UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

FACULDADE DE ARQUITETURA

PROGRAMA DE PÓS-GRADUAÇÃO EM DESIGN

Luciano Santos da Silva

DESIGN PARAMÉTRICO A PARTIR DA DIGITALIZAÇÃO 3D DE GEOMETRIAS DA NATUREZA COM PADRÃO DE CRESCIMENTO ESPIRAL

Dissertação de Mestrado

Porto Alegre

2017

LUCIANO SANTOS DA SILVA

Design paramétrico a partir da digitalização 3D de geometrias da natureza com padrão de crescimento espiral

Dissertação de Mestrado submetido ao Programa de Pós-Graduação em Design da Universidade Federal do Rio Grande do Sul, para a obtenção do título de Mestre em Design.

Orientador: Prof. Dr. Fabio Pinto da Silva

Porto Alegre

2017

CIP - Catalogação na Publicação

Silva, Luciano Santos da
Design paramétrico a partir da digitalização 3D de geometrias da natureza com padrão de crescimento espiral. / Luciano Santos da Silva. -- 2017.
114 f.

Orientador: Fabio Pinto da Silva.

Dissertação (Mestrado) -- Universidade Federal do Rio Grande do Sul, Escola de Engenharia, Programa de Pós-Graduação em Design, Porto Alegre, BR-RS, 2017.

1. Digitalização 3D. 2. Biônica. 3. Design Paramétrico. 4. Engenharia Reversa. I. Silva, Fabio Pinto da, orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

Luciano Santos da Silva

DESIGN PARAMÉTRICO A PARTIR DA DIGITALIZAÇÃO 3D DE GEOMETRIAS DA NATUREZA COM PADRÃO DE CRESCIMENTO ESPIRAL

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Design, e aprovado em sua forma final pelo Programa de Pós-Graduação em Design da UFRGS.

Porto Alegre, 20 de março de 2017.

Prof. Dr. Régio Pierre da Silva

Coordenador do Programa de Pós-Graduação em Design da UFRGS

Banca Examinadora:

Orientador: **Prof. Dr. Fabio Pinto da Silva**

Departamento de Design e Expressão Gráfica (DEG)

Prof. Dr. Luiz Carlos Gertz

ULBRA / ENGENHARIA MECÂNICA – Examinador Externo

Prof. Dr. Rodrigo Antonio Marques Braga

UFSC / CCE / EGR – Examinador Interno

Prof. Dr. Luis Henrique Alves Cândido

Departamento de Design e Expressão Gráfica (DEG) – Examinador Interno

AGRADECIMENTOS

À minha família, em especial a minha esposa Marilu e meu filho Maurício pelo apoio e incentivo. Aos meus Pais Nilo e Fátima e meus irmãos pelo incentivo e pensamentos positivos.

Ao meu orientador, Professor Dr. Fabio Pinto da Silva, pelas orientações e pela oportunidade de realizar este trabalho com tanta aplicação.

Aos Professores do Programa de Pós-Graduação em Design e Tecnologia, em especial aos professores Wilson Kindlein Júnior, Liane Roldo, Lauren da Cunha Duarte, Luis Henrique Alves Cândido, Everton Sidnei Amaral da Silva, Underléa Miotto Bruscato, Joyson Luiz Pacheco, Rodrigo Antonio Marques Braga, Fabio Gonçalves Teixeira, pelos ensinamentos e considerações nas etapas desta pesquisa.

Aos amigos Gabriel, Eloisa, Márcia, Jaqueline, Ângela, Denise, Elisa, Jorge, Mariana, Paulo Victor, Silvie, Ulisses, Viviane, Israel, Hilton, Suzana, Guilherme, João Rogério, Mariana, Wagner, Yuri Walter, Juliana, Thiago, Fernando, João Nercimar, Laura, Ivan, Marcelo e ao colega Félix Bressan pelos ensinamentos iniciais do Grasshopper.

Finalmente, gostaria de agradecer à UFRGS pelo ensino de qualidade a CAPES que por meio do seu apoio financeiro, possibilitou esta pesquisa.

A todos mais que eu não tenha citado nesta lista de agradecimentos, mas que de uma forma ou de outra contribuíram para a minha dissertação.

Este trabalho foi realizado com o apoio do CNPq e CAPES.

RESUMO

SILVA, L. S. **Design paramétrico a partir da digitalização 3D de geometrias da natureza com padrão de crescimento espiral**. Dissertação (Mestrado em Design) – Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

