format
%       .X X Coordinates of the regular grid
%       .Y Y Coordinates of the regular grid
%       .Values         Values of the regular grid
%       .ValueIsSetMask 1 if value is set by function, 0 otherwise.
%       .Range          Minimum and maximum values for X and Y.
%       .StepSize       Step size of the grid
%
% This function creates a regular grid at the range of flat patch
% coordinates sampled in step size. Vertex values of the patch are
```

```
% interpolated to the grid. Regular grid data is very useful to do
% operations on the data (such as gradient calculation or pixel
% operations)
%
% Example:
% MeshData = FSTBX_Value2MeshData(V_ecc, FlatPatch, 0.5, 'cubic')
%

if (FSTBX_Patch.IsFullSurface)
    error('FSTBX_Patch can not be a full surface');
end

if (FSTBX_Patch.IsFlat == false)
    error('Input patch should be a flat 2d patch')
end

Values = FSTBX_Value.Values(FSTBX_Patch.VertexIdx(:));

% Determine the range for grid and construct the grid
Max = ceil(max(max(FSTBX_Patch.Vertices)));
Min = floor(min(min(FSTBX_Patch.Vertices)));

% Create a regular grid
[FSTBX_MeshData.X,FSTBX_MeshData.Y] = meshgrid(Min:StepSize:Max,
                                               Min:StepSize:Max);

% Calculate Values corresponding to the grid coordinates by linear
% interpolation
FSTBX_MeshData.Values = griddata(FSTBX_Patch.Vertices(:,1),
                                 FSTBX_Patch.Vertices(:,2),
                                 Values,
                                 FSTBX_MeshData.X,
```

```
                                    FSTBX_MeshData.Y,
                                    method);

% Create a Mask of the values that are set by the griddata
FSTBX_MeshData.ValueIsSetMask = ~isnan(FSTBX_MeshData.Values);

% Give a 0 value to the data that are not set by griddata
[i,j] = find(~FSTBX_MeshData.ValueIsSetMask);
ind = sub2ind(size(FSTBX_MeshData.Values),i,j);
FSTBX_MeshData.Values(ind)=0;

%Save the range
FSTBX_MeshData.Range = [Min Max];

% Save the stepsize
FSTBX_MeshData.StepSize = StepSize;
```

B.6 FSTBX_MeshData2Value

```
function FSTBX_Value = FSTBX_MeshData2Value(FSTBX_MeshData,
                                            FSTBX_Patch, method)
%
% Usage: FSTBX_Value = FSTBX_MeshData2Value(FSTBX_MeshData,
%                                            FSTBX_Patch, method)
%
% Input:
%   FSTBX_MeshData      Structure of regular grid data.
%   FSTBX_Patch         2D flat patch.
%   method String of interpolation method. Available
%                       options are: 'linear','cubic'. Check griddata
%                       function of matlab for further information
```

```
%
% Output:
%   FSTBX_Value FSTBX Value format. Grid data resampled at
%                       vertex coordinates.
%
% This function resamples the data on regular grid at the flat patch
% coordinates. Inverse of the FSTBX_Value2MeshData function. Typical
% usage is to create a mesh data structure of the vertex values using
% FSTBX_Value2MeshData function, then do operations on the Values field.
% And again retun to the Value structure using this function.
%
% Example:
% MeshData = FSTBX_Value2MeshData(V_ecc, FlatPatch, 0.5, 'cubic');
% some_function(MeshData);
% V_ecc = FSTBX_MeshData2Value(MeshData, FlatPatch, 'cubic');

% Re-calculate the values corresponding to the patch vertices by
% interpolating the mesh data
PatchValues = interp2(FSTBX_MeshData.X,FSTBX_MeshData.Y,
                      FSTBX_MeshData.Values,FSTBX_Patch.Vertices(:,1),
                      FSTBX_Patch.Vertices(:,2),method);

% Re-calculate the full surface values using the patch values and patch
% vertex numbers
FSTBX_Value.Values = zeros(FSTBX_Patch.NumberOfFullSurfaceVertices,1);
FSTBX_Value.Values(FSTBX_Patch.VertexIdx(:)) = PatchValues(:);

FSTBX_Value.ValueIsSetMask = FSTBX_Patch.LookupMask;

FSTBX_Value.ValueFormat = 'Processed';
```

B.7 FSTBX_NewMeshData

```
function FSTBX_MeshData = FSTBX_NewMeshData(FSTBX_Patch, StepSize)
%
% Usage: FSTBX_MeshData = FSTBX_NewMeshData(FSTBX_Patch, StepSize)
%
% Input:
%   FSTBX_Patch         2D flat patch.
%   StepSize Step size of the new regular grid.
%
% Output:
%   FSTBX_MeshData FSTBX Mesh data format. Empty regular grid
%                       data structure with values set to 0.
%
% This function creates an empty regular grid at the range of given
% Patch. The values of the regular grid can be manipulated and
% converted to the Value format using FSTBX_MeshData2Value function.
%
% Example:
% NewMeshData = FSTBX_NewMeshData(FlatPatch, 0.5);

if (FSTBX_Patch.IsFullSurface)
    error('FSTBX_Patch can not be a full surface');
end

if (FSTBX_Patch.IsFlat == false)
    error('Input patch should be a flat 2d patch')
end

% Determine the range for grid and construct the grid
Max = ceil(max(max(FSTBX_Patch.Vertices)));
Min = floor(min(min(FSTBX_Patch.Vertices)));
```

