3D Crustal Temperature Modeling over Japan by Combine Thermal Remote Sensing and Well-logging Data for Geothermal Resource Assessment

Bingwei Tian

Graduate School of Engineering

Kyoto University

This dissertation is submitted for the degree of

Doctor of Philosophy

October 2014

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains fewer than 100,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Bingwei Tian
August 2014

Acknowledgements

I would like to first thank my supervisor, Prof. Dr. Katsuaki Koike, for all of his support, encouragement, and advice over the years. Koike-sensei's precious and steady support allowed me to grow into a contributing member of the scientific community. He gave me substantial liberties, but was always been available to aid in my understanding of problems and to give ideas when I was unsure of what to do going forward. I could not have completed this work without him. I also would like thank my vice supervisors, Asso. Prof. Dr. Yoshitada Mito and Asso. Prof. Dr. Yosuke Higo. Especially I am grateful to Mito-sensei as my asvisor of "The Education Program for Global Leaders in Advanced Engineering and Pharmaceutical Sciences (GL), Kyoto University". My grateful thanks also extend to my committee members as well. Prof. Dr. Tamura and Prof. Dr. Mito have reviewed my dissertation and offered valuable insight. My work has been better for their input. I would like to thank all staffs and members in the "Laboratory of Geosphere Environment, Kyoto Univerity" and "Laboratory of Applied Geosciences and Technology, Kumamoto University". GL member, GELK members, especiaaly KASHIWAYA, Kouki, and Asaue for their many valuable discussions

I need to thank several people without whom this research would not have been completed. Steve Saleeby, thank you for the budgeting code, and for countless modeling questions answered. Tristan L'Ecuyer, thank you for your help and insight into observational data, and for your helpful suggestions as a co-author. Matt Lebsock, thank you for providing model data, and for helpful conversations about dealing with satellite data. Natalie Tourville, thank you for keeping my computers working over the past few years. All of the members of the van den Heever research group have been a great help, and I think them all for the useful conversations and the fun ones. I need to express my gratitude to my friends and family who have supported me through this process, and haven't let a crazy, stressed out grad student scare them away.

Finally, I would like to dedicate this dissertation to my beloved wife Ling Wang, and my parents, sisters and brother. Without their patient love, encouragement and constant support, this research could not have been completed. Especially, I would like to give my special thanks to my wife Ling who has always worked to make my life happy and fulfilling

with love.

Abstract

This is where you write your abstract ...

Table of contents

List of figures

List of tables

MAP	Maximum-A-Posteriori
MPE	Most Probable Explanation
MMSE	Minimum Mean Squared Error
MPM	Maximum Posteriori Marginal
ERM	Empirical Risk Minimization
MRF	Markov Random Field
CRF	Conditional Random Field
HCRF	Hidden Conditional Random Field
SVM	Support Vector Machine
mi-SVM	microscopic Multiple-Instance SVM
MI-SVM	macroscopic Multiple-Instance SVM
LP	Linear Program
QP	Quadratic Program
L-BFGS	limited-memory Broyden-Fletcher-Goldfarb-Shanno
OWL-QN	Orthant-Wise Limited-memory Quasi-Newton
SGD	Stochastic Gradient Descent
FISTA	Fast Iterative Shrinkage-Thresholding
ADMM	Alternating Direction Method of Multipliers
CCCP	Concave-Convex Procedure
DC	Difference of convex functions
SA	Simulated Annealing
MCMC	Markov Chain Monte Carlo
CD	Contrastive Divergence
KL	Kullback-Leibler

BP	Belief Propagation
MPLP	Max-Product Linear Programming
TRWS	Sequential Tree-Reweighted Message Passing
LPQP	Linear and Quadratic Program relaxation
LPQP-U	LPQP with Uniform Penalty
LPQP-T	LPQP with Tree-weighted Penalty
IBFS	Incremental Breadth First Search
FVS	Feedback Vertex Set
HOG	Histogram of Oriented Gradients
SIFT	Scale-invariant feature transform
OCR	Optical Character Recognition
VOC	Visual Object Classes

References

[1] Y. Teng and K. Koike. Three-dimensional imaging of a geothermal system using temperature and geological models derived from a well-log dataset. *Geothermics*, 36:518–538, 2007. ISSN 03756505. doi: 10.1016/j.geothermics.2007.07.006.