A modelagem de geometrias da natureza pode ser um processo complexo devido ás características orgânicas dos elementos. Propõe-se com essa dissertação identificar geometrias espaciais que sigam o padrão de crescimento espiral observado na natureza, utilizando as Tecnologias 3D como ferramentas para o processo de projeto. Para a execução do trabalho foram investigadas os Métodos de Biônica, Crescimento Espiral e a Sequência de Fibonacci, Engenharia Reversa e Design Paramétrico. O processo de representação dos elementos foi realizado em conformidade com a Metodologia para o Desenvolvimento de Produtos Baseados no Estudo da Biônica com o acréscimo das tecnologias de digitalização tridimensional e de processamento de nuvem de pontos, complementado pela parametrização de superfícies à base de curvas. Foram utilizados três processos para modelagem de curvas paramétricas representadas (i) pelo desenho de linhas sobre a malha digitalizada em 3D, (ii) por programação visual no *software Grasshopper* e (iii) por programação com *scripts Python.* Foi avaliada como melhor alternativa para o Design Paramétrico a utilização da programação visual otimizada com a programação por *scripts*, a qual apresentou melhor aproximação entre as curvas analisadas. Estudos de casos realizados com elementos da natureza (abacaxi e pinha) demonstraram a viabilização do método. Desta maneira a sistematização do conhecimento permitirá a proposição de um modelo paramétrico baseado na Biônica para fase inicial de inspiração e concepção de alternativas do projeto de produto.

Palavras-chave: Digitalização 3D, Biônica, Design Paramétrico, Engenharia Reversa.

ABSTRACT

SILVA, L. S. **Design paramétrico a partir da digitalização 3D de geometrias da natureza com padrão de crescimento espiral**. Dissertação (Mestrado em Design) – Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

Modeling the geometries of nature can be a complex process due to the organic characteristics of the elements. It is proposed with this dissertation to identify spatial geometries that follow the pattern of spiral growth observed in nature, using 3D Technologies as tools for the design process. For the execution of the work were investigated the Bionics, Spiral Growth and Fibonacci Sequence, Reverse Engineering and Parametric Design. The process of representation of the elements was carried out in accordance with the Methodology for the Development of Products Based on the Study of the Bionics with the addition of the technologies of three-dimensional digitization and processing of cloud of points, complemented by the parameterization of surfaces based on curves. Three methods were used for modeling parametric curves represented by (i) the drawing of lines on the 3D scanned mesh, (ii) by visual programming in the Grasshopper software and (iii) by programming with Python scripts. It was evaluated as the best alternative for Parametric Design the use of optimized visual programming with programming by scripts, which presented better approximation between the analyzed curves. Case studies carried out with nature elements (pineapple and pine cone) demonstrated the viability of the method. In this way the systematization of the knowledge will allow the proposition of a parametric model based on the Bionics for the initial phase of inspiration and design of alternatives of the product design.

Keywords: 3D Scanning, Bionics, Parametric Design, Reverse Engineering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

2D	Bidimensional
3D	Tridimensional
ABNT	Associação Brasileira de Normas Técnicas
CAD	*Computer Aided Designer* (Projeto Auxiliado por Computador)
CAE	*Computer Aided Engineering* (Engenharia (simulação) Auxiliado por Computador)
CAI	*Computer Aided Inspection* (Inspeção Auxiliada por Computador)
CAM	*Computer Aided Manufacturing* (Manufatura Auxiliada por Computador)
CAx	*Computer Aided Applications* (Aplicações Auxiliada por Computador)
CCD	*Charge Coupled Device* (Dispositivo de Carga Acoplada)
CNC	*Computerized Numerical Control* (Comando Numérico Computadorizado)
CMM	*Coordinate Measuring Machine* (Máquina de Medição por Coordenadas)
DNA	*Deoxyribonucleic Acid* (Ácido Desoxirribonucléico)
LdSM	Laboratório de Design e Seleção de Materiais
mm	Milímetros
µm	Micrometros
UFRGS	Universidade Federal do Rio Grande do Sul
STL	*STereoLithography*

SUMÁRIO

1 INTRODUÇÃO

É notório que os sistemas de computação evoluem com extrema rapidez, disponibilizando inúmeras ferramentas para os designers desenvolverem e apresentarem seus projetos, principalmente pela popularização e sucesso nas últimas décadas das soluções apresentadas pelos *softwares* de computação gráfica. Atender as demandas por inovação e criatividade aumentam a complexidade dos projetos fazendo com que os designers, arquitetos e engenheiros busquem novas formas de inspiração. Uma maneira para atender estas demandas é a inspiração nas formas complexas da natureza para a concepção do projeto, para o desenho e para a simulação do crescimento e da interação destes elementos naturais com as tecnologias 3D, possibilitando assim a geração de alternativas para a inovação dos projetos (BENTLEY & KUMAR, 1999).

Um dos avanços no design está na introdução das ferramentas paramétricas no projeto assistido por computador (CAD). Este método de concepção de projetos acompanha a tendência da evolução da linguagem dos sistemas computacionais, simplificando a modelagem tridimensional na fase inicial de concepção de projeto, pelo meio de algoritmos e operações matemáticas complexas, trazendo uma nova dinâmica de concepção para o design de produto (SALGUEIRO, 2011). A dinâmica dos projetistas está no uso do *software* de CAD com o objetivo de atender as expectativas de criação, concepção e desenvolvimento demandantes do mercado. Aproveitar as ferramentas de modelagem paramétrica possibilita controlar e registrar informações e atender os requisitos dos parâmetros de projeto, transformando, por intermédio do design paramétrico, os requisitos em alternativas possíveis (KRISH, 2011).

