Extensão para visualização temporal de dados no TerraView

Carolina Galvão dos Santos[1], **Matheus Cavassan Zaglia**[2]

[1]Instituto Nacional de Pesquisas Espaciais – INPE
Caixa Postal 515 – 12227-010 – São José dos Campos – SP, Brasil

carolinagalvaosantos@gmail.com, mzaglia@dpi.inpe.com

Resumo. *Uma das dificuldades apresentadas em alguns Sistemas de Informações Geográficas é a possibilidade de visualizar dados com base em sua linha do tempo, principalmente quando estes possuem localizações fixas. Portanto este trabalho propõe-se desenvolver uma extensão para o TerraView utilizando a TerraLib capaz de suprir esta necessidade de realizar a visualização de dados com base no tempo.*

Abstract. *One of the difficulties presented in some Geographic Information Systems is the possibility to visualize data based on their timeline, especially when they have fixed locations. Therefore, this work proposes to develop an extension for TerraView using TerraLib capable of satisfying the need to perform data visualization based on time.*

1. Introdução

Os Sistemas de Informações Geográficas (SIG) são sistemas criados para facilitar a análise, gestão, representação do espaço e dos fenômenos que nele ocorrem, sendo possível através de procedimentos computacionais, informações espaciais e os próprios recursos humanos. Eles permitem realizar análises complexas, ao integrar dados de diversas fontes e ao criar bancos de dados geo-referenciados, tornando ainda possível automatizar a produção de documentos cartográficos [Câmara 2001]. Estes grandes e poderosos sistemas trouxeram ao mundo da geoinformação a utilização dos computadores como instrumento de representação de dados espacialmente referenciados.

Juntamente a esses sistemas surgiram questões a serem discutidas sobre como se representar computacionalmente o espaço geográfico ou o mundo real da melhor forma possível. Além de questões relacionadas a representação do espaço, surgiu a necessidade de inclusão da variável tempo as representações. Enquanto os SIGs convencionais são desenvolvidos para análises espaciais estáticas, ainda notamos dificuldades na modelagem dos sistemas para se representar e realizar análises bem sucedidas de fenômenos temporais. Atualmente, ainda não encontramos uma padronização ou uma metodologia completa na inclusão do *tempo* nestes sistemas. Entretanto existem estudos, pesquisas e até mesmo ferramentas implementadas ou em fase de implementação que auxiliam na representação de dados temporais em ambientes de sistemas geográficos.

Um grande exemplo de SIG é o TerraView (Figura 1) construído com base na biblioteca TerraLib, que é uma biblioteca de código aberto escrita na linguagem C++, que da suporte a construção de aplicações geográficas. A TerraLib disponibiliza o módulo *Spatio Temporal*, uma extensão baseada em um modelo de dados temporais proposto por

Karine Ferreira que define um conjunto de tipos de dados e operações capazes de representar dados espaço-temporais de diferentes aplicações [Ferreira 2014] (Figura 2). Estes conceitos foram implementados na biblioteca TerraLib, além de outras funcionalidades que estão sendo trabalhadas atualmente pela equipe de desenvolvimento.