Appendix A

R Code of Preamble

A.1 R Code Chunk Options

```
library(knitr)
### Speed  Up
options(replace.assign=TRUE, width = 112, digits = 3, max.print=72)
#  replace "=" with "<-"
#  R output can go to 72 characters per line before wrapping
#  print 3 significant digits,
#  If I ask to see a big matrix or something, only show 72 lines
### Fontsize: tell knitr to use smaller font for code chunks
opts_chunk$set(size='footnotesize')

### Tidy and Wrap and Highlight(Code Color)
opts_chunk$set(concordance=TRUE, tidy=FALSE, highlight=TRUE)

### Cache
opts_chunk$set(cache = TRUE)
opts_chunk$set(cache.path = "/home/tian/Dropbox/2data/cache/")

### Figure Setup
opts_chunk$set(fig.path = "/home/tian/Dropbox/3figs/phdthesis")
opts_chunk$set(dev = "pdf") # 0:pdf 0:png
#opts_chunk$set(fig.width = 8, )
opts_chunk$set(fig.align='center')
opts_chunk$set(fig.show='hold') #0: behind text, 1:hold in asis
opts_template$set(
  fig.3by3 = list(fig.width = 3, fig.height = 3),
  fig.5by5 = list(fig.width = 5, fig.height = 5),
  fig.6by3 = list(fig.width = 6, fig.height = 3),
  fig.6by4 = list(fig.width = 6, fig.height = 4),
  fig.6by5 = list(fig.width = 6, fig.height = 5),
  fig.6by6 = list(fig.width = 6, fig.height = 6),
  fig.7by4 = list(fig.width = 7, fig.height = 4),
  fig.7by5 = list(fig.width = 7, fig.height = 5),
```

```
  fig.7by6 = list(fig.width = 7, fig.height = 6),
  fig.7by7 = list(fig.width = 7, fig.height = 7)
)


#opts_chunk$set(error=FALSE)


#opts_chunk$set(warning = FALSE)
#opts_chunk$set(message=TRUE)  # 0:##
#opts_chunk$set(highlight=FALSE) # 0:TRUE

#opts_chunk$set(tidy=TRUE, size='normalsize')
#opts_knit$set(progress = F, verbose = F)
### export formats
## nocode
opts_chunk$set(echo=FALSE) # 0: TRUE
## no results
#opts_chunk$set(eval=FALSE) # control the export 0:TRUE, with figures
## no coments sign
opts_chunk$set(comment=NA)  # 0:##
opts_chunk$set(warning=FALSE)
opts_chunk$set(message=FALSE) #0:TRUE
opts_chunk$set(prompt=FALSE) # 0:TRUE >
```