A questão é de como utilizar nas fases iniciais do projeto as ferramentas de CAD para explorar as possibilidades da subjetividade da criação. Uma das soluções apresentadas é o uso do Design Paramétrico aplicado para explorar milhares de possibilidades de design dentro do ambiente de CAD (KRISH, 2011). O implemento das ferramentas 3D com o Design Paramétrico parte da base do modelo gerando diversas possibilidades de projeto por meio de um conjunto de regras e algoritmos planejados (COSTA, 2008). Com a utilização do método é possível melhorar a

capacidade de explorar as variações quando o projeto ainda está em formulação, representando um importante recurso de decisão para otimizar o resultado do projeto (KRISH, 2011). Estudos realizados por Celani (2011) e Florio (2012) evidenciam a possibilidade do uso das novas tecnologias e dos programas paramétricos no estágio inicial de concepção do projeto de arquitetura, permitindo a produção de formas livres de maneira rápida e com dinamismo.

Para geração das alternativas de projeto na fase de concepção é fundamental a participação da equipe de projeto liderado por um projetista, utilização de um método e ferramentas que permitam a manifestação da criatividade (DETANICO et al., 2010). Assim, esta pesquisa situa-se na área da biônica, fundamentando a inovação em design aplicado na concepção de produtos baseados em sistemas naturais. Propõe-se o uso de tecnologias 3D para o estudo da natureza e parametrização de geometrias com vistas ao desenvolvimento de produtos.

Para esta pesquisa a Engenharia Reversa é considerada uma técnica primordial, indispensável para o estudo e reconstrução tridimensional de elementos da natureza. A aplicação das tecnologias 3D de digitalização e parametrização em software 3D constituem uma potencial ferramenta para a geração de superfícies para representação de elementos da natureza.

1.1 Problematização e Justificativa

Modelar geometricamente um elemento da natureza é, em geral, um processo não trivial devido à complexidade geométrica da maioria dos elementos. Manter o realismo visual é um grande desafio para a computação gráfica, dada a extrema complexidade que estes tipos de elementos podem vir a ter. É de crucial importância, no entanto, poder gerar com realismo estes objetos, já que os elementos da natureza estão presentes em grande parte de produções que envolvem computação gráfica (BENTLEY & KUMAR, 1999).

O problema é a limitação das informações sobre a aplicação das tecnologias 3D associadas aos elementos da natureza para geração de soluções paramétricas

em projeto de produto. Neste sentido, pergunta-se como utilizar as ferramentas de Digitalização 3D, Design Paramétrico e Engenharia Reversa para interpretar as formas da natureza (Biônica) e modelar alternativas paramétricas para o projeto de produto. Há a carência de uma metodologia de design paramétrico que permita relacionar essas ferramentas de maneira que possibilitem a reconstrução tridimensional de superfícies externas do modelo analisado. O uso dessas tecnologias ainda é muitas vezes restrito, devido ao alto nível exigido de conhecimento sobre modelagem tridimensional. Analisar a malha da superfície de um elemento orgânico digitalizado e identificar um padrão de crescimento requer habilidade e metodologia para simplificar e facilitar a etapa de modelagem por *software* de CAD. Assim, o presente trabalho justifica-se por propor uma metodologia partindo da digitalização 3D, capaz de relacionar o padrão da natureza com o método de concepção de produto utilizando os recursos da Engenharia Reversa e das tecnologias 3D.

1.2 Objetivo Geral

O objetivo deste trabalho é propor um algoritmo para parametrização de geometrias espaciais que sigam o padrão de crescimento espiral observado na natureza, a partir da digitalização 3D, e representa-las em *software* CAD.

1.3 Objetivos Específicos

- Revisar a literatura acerca dos métodos aplicados à Biônica e propor a inserção de uma etapa de engenharia reversa para análise de formas;
- Estudar o padrão de crescimento espiral e a sequência de Fibonacci para aplicação em uma etapa de design paramétrico;
- Selecionar uma forma da natureza representativa do padrão de crescimento espiral para estudo piloto;
- Aplicar a tecnologia de digitalização 3D para aquisição e estudo da forma;

- Propor o uso de ferramentas CAD (*Rhinoceros* e *Grasshopper*) aliadas à digitalização 3D, visando a parametrização da forma natural;
- Avaliar a fidelidade das formas geradas de maneira paramétrica;
- Sistematizar um fluxo de desenvolvimento para a modelagem tridimensional;
- Replicar as etapas de Engenharia Reversa e design paramétrico para outros elementos da natureza com padrão de crescimento espiral.

1.4 Estrutura da Dissertação

Este primeiro capítulo apresenta a introdução, a problematização da pesquisa e os objetivos do trabalho. O capítulo 2 apresenta a revisão bibliográfica sobre os temas abordados correspondentes aos assuntos sobre Métodos de Biônica, Crescimento Espiral e a Sequência de Fibonacci, Engenharia Reversa, Design Paramétrico e o Estado da Arte. O capítulo 3 apresenta os métodos utilizados neste trabalho, os quais focam no uso das tecnologias 3D para obtenção de geometrias através da digitalização tridimensional e de ferramentas de parametrização para aplicação da Engenharia Reversa. O capítulo 4 apresenta os resultados obtidos, bem como as discussões. O capítulo 5 apresenta as conclusões e as sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Nesse capítulo são abordados os assuntos sobre Métodos de Biônica, Crescimento Espiral e a Sequência de Fibonacci, Engenharia Reversa, Design Paramétrico e Estado da Arte associados para representação dos elementos construtivos como ferramentas de modelagem criativa e funcional para o desenvolvimento de produtos baseados na Biônica.