```
[FSTBX_MeshData.X,FSTBX_MeshData.Y] = meshgrid(Min:StepSize:Max,
                                               Min:StepSize:Max);
FSTBX_MeshData.Values = zeros(size(FSTBX_MeshData.X));
FSTBX_MeshData.ValueIsSetMask = true(size(FSTBX_MeshData.Values));
FSTBX_MeshData.Range = [Min Max];
FSTBX_MeshData.StepSize = StepSize;
```

B.8 FSTBX_Smooth_Value

```
function FSTBX_Value_Out = FSTBX_Smooth_Value(FSTBX_Value_In,
                                              FSTBX_Surface,
                                              nIterations)
%
% Usage: FSTBX_Value_Out = FSTBX_Smooth_Value(FSTBX_Value_In,
%                                              FSTBX_Surface,
%                                              nIterations)
% Input:
%   FSTBX_Value_In      Values to smooth.
%   FSTBX_Surface Surface or patch that the values smoothed on.
%   nIterations         Number of smoothing iterations.
%
% Output:
%   FSTBX_Value_Out Smoothed values.
%
% This function smooths the vertex values on a surface. The value of
% a vertex and values of the vetices that share the same face are averaged
% and assigned to the center vertex. This is repeated nIterations times.
% Before smoothing the vertex neighbour list of the surface should be set
% by using FSTBX_Calculate_Neighbours.
%
```

```
% Example:
% V_Smoothed = FSTBX_Smooth_Value(V_ecc, FullSurface, 10);

if (isfield(FSTBX_Surface,'NeighbourList') == false)
    error('First calculate the neighbours using
           FSTBX_Calculate_Neighbours');
end

% Determine the vertex values if the surface is patch.
if (FSTBX_Surface.IsFullSurface == false)
    Values = FSTBX_Value_In.Values(FSTBX_Surface.VertexIdx(:));
else
    Values = FSTBX_Value_In.Values;
end

% Smooth the values. This operation is quite slow.
for count = 1:nIterations
    for n=1:FSTBX_Surface.NumberOfVertices
        sValues(n,1) = mean(Values(FSTBX_Surface.NeighbourList{n}));
    end
    Values = sValues;
end

FSTBX_Value_Out = FSTBX_Value_In;

if (FSTBX_Surface.IsFullSurface == false)
    FSTBX_Value_Out.Values(FSTBX_Surface.VertexIdx(:)) = Values(:);
else
    FSTBX_Value_Out.Values = Values;
end
```

B.9 FSTBX_Calculate_Neighbours

```
function FSTBX_SurfaceOut = FSTBX_Calculate_Neighbours(FSTBX_SurfaceIn)
%
% Usage: FSTBX_SurfaceOut = FSTBX_Calculate_Neighbours(FSTBX_SurfaceIn)
%
% Input:
%   FSTBX_SurfaceIn     Input surface or patch.
%
% Output:
%   FSTBX_SurfaceOut Output surface with .NeighbourList added.
%
% Adds .NeighbourList field to the input surface which has the vertex
% neighbourhood information.
%
% Example:
% Surf = FSTBX_Smooth_Value(Surf);

if (isfield(FSTBX_SurfaceIn,'NeighbourList') == false)
    TempFaces = FSTBX_SurfaceIn.Faces;
    NB = cell(FSTBX_SurfaceIn.NumberOfVertices,1);
    NumberOfResiduals = 1;
    while (NumberOfResiduals > 0)
    [Mask,ind] = ismember(1:FSTBX_SurfaceIn.NumberOfVertices,TempFaces);
    [i,j] = ind2sub(size(FSTBX_SurfaceIn.Faces),ind);
    for n=1:FSTBX_SurfaceIn.NumberOfVertices
        if (i(n)~=0)
            NB{n} = cat(2,NB{n}, FSTBX_SurfaceIn.Faces(i(n),1:3));
            TempFaces(i(n),j(n)) = 0;
        end
    end
    NumberOfResiduals = sum(sum(TempFaces ~= 0));
```

```
    end
    for n=1:FSTBX_SurfaceIn.NumberOfVertices
            NB{n} = unique(NB{n});
    end
    FSTBX_SurfaceIn.NeighbourList = NB;
end
FSTBX_SurfaceOut = FSTBX_SurfaceIn;
```

B.10 Low Level Functions

```
function [vertex_coords, faces] = read_surf(fname)
%
% Usage: [vertex_coords, faces] = read_surf(fname)
%
% Reads the vertex coordinates and face lists from a surface file. Vertex
% indices are unique ID's for surfaces, patches and w files given in RAS
% coordinates. Face list gives the vertice indices that define a face.
%
% Supported Ext: .graymid .inflated .orig .white .smoothwm .pial
%
% Triangle File Format :
% FileTypeId        int3
% Creator           null(0 or 0x0A) term string
% Time              null(0 or 0x0A) term string
% VertexCount       int4
% FaceCount         int4
% {
%       Right       float4
%       Anterior    float4
%       Superior    float4
% } #VertexCount#
```

```
% {
%       Vertex1     int4
%       Vertex2     int4
%       Vertex3     int4
% }
%
%
% NOTE: Faces are in C convention (starting with 0).
% So add 1 to use with matlab.
%       Faces = Faces + 1;
%       Reading the faces from a quad file can take a very long
%       time due to the goofy format that they are stored in.