A.2 R Variables and Fuctions in this Dissertaion

```
################################################################################
## R code chunk: Phd Thesis Variables and Functions
################################################################################
# Extent of Study area of Japan in Phd Thesis
xlimJP <- c(128.5, 146.5)
ylimJP <- c(30.2, 45.8)
certerJp <- c(137.5, 38)

# Coordinate Reference Systems of Japan
## Geographic Coordinate Reference Systems
#tokyoGRS <- "+init=epsg:4301"
tokyoGRS  <- "+proj=longlat +ellps=bessel +no_defs"
#wgs84GRS <- "+init=epsg:4326"
wgs84GRS  <-"+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"
#jgd2000GRS <- "+init=epsg:4612"
jgd2000GRS  <- "+proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +no_defs"

## Projected Coordinate Reference Systems
lccBessel <- "+proj=lcc +lat_1=32.8 +lat_2=43.2 +lat_0=38 +lon_0=137.5 +x_0=1000000 +y_0#=1000000 +e
## change the datum to WGS84 20140508
lccWgs84 <- "+proj=lcc +lat_1=32.8 +lat_2=43.2 +lat_0=38 +lon_0=137.5 +x_0=1000000 +y_0=1000000 +dat
```

```
#  +ellps=WGS84 +towgs84=0,0,0

codeDir <- "~/Dropbox/1code"
dataDir <- "~/Dropbox/2data"
dataData <- "~/Dropbox/2data/data"
dataRaw <- "~/Dropbox/2data/dataRaw"
dataPro <- "~/Dropbox/2data/dataProduct"
figsDir <- "~/Dropbox/3figs"
phd.slice100m  <- function(df,v,int=100) {
        ## creat slice factors
        intervals  <- as.numeric()
        ## TODO power     <- log(10,int)
        for (i in 1:length(v)) {
                if(round(v[i],-2) == round(v[i],-1)) {
                        ##intervals[i]  <- round(v[i],-2)%/%int
                        intervals[i]  <- round(v[i],-2)
                } else {
                        intervals[i] <- NA
                }
        }
        df$slice  <- intervals
        data  <- na.omit(df)
        ## Delete very rared duplicated case in one slice
        undup  <- function(df){
                df[!duplicated(df[,1]),] # Careful for ID
        }
        data.l  <- by(data, data$slice, undup)
        if (require(plyr)){
                data.d  <- rbind.fill(data.l)
        }
### for voxler soft export
        data.d$from <- data.d$slice - 50
        data.d$to <- data.d$slice + 50
        return(data.d)
}
phd.url.table  <- function(url, table = 1 ) {
        ## scrape the table to dataframe from a url website
      if(!require(XML)){
              installed.packages("XML")
      } else {
              url.doc <- htmlParse(url)
              url.l <- readHTMLTable(url.doc)
              url.d <- url.l[[table]]
      }
      return(url.d)
}
##  # url <- "http://ja.wikipedia.org/wiki/%E5%9C%B0%E7%86%B1%E7%99%BA%E9%9B%BB"
### phd.urltable(url,3)
phd.write.csv <- function(df) {
        ## write data.frame as csv and rds file into data folder using df name itself
        wd  <- getwd()
        setwd(dir= )(paste0(wd,"/data"))
```

```
        now <- format(Sys.time(), "_%y%m%d_%H%M%S")
        csvName  <- paste(deparse(substitute(df)), now, ".csv", sep = "")
        rdsName  <- paste(deparse(substitute(df)), now, ".Rds", sep = "")
        write.table(df, csvName, sep = ",", quote = F, row.names = F)
        saveRDS(df, rdsName)
        setwd(wd)
}
phd.write.csv2 <- function(df) {
        ## write data.frame as csv and rds file into data folder using df name itselt
        wd  <- getwd()
        setwd(dir= )(paste0(wd,"/data"))
        now <- format(Sys.time(), "_%y%m%d_%H%M%S")
        csvName  <- paste(deparse(substitute(df)), now, ".csv", sep = "")
        rdsName  <- paste(deparse(substitute(df)), now, ".Rds", sep = "")
        write.table(df, csvName, sep = ";", quote = F, row.names = F)
        saveRDS(df, rdsName)
        setwd(wd)
}
phd.saveshp.prj <- function(obj) {
        ## write obj as shp, rds and csv file
        wd  <- getwd()
        if(!file.exists("./data/shp")) {
                dir.create("./data/shp",recursive = TRUE)