2.1 Métodos de Biônica

A biônica, segundo Wolf (2005), estuda os modelos naturais imitando ou buscando inspiração para o processo de design visando solucionar problemas humanos. É um modo de visualizar, aprender e valorizar a natureza, buscando referencias geométricas e conceitos para serem aplicados. A “biomimética se apodera da sabedoria da natureza acumulada em 3,8 bilhões de anos evolutivos para determinar o que funciona, o que é apropriado e o que permanecerá” (MEZINI, 2012).

No campo do design a Biônica pode ser definida como a identificação nos processos biológicos de princípios de funcionamento, características estruturais, formais e funcionais de soluções que serão aplicadas, se devidamente adaptados, na solução de problemas de projeto por intermédio de objetos com formas ou funções análogas (BARBA NETO, 2013).

No conceito de Biônica é apresentado que é possível analisar e adaptar os sistemas naturais e reproduzir estes princípios. As quais permitem a criação de formas análogas aos sistemas naturais. Em Detanico et al. (2010) os autores complementam que “tanto na forma de analogia como através de seus padrões geométricos/matemáticos. É possível observar, por exemplo, constantes proporções matemáticas na constituição de seres humanos, animais e vegetais” consistindo

importantes referências para o processo criativo, baseados nos elementos da natureza contribuindo para o projeto de produto.

No trabalho realizado por Coelho & Versos (2010), os autores identificaram três métodos gerais para a geração do conceito de design a partir da biônica. Estes métodos abordam sobre análise do problema e procura da solução inspirada na forma do sistema natural ou ainda na procura de uma analogia entre a solução natural e o projeto desenvolvido.

No primeiro método proposto os autores apresentam o conceito da “Bio-mimetismo” e os processos envolvidos para seleção e análise das amostras, listando os princípios do trabalho com o sistema natural. No entanto, este método não inclui normas para a transferência das características encontradas nas amostras naturais para o design. O método é distribuído em fases, partindo da identificação de necessidades, escolha e amostragem necessária, observação e análise dos componentes da estrutura morfológica, funções e processos. Após a aquisição da informação e da análise funcional, o designer identificará a viabilidade da aplicação de forma análoga entre a amostra estudada e o produto de design. O resultado será o projeto implementado, considerado pela aplicação das características da amostra aplicada e atendendo os requisitos de o produto proposto (COELHO & VERSOS, 2010).

O segundo método de concepção apresentado, o “projeto espiral”, dá ênfase ao ciclo de vida do produto, dando atenção a questões como processos de fabricação, embalagem e reciclagem do produto. Neste método, interações estão implícitas e a avaliação do resultado de cada etapa é também recomendada. O método é distribuído nas etapas de identificar, traduzir as funções do projeto em tarefas associadas na natureza, selecionar os melhores modelos naturais, selecionar a estratégia mais relevante, simular, avaliar e identificar a melhor alternativa (COELHO & VERSOS, 2010).

Finalmente, para o terceiro método apresentado no design de “Bio-inspiração” o processo de definição do problema está na procura de soluções biológicas e é apoiado por um conjunto de técnicas que promovam a descoberta. O método é dividido nas etapas de seleção e definição do problema, questionar com a pergunta: “Como as soluções biológicas executam esta função?”, encontrar a melhor

solução biológica, identificar a estrutura e o mecanismo, extrair os princípios relevantes para solução, tradução da solução biônica aplicando para o projeto (COELHO & VERSOS, 2010).

Na proposta de metodologia para o desenvolvimento de produtos baseados no estudo da Biônica, definida por Kindlein Júnior et al. (2002), os autores apresentam que a biônica consiste numa ferramenta alternativa para buscar a diferenciação de produtos em meio às exigências de competitividade mercadológica, “pois é uma ciência multidisciplinar que pesquisa nos sistemas naturais princípios e/ou propriedades (estruturas, processos, funções, organizações e relações) e seus mecanismos com o objetivo de aplicá-los na criação de novos produtos ou para solucionar problemas técnicos existentes na projetação”. Para Allgayer (2009) quando se quer aplicar a biônica no design esta transposição “depende essencialmente da criação de uma metodologia capaz de orientar e capacitar o processo da pesquisa”. Assim, como definido por Kindlein Júnior et al. (2002), o desenvolvimento de uma metodologia deve ser disposta em etapas, com a descrição das técnicas e procedimentos realizados que permitam uma compreensão objetiva, organizada e lógica dos conhecimentos, proporcionando aos designers e profissionais da área a utilização da Biônica no projeto de novos produtos. A metodologia é definida pelas etapas iniciais para o processo de seleção, coleta, observação e parametrização de amostras que desencadeiam as etapas finais, constituídas na analogia do sistema natural ao produto e à final aplicação projetual.