% Definitions for surface file format
TRIANGLE_FILE_MAGIC_NUMBER =  16777214;
QUAD_FILE_MAGIC_NUMBER =  16777215;

% open it as a big-endian file
fid = fopen(fname, 'rb', 'b');
if (fid < 0)
 str = sprintf('Could not open surface file %s.', fname);
 error(str);
end

% Read the format of the surface (Triangles or Quadrangles)
magic = fread3(fid) ;

% Read the quadrangle format surface
if (magic == QUAD_FILE_MAGIC_NUMBER)
 vnum = fread3(fid);
 fnum = fread3(fid);
 vertex_coords = fread(fid, vnum*3, 'int16') ./ 100;
```

```
 if (nargout > 1)
   for i=1:fnum
   for n=1:4
faces(i,n) = fread3(fid);
 end
 end
     end
% Read the triangle format surface
elseif (magic == TRIANGLE_FILE_MAGIC_NUMBER)
     % Read creator string
     fgets(fid);
     % Read time string
 fgets(fid);
 % Read the number of vertices
     vnum = fread(fid, 1, 'int32');
     % Read the number of faces
 fnum = fread(fid, 1, 'int32');
 % Read vertices
     vertex_coords = fread(fid, vnum*3, 'float32');
 % Read faces
     faces = fread(fid, fnum*3, 'int32');
 faces = reshape(faces, 3, fnum)';
end

vertex_coords = reshape(vertex_coords, 3, vnum)';

% Close the file
fclose(fid);

function [Vertex_Number, Vertex_Coords] = read_patchfile(fname)
%
% Usage : [Vertex_Number, Vertex_Coords] = read_patchfile(fname)
```

```
%
% Patches are subsections of complete surfaces
% Retain the same vertex numbering with the surface, so that a patch's
% vertices can be cross referenced to the complete set.
%
% Supported Ext: .patch.flat .patch.3d
%
% Patch File Format :
% PointCount        int4
% {
%       VtxIdx      int4
%       X           int2
%       Y           int2
%       Z           int2
% } #PointCount#
%
%
% NOTE: A negative Vertex_Number indicates vertex is not on the border
%       The Vertex_Number is already in Matlab convention (starting with 1)

% Open file as big endian binary
fid = fopen(fname, 'rb', 'b') ;

if (fid < 0)
 str = sprintf('Could not open patch file %s.', fname) ;
 error(str) ;
end

% Read the Point Count
nVertices = fread(fid, 1, 'int32') ;

% Read the corresponding vertex indices
```

```
Vertex_Number = fread(fid,nVertices,'int32',6);

% Return to the begining of the file
frewind(fid);

% Skip 8 bytes ; one for PointCount and one to skip first vertex index
fread(fid, 2, 'int32');

% Read the patch coordinates
Vertex_Coords = fread(fid, 3*nVertices, '3*int16', 4);
Vertex_Coords = reshape(Vertex_Coords, 3, nVertices)' ;
Vertex_Coords = Vertex_Coords ./ 100;

% Close the file
fclose(fid);

function curv = read_curv(fname)
%
% Usage: curv = read_curv(fname)
%
% Reads a binary curvature file into a vector. Same length as vertex list
% of surface. Each value is the value of corresponding vertex. This is a
% general format to give additional values per vertices.
%
% Supported Ext: .sulc .curv .thickness .defect_status .defect_labels
%
% New Version File Format:
% FileTypeId            int3
% VertexCount           int4
% FaceCount             int4
% ValsPerVertex         int4
% {
```

```
%       VertexValue     float4
% } #VertexCount#
%
% Old Version File Format:
% VertexCount           int3
% FaceCount             int3
% {
%       VertexValue     int2
% } #VertexCount#

% There are two versions new and old
NEW_VERSION_MAGIC_NUMBER = 16777215;

% open it as a big-endian file
fid = fopen(fname, 'rb', 'b') ;
if (fid < 0)
 str = sprintf('could not open curvature file %s.', fname) ;
 error(str) ;
end

% First 3 byte is type for new version and #Vertices for old version
vnum = fread3(fid) ;

if (vnum == NEW_VERSION_MAGIC_NUMBER)
vnum = fread(fid, 1, 'int32');
fnum = fread(fid, 1, 'int32'); % Not useful
vals_per_vertex = fread(fid, 1, 'int32'); % Currently only 1 is supported
    curv = fread(fid, vnum, 'float');
    fclose(fid) ;
else
    fnum = fread3(fid);
    curv = fread(fid, vnum, 'int16') ./ 100;
```

```
    fclose(fid);
end

function [Value,Vertex] = read_wfile(fname)
%
% Usage : [Value,Vertex] = read_wfile(fname)
%
% A W file provides values to associate with vertices.
% Retain the same vertex numbering with the surface
%
% Supported Ext: .w
%
% W File Format :
% Latency           int2
% ValueCount        int3
% {
%       VtxNum      int3
%       Value       float4
% } #ValueCount#
%
%
% NOTE: Possibly providing values for only a subset of a surface's vertices
%       Possibly not in sequential order

% Open file as big-endian binary
fid = fopen(fname, 'rb', 'b') ;
if (fid < 0)
  str = sprintf('Could not open w file %s.', fname) ;
  error(str) ;
end

% Skip first 2 bytes
```

```
fread(fid, 1, 'int16') ;

% Read the number of vertex values
vnum = fread3(fid) ;

% Read 3byte integer that corresponds to the vertex number
tVertex = fread(fid, 3*vnum, '3*uchar', 4);
tVertex = reshape(tVertex, 3, vnum)' ;
Vertex(:,1) = bitshift(tVertex(:,1),16) + bitshift(tVertex(:,2),8)
      + tVertex(:,3);

% Return to the begining of the file
frewind(fid);