        }
        setwd(paste0(wd, "/data/shp"))
        now <- format(Sys.time(), "_%y%m%d_%H%M%S")
        csvName  <- paste(deparse(substitute(obj)), now, ".csv", sep = "")
        rdsName  <- paste(deparse(substitute(obj)), now, ".Rds", sep = "")
        shpName  <- paste(deparse(substitute(obj)), now, sep = "")
        write.table(as.data.frame(obj), csvName, sep = ",", quote = F, row.names = F)
        saveRDS(obj, rdsName)
        if(require(rgdal)){
                writeOGR(obj, dsn = '.', layer = shpName, driver = "ESRI Shapefile")
        }
        setwd(wd)
}
phd.saveshp.geo <- function(obj) {
        ## write obj as shp, rds and csv file
        wd  <- getwd()
        if(!file.exists("./data/shp")) {
                dir.create("./data/shp",recursive = TRUE)
        }
        setwd(paste0(wd, "/data/shp"))
        now <- format(Sys.time(), "_%y%m%d_%H%M%S")
        csvName  <- paste0(deparse(substitute(obj)), now, ".csv")
        rdsName  <- paste0(deparse(substitute(obj)), now, ".Rds")
        shpName  <- paste0(deparse(substitute(obj)), now)
        kmlName  <- paste0(deparse(substitute(obj)), now)
        kmlDsn  <-  paste0("./", kmlName, ".kml")
        write.table(as.data.frame(obj), csvName, sep = ",", quote = F, row.names = F)
        saveRDS(obj, rdsName)
        if(require(rgdal)){
```

```
                writeOGR(obj, dsn = '.', layer = shpName, driver = "ESRI Shapefile")
                writeOGR(obj, dsn = kmlDsn, layer = kmlName, driver = "KML")
        }
        setwd(wd)
}
phd.ggsave <- function(name) {
        ## write data.frame as csv and rds file into data folder using df name itselt
wd  <- getwd()
setwd(paste0(wd,"/ggsave"))
now <- format(Sys.time(), "_%y%m%d_%H%M%S")
## eps/ps, tex (pictex), pdf, jpeg, tiff, png, bmp, svg and wmf (windows only).
pngName  <- paste0(deparse(substitute(name)), now, ".png")
tifName  <- paste0(deparse(substitute(name)), now, ".tiff")
pdfName  <- paste0(deparse(substitute(name)), now, ".pdf")
ggsave(pngName)
ggsave(tifName)
ggsave(pdfName)
setwd(wd)
}

phd.fishnet  <- function(x1,x2,y1,y2,rx=1000,ry=rx) {
        x.range.v <- seq(as.numeric(x1),as.numeric(x2),by = rx)
        y.range.v <- seq(as.numeric(y1),as.numeric(y2),by = ry)
        grid.m    <- outer(x.range.v, y.range.v, paste, sep = ",")
        grid.v    <- as.vector(grid.m)
        grid.d    <- as.data.frame(grid.v)
        fishnet.d <- data.frame(do.call('rbind', strsplit(as.character(grid.d[,1]), ',', fixed=TRU
        colnames(fishnet.d)  <- c("x","y")
}

phd.crsTransfer  <- function(df, x, y, xName, yName, fromCRS, toCRS) {
      ### 140510 transfer CRS in a dataframe fromat, x, y will be repaced in data.frame
       df$x  <- df[,which(colnames(df) == as.character(substitute(x)))]
       df$y  <- df[,which(colnames(df) == as.character(substitute(y)))]
      library(sp)
      library(rgdal)
      coordinates(df)  <- c("x", "y")
      proj4string(df)  <- CRS(fromCRS)
      df <- spTransform(df,CRS(toCRS))
      df <- as.data.frame(df)
      names(df)[which(names(df) == "x")] <- as.character(substitute(xName))
      names(df)[which(names(df) == "y")] <- as.character(substitute(yName))
      return(df)
    }
    phd.wraplm <- function(lm) {
      #wrap a lm resuts to data frame format
      cf <- coef(lm)
      tinfo <- summary(lm)$coefficients[2, c(2, 4)]
      r2 <- summary(lm)$r.squared
      data.frame(intercept = cf[1], slope = cf[2], n = length(resid(lm)),
                 slope.se = tinfo[1], pval = tinfo[2], Rsq = r2)
    }
```

```
    #20140524
phd.voxler.collars  <- function(ID, Easting, Northing, Elevation, Azimuth = 0, Dip = 0, Depth) {
      ID  <- ID
      well  <- as.data.frame(ID)
      well$Easting  <- Easting
      well$Northing  <- Northing
      well$Elevation  <- Elevation
      well$Azimuth  <- Azimuth
      well$Dip  <- Dip