> “A concepção do produto quando realizada através da Metodologia aplicada à pesquisa da Biônica, torna-se prática e eficaz, devido a confiabilidade dos resultados analisados pelo processo evolutivo natural. Estes resultados podem ser verificados através dos exemplos de estudo de casos. Um dos produtos oriundos da Biônica, e que esta no mercado é o *Velcro*®”...”mecanismo de fixação proposto por George Mestrel (inventor do Velcro)”...”o exemplo do “*Honey Comb*”, produto baseado nas observações dos alvéolos de abelhas, cuja disposição - forma geométrica hexagonal - e simetria proporcionam resistência e leveza” (KINDLEIN JÚNIOR *et al.* 2002).

No método de concepção proposto por Kindlein Júnior et al. (2002), utilizado no Laboratório de Design e Seleção de Materiais (LdSM/UFRGS), a etapa inicial é definida pela seleção e identificação da amostra para à pesquisa que surge representada pela “análise do ambiente/produto; isto é; da identificação de uma necessidade não atendida satisfatoriamente” detectando o problema ou inovação desejada ao produto, “o que permite a preparação de um problema concreto. Outra forma é de simplesmente buscar características apreciativa utilizando-se da analogia estrutural, funcional e formal para futura aplicação em um projeto”. Na etapa de observação da amostra “é interessante salientar que o sistema biológico que está sendo observado deve ser considerado como um protótipo a ser investigado” mantendo o foco nos componentes físicos do elemento de pesquisa, estrutura, morfologia, funções, processos, distribuição no tempo, distribuição espacial, relações com o meio ambiente e classificação.

A metodologia em questão recomenda para a etapa de observação e mapeamento da amostra a utilização de algumas ferramentas bidimensionais (olho nú, foto macro, lupa ótica e Microscopia Eletrônica de Varredura - MEV). Da mesma forma, para a etapa de parametrização das informações utilizam-se softwares vetoriais (Corel Draw, Adobe Illustrator). Estudos mais recentes realizados por Cargnin et al. (2010) apresentaram a possibilidade da aplicação da digitalização 3D, neste caso para o estudo da geometria das paredes internas em fósseis das conchas dos moluscos cefalópodes. Os autores desenvolveram padrões geométricos a partir das suturas da concha e Realizaram a parametrização bidimensional da geometria digitalizada, criando novas formas para os núcleos geométricos que formam os elementos de reforço estrutural em painéis tipo sanduíche. Estes reforços estruturais, após serem prototipados, foram submetidos ao ensaio de compressão gerando as análises comparativas de resistência destas novas formas. A proposta desenvolvida proporcionou a inserção da digitalização 3D como ferramenta para avaliação, observação e estudo do modelo do sistema natural. Essa solução associada ao método proposto por Kindlein Júnior et al. (2002), proporcionou conhecer princípios, propriedades e mecanismos da natureza a fim de desenvolver novos produtos ou até mesmo servir de fonte de criatividade para o design de produtos.

2.2 Crescimento Espiral e a Sequência de Fibonacci

As espirais sempre foram muito populares, aplicadas nos ornamentos e nas peças de arte, razão da identificação da geometria na natureza, relacionada com o movimento, apresentadas em inúmeras peças exemplo das espirais de Arquimedes aplicadas na arte da Grécia antiga em colunas Jônicas (HELBIG, 2001).

Para Cruz (2012), das diversas formas encontradas na Natureza a espiral é a forma geométrica mais clássica, sua particularidade reside no fator movimento de rotação e translação. Estas espirais podem ser divididas em espirais constantes e as algorítmicas, também conhecidas como espiral logarítmica e espiral de Arquimedes. As espirais constantes podem ser encontradas nas teias de aranha e as algorítmicas nos Nautillus, caracóis e nuvens de tempestade. Enquanto que as espirais constantes tem um crescimento constante, as logarítmicas crescem segundo um somatório das partes, sendo uma parte delas de acordo com a sequência de Fibonacci (CRUZ, 2012).