% Skip 8 bytes - 2 for Latency, 3 for vnum, 3 for first vertex number
fread(fid,8,'uchar');

% Read vertex values
Value = fread(fid, vnum, 'float', 3);

% Close the file
fclose(fid) ;

function fs = read_fs(fname)
%
% Usage: fs = read_fs(fname)
%
% Reads the Fieldsign and Weight of field sign files.
%
% Supported Ext: .fs .fm
%
% FS File Format
```

```
% VertexNumber          int32
% {
%       Value           float
% }

% open it as a big-endian file
fid = fopen(fname, 'rb', 'b') ;
if (fid < 0)
 str = sprintf('Could not open fieldsign file %s.', fname) ;
 error(str) ;
end

vnum = fread(fid, 1, 'int32') ;
fs = fread(fid, vnum, 'float');
fclose(fid);

function [retval] = fd3(fid)
% usage: [retval] = fd3(fid)
% read a 3 byte integer out of a file
b1 = fread(fid, 1, 'uchar') ;
b2 = fread(fid, 1, 'uchar') ;
b3 = fread(fid, 1, 'uchar') ;
retval = bitshift(b1, 16) + bitshift(b2,8) + b3 ;
```

APPENDIX C. MATLAB CODE FOR FREQUENCY ANALYSIS

C.1 fourier_map

```
function fourier_map( AFNIInput )
% -----------
% fourier_map
% -----------
%
% Usage : fourier_map( AFNIInput )
%

% Load the AFNI brik. BrikLoad function is in AFNI Matlab library
% that can be downloaded from the web page of AFNI/SUMA
[err, Image, Info, ErrMessage] = BrikLoad (AFNIInput);

% While calculate_fourier_map needs slice idx, loop over slices.
Nz = size(Image,3);
for i=1:Nz
    % In our case DFFT index of the stimulus frequency is 17.
    [FMap, Bold] = calculate_fourier_map(Image, 17, i);

    % Create a slice scan order table. We use equally spaced
    % interleaved increasing order.
    SliceOrder = [1:2:30 2:2:30]';

    % Correct phase delay caused by interleaved slice ordered ge-epi.
    FMap = correct_phase_delay(FMap, SliceOrder, 2, 32, 1, i);
```

```
    % Seperate the real and imaginary parts of the complex values.
    Real = real(FMap);
    Imaginary = imag(FMap);

    % Write Coherence values to bfloat files
    SaveFileName = sprintf('%s_bold_%03d', AFNIInput, i-1);
%convert to zero based c indexing
    save_bfloat(Bold', SaveFileName, 'b');

    % Write real values to bfloat files
    SaveFileName = sprintf('%s_real_%03d', AFNIInput, i-1);
%convert to zero based c indexing
    save_bfloat(Real', SaveFileName, 'b');

    % Write the imaginary values to bfloat files
    SaveFileName = sprintf('%s_imag_%03d', AFNIInput, i-1);
    save_bfloat(Imaginary', SaveFileName, 'b');
end
```

C.2 calculate_fourier_map

```
function [FourierMap,Bold] = calculate_fourier_map( Image4D, StimulusIdx,
    SliceIdx )
% ---------------------
% calculate_fourier_map
% ---------------------
%
% Usage : [FourierMap,Bold] = calculate_fourier_map( Image4D, StimulusIdx,
%      SliceIdx )
%
```

```
% Get fourier transform along the time axis
ComplexSignal = fft(Image4D(:,:,SliceIdx,:), [], 4);

% Get the amplitudes for each frequency
FreqAmp = abs(ComplexSignal);

% We quantified the BOLD change as the ratio of the amplitude of the
% stimulus sinusoidal to the base bold amplitude
Bold = 100 * (FreqAmp(:,:,:,StimulusIdx) ./ FreqAmp(:,:,:,1));

% Zero out the amplitudes that is not wanted to contribute in the
% calculation of the signal variance
FreqAmp(:,:,:,1) = 0;
FreqAmp(:,:,:,129:256) = 0;

% Get the power (sum square of amplitudes) for selected frequencies of each
% voxel
SumFreqAmpSqr = sum(FreqAmp .* FreqAmp,4);

% Calculate the complex values for each voxel so that the amplitude is the
% coherence of the signal for that voxel.
FourierMap = ComplexSignal(:,:,:,StimulusIdx) ./ sqrt(SumFreqAmpSqr);
```

C.3 correct_phase_delay

```
function FMap_Out = correct_phase_delay(FMap_In, SliceOrder, TR, StimPeriod,
ReferenceSlice, SliceNo)
% -------------------
% correct_phase_delay
% -------------------
%
```

```
% Usage : FMap_Out = correct_phase_delay(FMap_In, SliceOrder, TR,
%                                        StimPeriod, ReferenceSlice, SliceNo)
%

% Get the number of slices from SliceOrder array.
NumSlices = size(SliceOrder,1);

% Calculate phase shift to scan a single slice.
ShiftPerSlice = (2 * pi) * (TR/StimPeriod) * (1/NumSlices);

% Determine the phase shift for SliceNo.
PhaseShift = (find(SliceNo == SliceOrder) - find(ReferenceSlice == SliceOrder))
     * ShiftPerSlice;

% Convert complex value to polar form.
R = abs(FMap_In);
Theta = angle(FMap_In);

% Shift the phases.
Theta = Theta - PhaseShift;

% Wrap values outside the -pi:pi range.
Mask = Theta > pi;
Theta = Theta - (2 * pi * double(Mask));
Mask = Theta < pi;
Theta = Theta + (2 * pi * double(Mask));