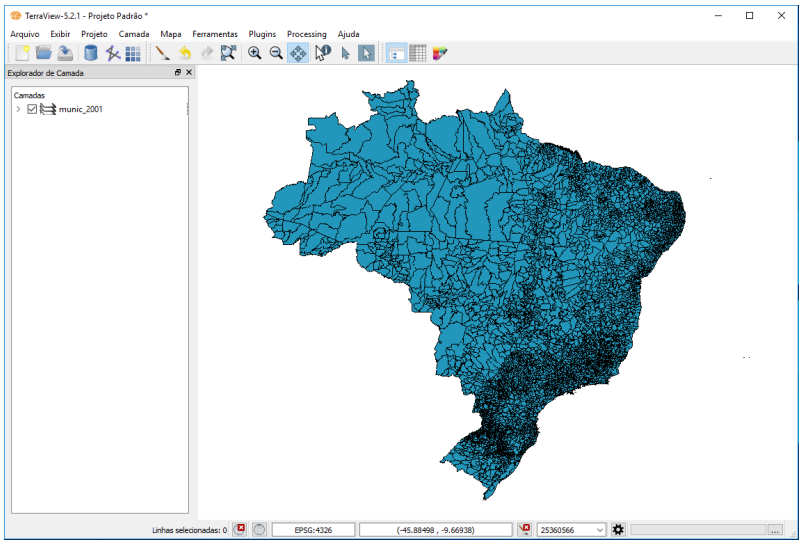

Figura 1. Aplicação TerraView versão 5.2.1

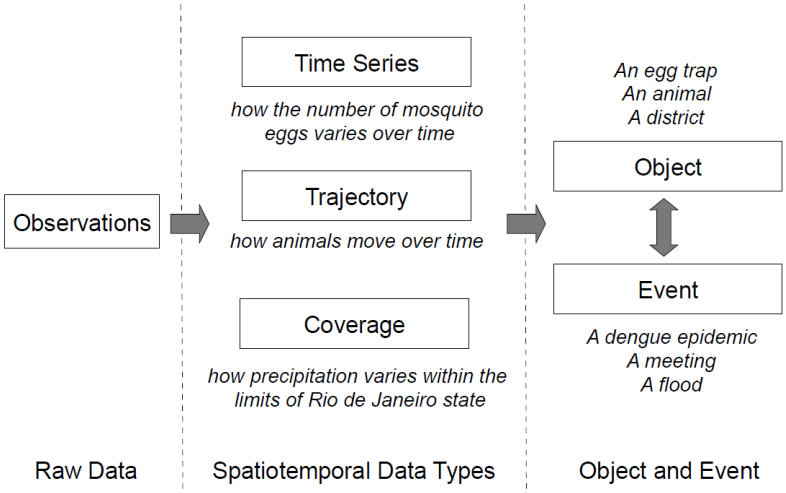

Figura 2. Modelo de dados espaço-temporais proposto por Karine [Ferreira 2014]

Atualmente possibilitar a visualização de dados com base em sua linha do tempo é uma das grandes dificuldades apresentadas em parte dos SIGs, principalmente quando estes possuem localizações fixas como, por exemplo, um banco de dados de amostras que possui diversos registro de um único objeto em diferentes datas, mas em cada data este objeto pode sofrer alterações em algum de seus atributos. Portanto com base na contextualização acima este trabalho propõe o desenvolvimento de uma extensão para o TerraView que possibilite visualizar dinamicamente dados temporais.

2. Metodologia

2.1. Materiais

A extensão para o TerraView, foi implementada como um plugin da biblioteca TerraLib de versão 5.2.1. É basicamente composta por elementos de layout da ferramenta Qt [Company and Qt 2017] de versão 5.4.1 que permite desenvolvimento multiplataforma de criação aplicações com interfaces gráficas, e o módulo *Data Access* da TerraLib, que permite acessado a base de dados de forma genérica. Foi desenvolvido na linguagem de programação C++, utilizando como ambiente de desenvolvimento o Visual Studio 12 2013 através do sistema operacional Windows 10. A extensão é compatível com todos os sistemas operacionais que a TerraLib oferece suporte.

2.2. Conjunto de Dados

O conjunto de dados que serão utilizados para a validação da extensão são amostras de pontos distribuídos por todo o espaço geográfico, e amostras com localização fixa, mas que variam em seus demais atributos conforme o decorrer do tempo. Os dados foram obtidos através uma base de dados relacional, utilizada pelo projeto E-Sensing [INPE 2016] para realização de testes e utilizando *shape files* da base de dados do programa queimadas [DGI 2016]. Vale ressaltar que a extensão necessita somente de dados vetoriais com atributos temporais.

3. Implementação

Esta extensão irá permitir a partir de um dado de amostras visualizar os registros de um objeto de acordo com uma linha do tempo em um período determinado pelo usuário, tornando possível produzir uma animação do objeto ao decorrer do tempo. Foi chamada de "Time Viewer" e construída como um plugin da biblioteca TerraLib, onde será disponibilizada utilizando a interface da aplicação TerraView para acoplar o *layout* da ferramenta, conforme protótipo a seguir (Figura 3).