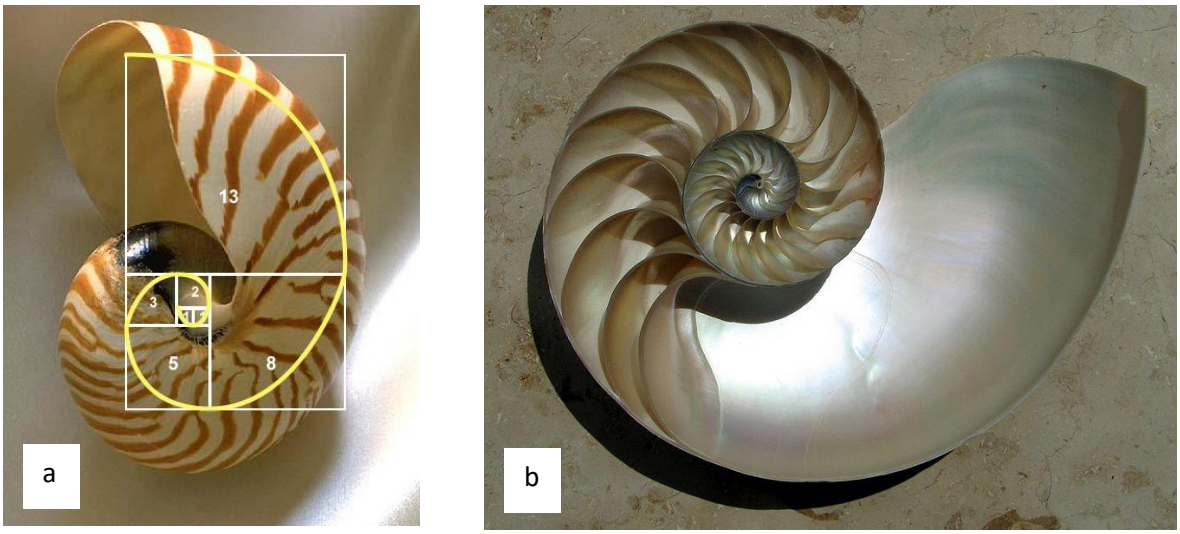
Pode ser observada a similaridade no padrão de crescimento da espiral logarítmica na geometria das sementes de uma pinha, nas sementes do girassol e no padrão de crescimento do abacaxi. Quando analisadas as sementes que crescem na pinha, observa-se duas espirais que se encontram e movem-se em direções opostas. Cada sequência de sementes pertence a ambos os pares de espirais: 8 delas movem-se na direção horária dos ponteiros do relógio e 13 na direção contrária, numa razão muito próxima da áurea descrita por Fibonacci. No caso do girassol, há 21 espirais num sentido e 34 no sentido oposto, também em proporções próximas à áurea (MARTINS, 2011).

A sequência numérica de Fibonacci foi descrita, em 1202, pelo matemático italiano Leonardo de Pisa, para ilustrar numericamente o crescimento de uma população de coelhos. Em virtude da constatação de que a reprodução destes revelava uma sequência gradual, na qual os números subsequentes representavam sempre a soma dos dois números imediatamente anteriores (0,1,1,2,3,5,8,13,21,...), esta sequência numérica trouxe uma série de desdobramentos para as artes, música e construções. A relação proporcional resultante da divisão de cada número pelo seu antecessor tende, gradualmente, a uma razão numérica conhecida como

proporção áurea, representada pela constante irracional Phi (1,618...), que é capaz de descrever matematicamente uma série de componentes da natureza, como, por exemplo, a concha Nautillus (ALLGAYER, 2009).

A Figura 1 mostra o corte total de uma casca da concha Nautillus, as câmaras internas dispostas em forma de espiral logarítmica. O organismo possui simetria translacional da qual a identidade ou módulo padrão repete-se ao longo de uma curva espiral logarítmica, apresentando ainda variações proporcionais de escala vinculadas à sequência de Fibonacci (TINELLO, 2016).

Figura 1. Concha Nautillus.



A figura da Concha *Nautillus* apresenta: a) A espiral de Fibonacci sobre o *Nautillus shell* (*Nautillus Macromphalus*); B) Seção transversal do escudo do *Nautillus*, estrutura da câmara e do septo.
Fonte: Tinello, 2016.

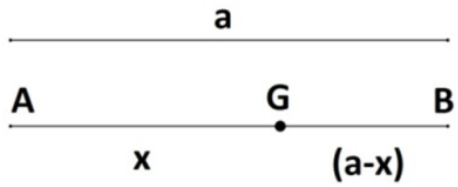
Segundo Cruz (2012), a representação da espiral logarítmica pode ser realizada graficamente pela construção do retângulo dourado, ou seja, que apresenta a proporção de crescimento na proporção do número de ouro (número áureo), cujo valor é 1,618, apresentando que:

> "A sequência de Fibonacci também está, de certa forma, presente no sistema modelar de Le Corbusier (Le Modulor). O nome surge da palavra módulo e ouro. Este cresce em conformidade com as leis do retângulo dourado, o qual pode ser interpretado como uma representação gráfica da sequência de Fibonacci ou também

> conhecida como proporção áurea. O número φ ou número de ouro, cujo valor é 1,618, é a proporção próxima do retângulo dourado, que consiste na divisão de um quadrado em dois retângulos de partes iguais. O uso destas proporções foi adaptado ao longo da História da Arte, nas várias expressões artísticas como, por exemplo, nas pinturas de Botticelli, e também na Mona Lisa ou no Homem Vitruviano de Leonardo da Vinci" (CRUZ, 2012).

Conforme Martins (2011), a sequência de Fibonacci pode ser representada de maneira gráfica, demonstrada na Figura 2 por uma linha reta. Considera que um segmento de linha reta de comprimento "AB" onde esta linha é dividida em um determinado ponto, na posição em "G". Apresentado na equação 1, de forma que o todo "AB" dividido pela maior parte da divisão "AG" seja igual a maior parte "AG" dividida pela menor parte "GB", igualdade que corresponde ao valor de Phi (número áureo).

Figura 2. Segmento de Reta "AB".