% Finally convert back the polar values to cartesian.
FMap_Out = R .* exp(i * Theta);
```

C.4 save_bfloat

```
function save_bfloat(Image, FileNameWithoutExtention, endian)
% -----------
% save_bfloat
% -----------
%
% Usage : save_bfloat(Image, FileNameWithoutExtention, endian)
%
% Stores a 3D or 2D matrix in bfloat format
%
% b: big endian (0 in hdr of bfloat)
% l: little endian (1 in hdr of bfloat)

DataFileName = strcat(FileNameWithoutExtention, '.bfloat');
HeaderFileName = strcat(FileNameWithoutExtention, '.hdr');

% Get the sizes of Image
Ni = size(Image,1);
Nj = size(Image,2);
Nz = size(Image,3);

% Open file in write mode with given endian
fid = fopen(DataFileName,'w',endian);

% Write data to .bfloat file
% To do: Check for multislice data
for (z=1:Nz)
    Image(:,:,z) = Image(:,:,z)';
    fwrite(fid, Image(:,:,z), 'single');
end
```

```
% Close the .bfloat file
fclose(fid);

% Convert the endian input to the .hdr format
if (endian == 'b')
    hdr4 = 0;
elseif (endian == 'l')
    hdr4 = 1;
else
    prinf('Unknown endian !');
    return;
end

% Write size and endian to the .hdr file
hdrfile = fopen(HeaderFileName,'w');
hdr = fprintf(hdrfile, '%d %d %d %d', Nj, Ni, Nz, hdr4);
fclose(hdrfile);
```

APPENDIX D. MATLAB CODE FOR RETINOTOPY ANALYSIS

D.1 analiz

```
function analiz(Subject,Week)

FolderName = sprintf('/home/andac/data/%s/%d', Subject, Week)
cd(FolderName);

sh_command = sprintf('3dvolreg -prefix expanding-vreg -1Dfile
             expanding-vreg-params.txt -base 3 expanding+orig');
system(sh_command);

fourier_map('expanding-vreg+orig');

system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
       -regdat register.dat -nslices 30 expanding-vreg+orig_real_%03d.bfloat
       rh graymid ./expanding_real_rh.w');
system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
       -regdat register.dat -nslices 30 expanding-vreg+orig_imag_%03d.bfloat
       rh graymid ./expanding_imag_rh.w');

system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
       -regdat register.dat -nslices 30 expanding-vreg+orig_real_%03d.bfloat
       lh graymid ./expanding_real_lh.w');
system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
       -regdat register.dat -nslices 30 expanding-vreg+orig_imag_%03d.bfloat
       lh graymid ./expanding_imag_lh.w');
```

```
system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
       -regdat register.dat -nslices 30 expanding-vreg+orig_bold_%03d.bfloat
       rh graymid ./expanding_bold_rh.w');
system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
       -regdat register.dat -nslices 30 expanding-vreg+orig_bold_%03d.bfloat
       lh graymid ./expanding_bold_lh.w');

delete *.bfloat;
delete *.hdr;
delete expanding-vreg+orig.*;

sh_command = sprintf('3dvolreg -prefix rotating-vreg -1Dfile
      rotating-vreg-params.txt -base ''expanding+orig[3]'' rotating+orig');
system(sh_command);

fourier_map('rotating-vreg+orig');

system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
       -regdat register.dat -nslices 30 rotating-vreg+orig_real_%03d.bfloat
       rh graymid ./rotating_real_rh.w');
system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
       -regdat register.dat -nslices 30 rotating-vreg+orig_imag_%03d.bfloat
       rh graymid ./rotating_imag_rh.w');

system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
       -regdat register.dat -nslices 30 rotating-vreg+orig_real_%03d.bfloat
       lh graymid ./rotating_real_lh.w');
system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
       -regdat register.dat -nslices 30 rotating-vreg+orig_imag_%03d.bfloat
       lh graymid ./rotating_imag_lh.w');

system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
```

```
        -regdat register.dat -nslices 30 rotating-vreg+orig_bold_%03d.bfloat
        rh graymid ./rotating_bold_rh.w');
system('/home/andac/freesurfer/FreeSurfer0.8/bin/Linux/paint
        -regdat register.dat -nslices 30 rotating-vreg+orig_bold_%03d.bfloat
        lh graymid ./rotating_bold_lh.w');

delete *.bfloat;
delete *.hdr;
delete rotating-vreg+orig.*;

for i=1:2
    if (i==1)
        Hemisphere = 'rh'
    else
        Hemisphere = 'lh'
    end

    SurfaceFileName = sprintf('/home/andac/freesurfer/subjects/%s/surf/
      %s.inflated',Subject,Hemisphere);
    PatchFileName = sprintf('/home/andac/freesurfer/subjects/%s/surf/
    %s.occip.patch.flat',Subject,Hemisphere);

    RotatingRealFileName = sprintf('rotating_real_%s.w',Hemisphere);
    RotatingImagFileName = sprintf('rotating_imag_%s.w',Hemisphere);

    ExpandingRealFileName = sprintf('expanding_real_%s.w',Hemisphere);
    ExpandingImagFileName = sprintf('expanding_imag_%s.w',Hemisphere);

    RotatingBoldFileName = sprintf('rotating_bold_%s.w',Hemisphere);
    ExpandingBoldFileName = sprintf('expanding_bold_%s.w',Hemisphere);

    SaveFileName = sprintf('%s.mat',Hemisphere);
```

```
sprintf('Reading Surfaces')
S = FSTBX_Read_Surface(SurfaceFileName);
P = FSTBX_Read_Patch(PatchFileName, S);