Figura 3. Protótipo do plugin de visualização acoplado ao TerraView

Foi criado um módulo que será compilado juntamente com a TerraLib. Este módulo é constituído pelos arquivos apresentados na Figura 4. Ao ser compilado com

será gerado uma biblioteca dinâmica que será carregada pela TerraLib atráves do sistema de *plugins*.

```
(...)
terralib
 |-- (...)
 '-- timeviewer
     '-- ui
       |-- Config.h
       |-- Plugin.cpp
       |-- Plugin.h
       |-- TimeViewerDockWidget.cpp
       '-- TimeViewerDockWidget.h
```

Figura 4. Árvore de arquivos implementados do TimeViewer

O arquivo *Plugin.h* descreve a classe *te::tv::Plugin* que herda os métodos *startup*(Figura 5) e *shutdown*(Figura 6) da classe *te::core::Plugin*. Em sua implementação feita no arquivo *Plugin.cpp*, o método *startup* é responsável por registrar no menu de *Plugins* do TerraView uma opção que cria e exibe a janela da extensão. Já o método *shutdown* realiza a remoção dos ponteiros criados anteriormente.

```
void te::tv::Plugin::startup()
{
 if(m_initialized)
   return;

 m_timeViewerAction = new QAction(this);
 m_timeViewerAction->setText("Time Viewer");
 m_timeViewerAction->setObjectName("TimeViewer");
 m_timeViewerAction->setIcon(QIcon::fromTheme("colored-calendar"));

 connect(m_timeViewerAction, SIGNAL(triggered()), this,
       SLOT(onTimeViewerActionTriggered()));

 te::qt::af::AppCtrlSingleton::getInstance().getMenu("Plugins")->addAction(
    m_timeViewerAction);

 m_initialized = true;
}
```

Figura 5. Método de inicialização do plugin TimeViewer

```
void te::tv::Plugin::shutdown()
{
  if(!m_initialized)
    return;

  delete m_tvDock;
  delete m_timeViewerAction;

  m_initialized = false;
}
```

Figura 6. Método de finalização do plugin TimeViewer

A janela da extensão possui um *combo box* com uma lista dos *layers* que permite a seleção de um *layer* previamente adicionado pelo usuário. Ao selecionar um item do *combo box* de *layers*, outros dois *combo boxes* são preenchidos com todos os atributos do item selecionado, nestes serão escolhidos um atributo para data inicial e/ou um atributo para data final. Também é necessário informar um formato para as datas. Com isso, é possível clicar no botão *apply*, onde será realizada uma consulta através da interface de *query* da TerraLib, que irá recuperar todas as datas do *layer* com base nos atributos informados nos *combo boxes*. Essas datas são inseridas em uma lista, onde são removidos os valores repetidos e os demais são ordenados em ordem crescente (Figura A.1).

Para navegar pela linha do tempo é utilizado um *slider*, que quando é movimentado, é realizada uma chamada para um método, onde consultas são montadas dinamicamente de acordo com os atributos de data inicial e final escolhidos anteriormente (Figura A.2). A extensão também permite escolher se a visualização será acumulativa, ou seja, a cada mudança na visualização os resultados antigos são mantidos na tela. A interface final da extensão pode ser vista na Figura 7.

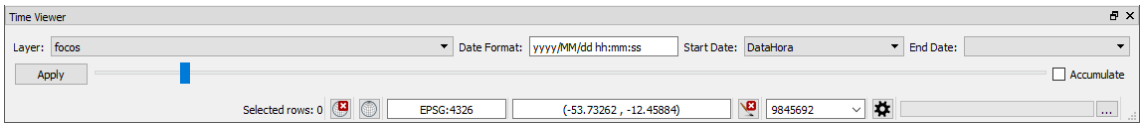

Figura 7. Tela implementada da extensão Time Viewer

4. Resultados

Para validar a implementação, foram feitos testes utilizando algumas amostras temporais obtidas através das bases de dados citadas anteriormente. Foram selecionados dois tipos de dados de exemplo, onde o primeiro possui amostras com dados de focos de incêndios em toda a região do Brasil, cada foco possui um atributo de data e hora únicas, sendo necessário que o usuário defina este formato de data através da interface gráfica e selecione o atributo da tabela que corresponda a data no campo de data inicial, ignorando o campo de data final conforme Figura 8.