Divisão do segmento de reta para definição da equação.

$$\frac{AB}{AG} = \frac{AG}{GB} \quad (1)$$

Para chegar à proporção áurea, realiza-se o cálculo assumindo que AB = a, AG = x, GB = a-x, assim chega-se na equação 2.

$$\frac{a}{x} = \frac{x}{a-x} \quad (2)$$

Resolvendo a igualdade, resulta uma equação de segundo grau conforme a equação 3:

$$x^2 + ax - a^2 = 0 \quad (3)$$

Resolvendo a equação por Bhaskara, para obtenção da relação entre “x” e “a”, obtem-se a igualdade definida na equação 4:

$$x = \frac{-a + \sqrt{5a^2}}{2} = \frac{(-1 + \sqrt{5})\, a}{2} = 0{,}618033\, a \quad (4)$$

O resultado para a relação “x / a” é igual ϕ = 0,61803. Ao calcular o inverso deste valor, nota-se que o resultado é igual a soma de 1 ao valor de ϕ, resultando desta forma o número áureo φ =1,618033, definido na equação 5:

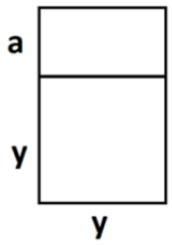
$$\frac{1}{\phi} = 1 + \phi = 1{,}618033\ldots = \varphi \quad (5)$$

Na sequência de Fibonacci (0,1,1,2,3,5,8,13,21,...), conforme se aumenta o número (n), a relação de cada dois números adjacentes é progressivamente mais próxima da razão áurea de 1:1,618, aplicando a equação matemática definida na equação 6:

$$F_{n+1} = F_n + F_{n-1} = F_n \times \varphi \quad (6)$$

Realizando o cálculo “F_n / F_{n-1}”, para um determinado valor de “n”, o resultado tende ao número áureo φ. Com a proporção áurea é possível construir o chamado retângulo áureo. Este retângulo, na Figura 3, possui seus lados nessa proporção, ou seja, conforme mostra a equação 7, o comprimento “y” somado a altura “a” dividido pela largura “y”, é igual a largura “y” dividido pela altura “a”.

Figura 3. Retângulo Áureo.

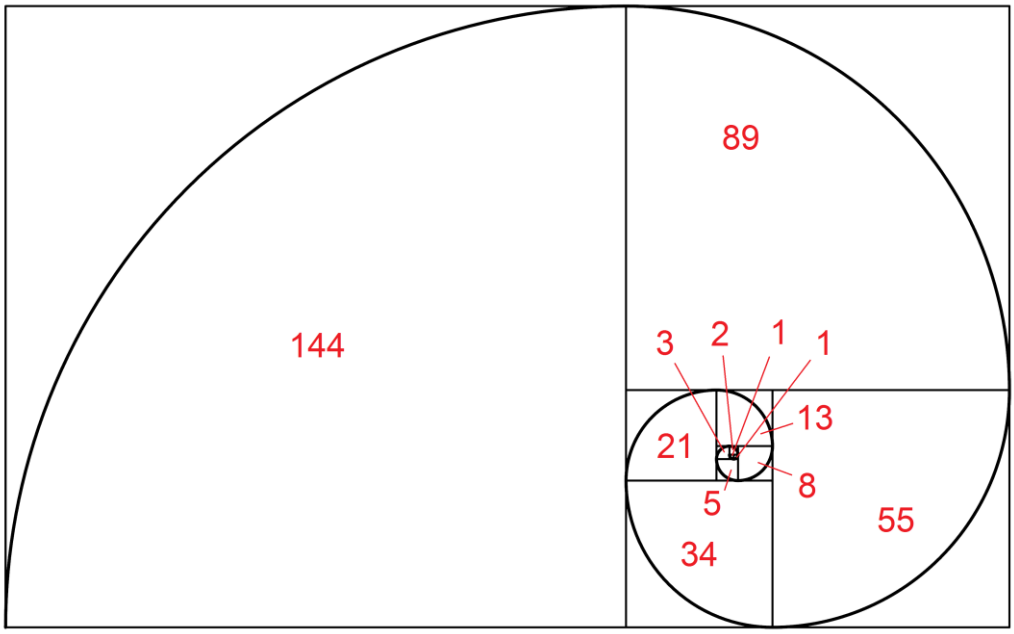


Divisão do retângulo Áureo e relação de igualdades para representação da equação.

$$\frac{y+a}{y} = \frac{y}{a} \quad (7)$$

Na Figura 4, apresenta-se a modelagem da divisão do retângulo áureo em um quadrado e um retângulo, o novo retângulo também é áureo, sucessivamente dividindo-o novamente em um quadrado e um novo retângulo. Para o desenho dos arcos é necessário unir os cantos por uma circunferência partindo do centro do canto oposto do quadrado gerado. Esse resultado é uma espiral áurea, conhecida por espiral dourada da série de Fibonacci (MARTINS, 2011).

Figura 4. Espiral Áurea, série de Fibonacci.



Fonte: Martins (2011).