      well$Depth  <- Depth
      well[sort(well$ID),]
      return(well)
    }
phd.voxler.trajectories  <- function(ID, MD, Azimuth = 0, Inclination = 0) {
      ID  <- ID
      well  <- as.data.frame(ID)
      well$MD  <- MD
      well$Azimuth  <- Azimuth
      well$Inclination  <- Inclination
      well[sort(well$ID),]
      return(well)
    }
phd.voxler.samples  <- function(ID, From, To, V1, ...) {
      ID  <- ID
      well  <- as.data.frame(ID)
      ## From to can get from phd.slice100m
      well$From  <- From
      well$To  <- To
      well$V1  <- V1
      well[sort(well$ID),]
      return(well)
    }
    ## Add linear function and R2 on the line in ggplot
    #http://stackoverflow.com/questions/7549694/ggplot2-adding-regression-line-equation-and-r2-on-gr
lm_eqn = function(m) {

        l <- list(a = format(coef(m)[1], digits = 2),
                  b = format(abs(coef(m)[2]), digits = 2),
                  r2 = format(summary(m)$r.squared, digits = 3));

        if (coef(m)[2] >= 0)  {
            eq <- substitute(italic(y) == a + b %.% italic(x)*","~~italic(r)^2~"="~r2,l)
        } else {
            eq <- substitute(italic(y) == a - b %.% italic(x)*","~~italic(r)^2~"="~r2,l)
        }

        as.character(as.expression(eq));
    }
phd.geocode <- function(nameLst) {
          # get longitude, latitude, and Name of given place
          lonlata  <- ggmap::geocode(nameLst, output = "latlona")
          names <- do.call("rbind", strsplit(lonlata[,3], ","))
```

```
          name  <- Hmisc::capitalize(names[,1])
          lonlatn  <- cbind(lonlata[,c(1,2)], name)
          return(lonlatn)
  }

phd.spatialize.wgs84 <- function(df, x, y) {
         # spatialize a data frame to spdf
         d  <- df
         d$x  <- d[,which(colnames(d) == as.character(substitute(x)))]
         d$y  <- d[,which(colnames(d) == as.character(substitute(y)))]
         wgs84GRS <- "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"
         coords  <- d[, c("x","y")]
         m  <- as.matrix(coords) #sp need numeric matrix
         mode(m)  <- "numeric"
         sp  <- sp::SpatialPoints(m, proj4string = sp::CRS(wgs84GRS))
         spdf <- sp::SpatialPointsDataFrame(m, data = d, proj4string=sp::CRS(wgs84GRS))
         return(spdf)
  }
phd.bbox  <- function(xmin,xmax,ymin,ymax,crs = NA){
        ### Make a bbox rect SPDF from LL and TR point.
        Polygon  <- sp::Polygon(cbind(c(xmin, xmin, xmax, xmax, xmin),
                           c(ymin, ymax, ymax, ymin, ymin)))
        Polygons  <- sp::Polygons(list(Polygon), 1) #ID for row.names = 1
        SP  <- sp::SpatialPolygons(list(Polygons))
        data_d  <- data.frame(name = "bbox", row.names= row.names(Polygons))
        SPDF  <- sp::SpatialPolygonsDataFrame(SP, data = data_d)
        proj4string(SPDF)  <- crs
        return(SPDF)
}
phd.grid  <- function(SPDF,x, y = x, type = "regular"){
        sp  <- sp::spsample(SPDF,type = type, cellsize = c(x,y), offset =c(0.5, 0.5))
        df  <- data.frame(id = as.factor(1:length(sp)))
        spdf  <- sp::SpatialPointsDataFrame(sp, data = df)
}

phd.sp2SPDF  <- function(sp) {
        df  <- data.frame(as.factor(1:length(sp)))
        spdf <- sp::SpatialPointsDataFrame(sp, df)
        grd  <- spdf
        sp::gridded(grd)  <- TRUE
        grd_SPDF  <- as(grd, "SpatialPolygonsDataFrame")
        return(grd_SPDF)
}

phd.spdf2SPDF  <- function(spdf) {
        grd  <- spdf
        sp::gridded(grd)  <- TRUE
        grd_SPDF  <- as(grd, "SpatialPolygonsDataFrame")
        return(grd_SPDF)
}

phd.bigpolys  <- function(x, area){
```

```
        polys <- lapply(x@polygons , slot , "Polygons" )
        area_fun  <- function(y) {
           sapply(y@Polygons, function(z) z@area)
        }
        areas_l <- lapply(x@polygons, area_fun)
        bigpolys <- sapply(areas_l, function(a) which(a > area))
        return(bigpolys)
}
```