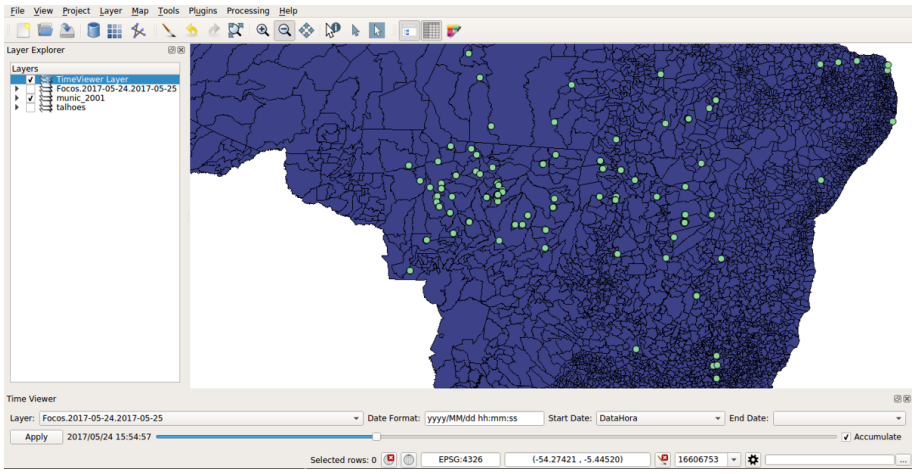

Figura 8. Utilizando o Time Viewer para visualização de dados amostrais de focos de incêndios

O segundo dado de exemplo possui amostras com dados de talhões da região de Cabo Verde no estado de Mato Grosso, onde cada talhão possui dois atributos de data sendo eles data inicial e data final, o usuário deve definir o formato de data através da interface gráfica e selecionar o atributo da tabela que corresponda a data inicial e o atributo que corresponda a data final, conforme Figura 9.

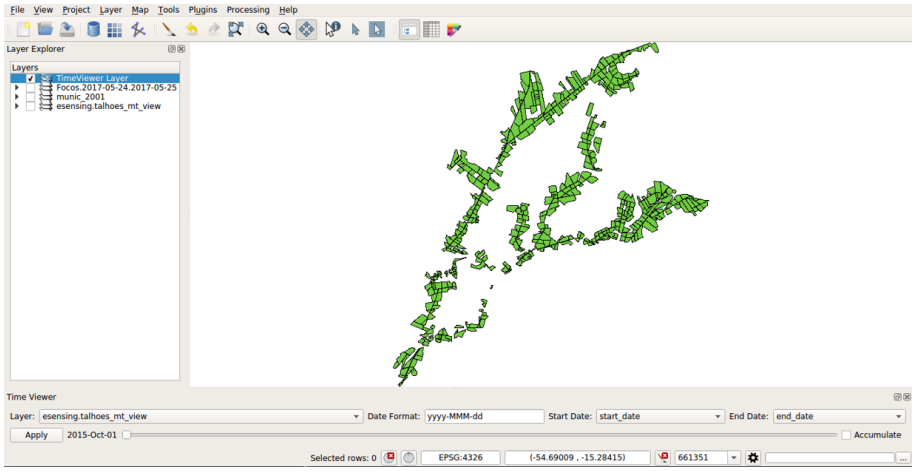

Figura 9. Utilizando o Time Viewer para visualização de talhões na região Cabo Verde - MT

5. Conclusões e Trabalhos Futuros

Conforme podemos notar a quantidade de dados geográficos com características temporais tende a crescer constantemente gerando um grande conjuntos de dados (Big Data), e com isso, dificulta-se a utilização destes dados em SIGs, uma vez que não há um padrão completo para dados temporais. Este é um problema que pode ser evitado através de iniciativas como a proposta realizada por este trabalho. A ferramenta desenvolvida buscou facilitar a visualização dos dados temporais de maneira simples, para que um usuário sem conhecimentos técnicos possa visualizar seus dados.

Os objetivos deste trabalho foram alcançados e com isso novas ideias foram concebidas para que a ferramenta possa auxiliar o usuários com novos recursos como, permitir o monitoramento de um único objeto e sua linha do tempo, permitir a criação de uma animação que possa ser executada no TerraView e salva na máquina do usuário, utilizar a ferramenta de legendas da própria aplicação para facilitar a visualização de alterações que ocorrem no tempo para atributos de amostras com localizações fixas no espaço e a integração com o módulo *Spatio Temporal* da TerraLib.

6. Agradecimentos

Gostaríamos de agradecer o Dr. Gilberto Ribeiro de Queiroz e a Dra. Lúbia Vinhas pelo apoio, confiança e orientações durante nossa jornada.