% Rotating
sprintf('Reading Polar Values')
Vr_r = FSTBX_Read_Value(RotatingRealFileName,S,'W');
Vr_i = FSTBX_Read_Value(RotatingImagFileName,S,'W');

sprintf('Calculating Neighbours for Patch')
P = FSTBX_Calculate_Neighbours(P);
sprintf('Smoothing Real')
Vr_r = FSTBX_Smooth_Value(Vr_r, P, 50);
sprintf('Smoothing Imaginary')
Vr_i = FSTBX_Smooth_Value(Vr_i, P, 50);

Vr_p = Vr_i;
Vr_p.Values = angle(Vr_r.Values + sqrt(-1) * Vr_i.Values);

Vr_a = Vr_i;
Vr_a.Values = abs(Vr_r.Values + sqrt(-1) * Vr_i.Values);

% Expanding
sprintf('Reading Eccentricity Values')
Ve_r = FSTBX_Read_Value(ExpandingRealFileName,S,'W');
Ve_i = FSTBX_Read_Value(ExpandingImagFileName,S,'W');

sprintf('Calculating Neighbours for Patch')
P = FSTBX_Calculate_Neighbours(P);
sprintf('Smoothing Real')
Ve_r = FSTBX_Smooth_Value(Ve_r, P, 50);
```

```
    sprintf('Smoothing Imaginary')
    Ve_i = FSTBX_Smooth_Value(Ve_i, P, 50);

    Ve_p = Ve_i;
    Ve_p.Values = angle(Ve_r.Values + sqrt(-1) * Ve_i.Values);

    Ve_a = Ve_i;
    Ve_a.Values = abs(Ve_r.Values + sqrt(-1) * Ve_i.Values);

    % Bold
    sprintf('Reading Bold Changes')
    Ve_bold = FSTBX_Read_Value(ExpandingBoldFileName,S,'W');
    Vr_bold = FSTBX_Read_Value(RotatingBoldFileName,S,'W');

    sprintf('Calculating Neighbours for Patch')
    P = FSTBX_Calculate_Neighbours(P);
    sprintf('Smoothing Expanding Coherence')
    Ve_bold = FSTBX_Smooth_Value(Ve_bold, P, 50);
    sprintf('Smoothing Rotating Coherence')
    Vr_bold = FSTBX_Smooth_Value(Vr_bold, P, 50);

    save(SaveFileName, 'S', 'P', 'Vr_p', 'Vr_a', 'Ve_p',
 'Ve_a', 'Ve_bold', 'Vr_bold');

end

delete *.w;
```

D.2 vfs

```
function vfs(Subject, Week, Hemisphere)
```

```
% function vfs(Subject, Week, Hemisphere)
%
% Select a region at the eccentricity center for phase correction.
% Calculate the VFS.
% Save some figures.
% Save the matlab variables to a mat file.

% Load the surfaces and smoothed phases saved by analiz.m function
DataFile = sprintf('/home/andac/data/%s/%d/%s.mat',
   Subject,Week,Hemisphere);
load(DataFile);

% ---------------------------------------------
%        PHASE OFFSET CORRECTION
% ---------------------------------------------
% The eccentricity phase offset should be corrected to calculate the
% magnification factor properly. Since the VFS depends on the
% change of the phases, this has no effect on VFS calculation.

% The sign of the phase depends on the direction of the stimulus
% and the sign of the exponent of the discrete fourier transform.
% Also add a pi to shift the range to 0-2pi
Ve_p.Values = (-Ve_p.Values) + pi;
Vr_p.Values = (-Vr_p.Values) + pi;

% Interpolate the eccentricity phase values of vertices to a
% regular grid
Me_p = FSTBX_Value2MeshData(Ve_p, P, 0.5, 'cubic');

% While we have data on a regular grid we can show it as a 2D image
figure,imshow(Me_p.Values,[]);
colormap Hot;
```

```
% Create a selection mask of the user selected region. This is to
% calculate the overall phase offset.
M_Mask = Me_p;
M_Mask.Values = double(roipoly);

% Resample the gridded data at the coordinates of the vertices
V_Mask = FSTBX_MeshData2Value(M_Mask,P,'cubic');

% While we use interpolation to find the values at vertices,
% again set a treshold to find the mask on vertices
V_Mask.Values = double(V_Mask.Values > 0.5);

% shift the phase values outside the selected region by 2pi
% so that it is guaranteed not to contribute to the calculation
% of the min phase.
Ve_p_temp = Ve_p;
Ve_p_temp.Values = Ve_p_temp.Values +
   2 * pi * double(V_Mask.Values == 0);

% The phase offset is the minimum of the phases in the region
phase_offset = min(Ve_p_temp.Values);

% Clear the temporary variables
clear M_Mask; clear Me_p; clear Ve_p_temp; clear V_Mask;

% Shift all the phases -phase_offset
Ve_p = addphase(Ve_p,-phase_offset);

% Shift the same amount also for the polar phases. This is
% just for visualisation purposes so I just apply the same
% offset to polar angles. If it is needed to have correct
% phases for polar angle, this should be revised.
```

```
Vr_p = addphase(Vr_p, -phase_offset);

% ------------------------------------------
%        VISUAL FIELD SIGN CALCULATION
% ------------------------------------------

% Calculate phase gradients on flattened surface
[eGx, eGy] = calc_grad(Ve_p,P);
[rGx, rGy] = calc_grad(Vr_p,P);