Referências

Câmara, G.; Davis, C. M. A. D. J. (2001). Introdução à Ciência da Geoinformação. `http://www.dpi.inpe.br/gilberto/livro/introd/`. [Online; accessed 25-May-2017].

Company and Qt, D. T. (2017). Qt - Software development made smarter. https://www.qt.io/. [Online; accessed 06-Jun-2017].

DGI, I. (2016). Portal do Monitoramento de Queimadas e Incêndios. http://www.inpe.br/queimadas. [Online; accessed 25-May-2017].

Ferreira, K. R., C. G. M. A. M. V. (2014). An algebra for spatiotemporal data: from observations to events. *Transactions in GIS*, page 253–269.

INPE (2016). e-sensing: big Earth observation data analytics for LUCC. http://www.esensing.org/. [Online; accessed 06-Jun-2017].

A. Figuras

```
te::da::FromItem* fromItem = new te::da::DataSetName(m_datasetName);
te::da::From* from = new te::da::From;
from->push_back(fromItem);

te::da::Fields* fields = new te::da::Fields;
fields->push_back(new te::da::Field(m_start));

if(!m_end.empty())
  fields->push_back(new te::da::Field(m_end));

te::da::Select* select = new te::da::Select();
select->setFields(fields);
select->setFrom(from);

try
{
  auto result = m_dsPtr->query(select);

  result->moveBeforeFirst();

  QSet<QDateTime> qDateSet;
  QLocale loc(QLocale::English, QLocale::UnitedStates);

  while(result->moveNext())
  {
    std::string startDate = result->getAsString(m_start);
    QDateTime qStartDate =
        loc.toDateTime(startDate.c_str(), m_ui->m_dateFormat->text());
    qDateSet.insert(qStartDate);
    if(!m_end.empty())
    {
      std::string endDate = result->getAsString(m_end);
      QDateTime qEndDate =
          loc.toDateTime(endDate.c_str(), m_ui->m_dateFormat->text());
      qDateSet.insert(qEndDate);
    }
  }

  m_dates.clear();
  m_dates = qDateSet.toList();
  qSort(m_dates);
}
catch(te::Exception& e)
{
  QMessageBox::warning(0, "TimeViewer", e.what());
  return;
}
m_ui->m_timeSlider->setMaximum(m_dates.size() - 1);
```

Figura A.1. Código fonte da construção da lista de datas que serão utilizadas pelo slider

```
te::da::FromItem* fromItem = new te::da::DataSetName(m_datasetName, alias);
te::da::From* from = new te::da::From;
from->push_back(fromItem);

te::da::Fields* fields = new te::da::Fields;

auto properties = m_dsPtr->getPropertyNames(m_datasetName);
for(auto property : properties)
{
  if(property != "FID")
    fields->push_back(new te::da::Field(alias + "." + property));
}

auto dsTypePtr = m_dsPtr->getDataSetType(m_datasetName);

te::gm::GeometryProperty* gp =
te::da::GetFirstGeomProperty(dsTypePtr.get());

fields->push_back(new te::da::Field(alias + "." + gp->getName()));

te::da::Where* where = nullptr;
te::da::PropertyName* start = new te::da::PropertyName(m_start);
te::da::Expression* expression = nullptr;
m_accumulate = m_ui->m_accumulate->isChecked();
if(m_accumulate)
{
  if(m_end.empty())
    expression = new te::da::LessThanOrEqualTo(start, literal);
  else
    expression = new te::da::LessThanOrEqualTo(
        new te::da::PropertyName(m_end), literal);

  where = new te::da::Where(expression);
}
else
{
  if(m_end.empty())
  {
    expression = new te::da::EqualTo(start, literal);
    where = new te::da::Where(expression);
  }
  else
  {
    expression = new te::da::GreaterThanOrEqualTo(start, literal);
    te::da::PropertyName* end = new te::da::PropertyName(m_end);
    te::da::Expression* expression2 =
        new te::da::LessThanOrEqualTo(end, literal->clone());

    te::da::And* andExpression = new te::da::And(expression, expression2);

    where = new te::da::Where(andExpression);
  }
}

te::da::Select* select = new te::da::Select();
select->setWhere(where);
select->setFrom(from);
select->setFields(fields);

m_layer->setQuery(select);
m_layer->computeExtent();
```

Figura A.2. Código fonte da construção das consultas realizadas através da movimentação do slider