% Calculate the cross product of ecc and polar phase
% gradients on a regular grid.
M_fs = FSTBX_NewMeshData(P,0.5);
M_fs.Values = (eGx.Values.*rGy.Values) -
      (eGy.Values.*rGx.Values);

% Resample the cross products at vertex coordinates
V_fs = Ve_p;
V_fs = FSTBX_MeshData2Value(M_fs,P,'cubic');

% The field sign is the sign of the cross product.
V_fs.Values = sign(V_fs.Values);

% Save the calculated variables
SaveFileName = sprintf('/home/andac/data/%s/%d/%s_vfs.mat',
       Subject,Week,Hemisphere);
save(SaveFileName, 'Ve_p','Vr_p', 'V_fs');
```

D.3 calc_grad

```
function [Gx, Gy] = calc_grad(FSTBX_Value, FSTBX_Patch)
```

```
V_0 = addphase(FSTBX_Value, 0);
M_0 = FSTBX_Value2MeshData(V_0, FSTBX_Patch, 0.5, 'cubic');
[Gx_0, Gy_0] = gradient(M_0.Values, M_0.StepSize);
G0_Amp = abs(Gx_0 + sqrt(-1) * Gy_0);

V_90 = addphase(FSTBX_Value, pi);
M_90 = FSTBX_Value2MeshData(V_90, FSTBX_Patch, 0.5, 'cubic');
[Gx_90, Gy_90] = gradient(M_90.Values, M_90.StepSize);
G90_Amp = abs(Gx_90 + sqrt(-1) * Gy_90);

G0_mask = (G0_Amp < G90_Amp);
G90_mask = ~G0_mask;

Gx = M_0; Gy=M_0;

Gx.Values = (Gx_0 .* double(G0_mask)) + (Gx_90 .* double(G90_mask));
Gy.Values = (Gy_0 .* double(G0_mask)) + (Gy_90 .* double(G90_mask));
```

D.4 addphase

```
function FSTBX_ValueOut = addphase(FSTBX_ValueIn, ShiftInRadyan)

% inp and outp may be Value or mesh

FSTBX_ValueOut = FSTBX_ValueIn;

FSTBX_ValueOut.Values = FSTBX_ValueOut.Values + ShiftInRadyan;

Mask = FSTBX_ValueOut.Values > 2 * pi;
FSTBX_ValueOut.Values = FSTBX_ValueOut.Values - (double(Mask) * (2*pi));
```

```
Mask = FSTBX_ValueOut.Values < 0;
FSTBX_ValueOut.Values = FSTBX_ValueOut.Values + (double(Mask) * (2*pi));
```

REFERENCES

1. Sereno, M. I., C. T. McDonald, and J. M. Allman, "Analysis of retinotopic maps in extrastriate cortex," *Cerebral Cortex*, Vol. 6, pp. 601–620, 1994.

2. Sereno, M. I., A. M. Dale, J. B. Reppas, K. K. Kwong, J. W. Belliveau, T. J. Brady, B. R. Rosen, and R. B. H. Tootell, "Borders of multiple visual areas in humans revealed by functional mri," *Science*, Vol. 268, pp. 889–893, 1995.

3. Wandell, B. A., "Computational neuroimaging of human visual cortex," *Annu. Rev. Neurosci.*, Vol. 22, pp. 145–173, 1999.

4. Tootell, R. B. H., A. M. Dale, M. I. Sereno, and R. Malach, "New images from human visual cortex," *Trends Neurosci.*, Vol. 19, pp. 481–489, 1996.

5. Tootell, R. B. H., J. D. Mendola, N. K. Hadjikhani, P. J. Ledden, A. K. Liu, J. B. Reppas, M. I. Sereno, and A. M. Dale, "Functional analysis of v3a and related areas in human visual cortex," *The Journal of Neuroscience*, Vol. 17, pp. 7060–7078, Sept 1997.

6. Sereno, M. I., "Plasticity and its limits," *Nature*, Vol. 435, pp. 288–289, 2005.

7. Gilbert, C. D., "Adult cortical dynamics," *Physiological Reviews*, Vol. 78, no. 2, pp. 467–485, 1998.

8. Buonomano, D. V., and M. M. Merzenich, "Cortical plasticity: From synapses to maps," *Annu.Rev.Neurosci.*, Vol. 21, pp. 149–186, 1998.

9. Trachtenberg, J. T., B. E. Chen, G. W. Knott, G. Feng, J. R. Sanes, E. Welker, and K. Svoboda, "Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex," *Nature*, Vol. 420, pp. 788–794, 2002.

10. Kaas, J. H., L. A. Krubitzer, Y. M. Chino, A. L. Langston, E. H. Polley, and N. Blair, "Reorganization of retinotopic cortical maps in adult mammals after lesions of the retina," *Science*, Vol. 248, pp. 229–231, 1990.

11. Chino, Y. M., J. H. Kaas, E. L. Smith, A. L. Langston, and H. Cheng, "Rapid reorganization of cortical maps in adult cats following restricted deafferentation in retina," *Vision Res.*, Vol. 32, pp. 789–796, 1992.

12. Schmid, L. M., M. G. P. Rosa, M. B. Calford, and J. S. Ambler, "Visuotopic reorganization in the primary cortex of adult cats following monocular and binocular retinal lesions," *Cereb. Cortex*, Vol. 6, pp. 388–405, 1996.

13. Smirnakis, S. M., A. A. Brewer, M. C. Schmid, A. S. Tolias, A. Schuz, M. Augath, W. Inhoffen, B. A. Wandell, and N. K. Logothetis, "Lack of long-term cortical reorganization after macaque retinal lesions," *Nature*, Vol. 435, pp. 300–307, May 2005.

14. Crist, R. E., W. Li, and C. D. Gilbert, "Learning to see: experience and attention in primary visual cortex," *Nature Neuroscience*, Vol. 4, no. 5, pp. 519–525, 2001.

15. Sugita, Y., "Global plasticity in adult visual cortex following reversal of visual input," *Nature*, Vol. 380, pp. 523–526, 1996.

16. Xerri, C., J. M. Stern, and M. M. Merzenich, "Alterations of the cortical representationof the rat ventrum induced by nursing behavior," *J.Neurosci.*, Vol. 14, pp. 1710–1721, 1994.

17. Recanzone, G. H., C. E. Schreiner, and M. M. Merzenich, "Plasticity in the frequency representation of primary auditory cortex following following discrimination training in adult owl monkeys," *Neurosci.*, Vol. 13, pp. 87–104, 1993.

18. Grill-Spector, K., R. Henson, and A. Martin, "Repetition and the brain: neural models of stimulus-specific effects," *Trends in Cognitive Sciences*, Vol. 10, pp. 14–23, Jan 2006.

19. Miller, E. K., and R. Desimone, "Parallel neuronal mechanisms for short-term memory," *Science*, Vol. 263, pp. 520–522, 1994.

20. Grill-Spector, K., and R. Malach, "fmr-adaptation: a tool for studying the functional properties of human cortical neurons," *Acta Psychol.(Amst.)*, Vol. 107, pp. 293–321, 2001.

21. Li, L., and et al., "The representation of stimulus familiarity in anterior inferior temporal cortex," *J. Neurophysiol.*, Vol. 69, pp. 1918–1929, 1993.

22. Desimone, R., "Neural mechanisms for visual memory and their role in attention," *Proc.Natl.Acad.Sci.U.S.A.*, Vol. 93, pp. 13494–13499, 1996.

23. Wiggs, C. L., and A. Martin, "Properties and mechanisms of perceptual priming," *Curr.Opin.Neurobiol.*, Vol. 8, pp. 227–233, 1998.

24. Sobotka, S., and J. L. Ringo, "Mnemonic responses of single units recorded from monkey inferotemporal cortex, accessed via transcommissural versus direct pathways: a dissociation between unit activity and behavior," *J. Neuroscience*, Vol. 16, pp. 4222–4230, 1996.

25. Henson, R. N., and M. D. Rugg, "Neural response suppression, haemodynamic repetition effets, and behavioural priming," *Neuropsychologia*, Vol. 41, pp. 263–270, 2003.

26. Vuilleumier, P., and et al., "Multiple levels of visual object constancy revealed by event-related fmri of repetition priming," *Nat.Neurosci.*, Vol. 5, pp. 491–499, 2002.

27. James, T. W., and et al., "Differential effects of viewpoint on object-driven activation in dorsal and ventral streams," *Neuron*, Vol. 35, pp. 793–801, 2002.

28. van Turennout, M., and et al., "Long-lasting cortical plasticity in the object naming system," *Nat.Neurosci.*, Vol. 3, pp. 1329–1334, 2000.

29. Warnking, J., M. Dojat, A. Guerin-Dugue, C. Delon-Martin, S. Olympieff, N. Richard, A. Chehikian, and C. Segebarth, "fmri retinotopic mapping-step by step," *NeuroImage*, Vol. 17, pp. 1665–1683, 2002.

30. Engel, S. A., G. H. Glover, and B. A. Wandell, "Retinotopic organization in human visual cortex and the spatial precision of functional mri," *Cerebral Cortex*, Vol. 7, pp. 181–192, 1997.

31. Dumoulin, S. O., R. D. Hoge, J. Curtis L Baker, R. F. Hess, R. L. Achtman, and A. C. Evans, "Automatic volumetric segmentation of human visual retinotopic cortex," *NeuroImage*, Vol. 18, pp. 576–587, 2003.

32. Kalsin, B. S., "Design and implementation of synchronized visual stimulation system for functional magnetic resonance imaging (fmri) scanners," Master's thesis, Bogazici University, Istanbul, Turkey, 2005.

33. Slotnick, S. D., and S. Yantis, "Efficient acquisition of human retinotopic maps," *Human Brain Mapping*, Vol. 18, pp. 22–29, 2003.

34. Dale, A. M., B. Fischl, and M. I. Sereno, "Cortical surface-based analysis, i.segmentation and surface reconstruction," *NeuroImage*, Vol. 9, pp. 179–194, 1999.

35. Fischl, B., M. I. Sereno, R. B. H. Tootell, and A. M. Dale, "High-resolution intersubject averaging and a coordinate system for the cortical surface," *Human Brain Mapping*, Vol. 8, pp. 272–284, 1999.

36. Fischl, B., M. I. Sereno, and A. M. Dale, "Cortical surface-based analysis, ii.inflation, flattening and a surface-based coordinate system," *NeuroImage*, Vol. 9, pp. 195–207, 1999.

37. Brewer, A. A., W. A. Press, N. K. Logothetis, and B. A. Wandell, "Visual areas in macaque cortex measured using functional magnetic resonance imaging," *The Journal of Neuroscience*, Vol. 22, pp. 10416–10426, Dec 2002.

38. Slotnick, S. D., S. A. Klein, T. Carney, and E. E. Sutter, "Electrophysiological estimate of human cortical magnification," *Clinical Neurophysiology*, Vol. 112, pp. 1349–1356, 2001.