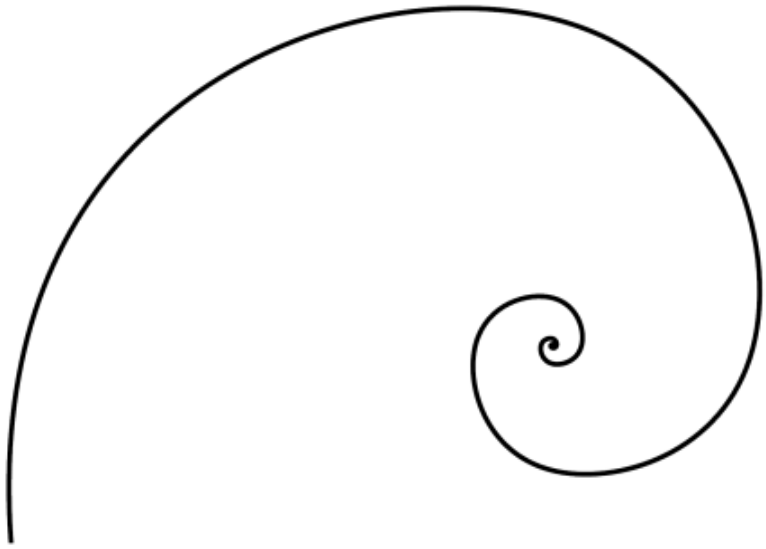
A espiral logarítmica, equiângular e de crescimento são similares às espirais curvas que muitas vezes aparecem na natureza. A espiral logarítmica foi primeiramente descrita por Descartes e mais tarde extensivamente investigada por Jacob Bernoulli, que a denominou de *Mirabilis Spira* (espiral maravilhosa) (HELBIG, 2001).

No caso da espiral logarítmica, o raio cresce exponencialmente com o ângulo polar e o seu inverso, dependendo apenas do raio inicial "r_0" escolhido. A equação do raio de curvatura da espiral logarítmica pode ser dada por coordenadas polares conforme mostra a equação 8, resultando o raio "r" do arco para representação geométrica da curva.

$$r = r_0 * e^{(a * phi)} \quad (8)$$

A partir da aplicação da equação 8 o resultado é uma espiral com distância de crescimento fixo, onde: "r" é o raio do arco; "r_0" é o raio de partida; "a" é o parâmetro de correção da curva e "Phi" é o número de ouro (φ). Por meio desta equação é possível desenhar uma espiral com ampliações constantes e de sentido anti-horário (a>0). Dessa forma, as espirais logarítmicas onsistem apenas de um ponto inicial, cujo sentido de rotação depende do sinal do parâmetro "*a*". O resultado da equação para a representação gráfica em *software* de CAD está definida pelos parâmetros apresentados na Figura 5 (HELBIG, 2001).

Figura 5. Espiral Logarítmica.



Resultado da Espiral Logarítmica
Fonte: Adaptado de Helbig (2001).

Segundo Helbig (2001), a espiral *Mirabilis Spira* é um caso especial de expressão logarítmica, que em geral é uma função com coordenadas polares, utilizando o elemento "t" para correção do arco, apresentado na equação 9:

$$r(t) = r_o * \ \mathrm{e}^{(\mathrm{t} * \mathrm{b})} \tag{9}$$

A constante "b" determinará o cálculo da ampliação da espiral em questão. A espiral dourada tem sua ampliação por quadrante de tal modo que para cada 1/4 de volta (*90° ou* $\pi/2$ *em radianos*) o valor da função logarítmica, para conversão do

número de ouro φ = *1,618033* para radianos, corresponde ao valor apresentado na equação 10.

$$cot\, b = \frac{ln\,\varphi}{\frac{\prod}{2}} = \frac{log_e 1{,}6180339\ldots}{\frac{\prod}{2}} = 0{,}306349\ldots \quad (10)$$

Na equação 11, substituindo o valor encontrado para b = 0,30635, obtem-se uma nova função, aplicada em *software* de programação e CAD, definida pela equação 9, tornando-se:

$$r(t) = r^0 * e^{(t * 0{,}30635)} \quad (11)$$

O modelo matemático é importante para a parametrização das formas que será realizada a partir da digitalização 3D de elementos da natureza. A interpretação da forma é o elemento principal para a reconstrução pela técnica da Engenharia Reversa que adota a curva como principal característica para o estudo do projeto.

2.3 Engenharia Reversa

Conforme Silva (2006) o método de digitalização tridimensional é utilizado “para captar imagens e dados em 3D e, com auxílio de ferramentas computacionais, permite obter com grande precisão detalhes de superfícies, texturas e mesmo de objetos inteiros”. Existem diversos princípios de digitalização 3D, os quais podem ser divididos em sistemas de digitalização baseados em contato e sistemas sem contato. Dentre os sistemas de digitalização sem contato estão as tecnologias descritas por Silva (2011), Digitalização a Laser por Holografia Conoscópica, Digitalização a Laser por Triangulação, Digitalização por Luz branca (Luz Estruturada) e Digitalização Baseada em Fotografia (fotogrametria).

Após a digitalização tridimensional os dados da superfície da peça são processados na forma de pontos distribuídos no espaço com coordenadas (x,y,z). A ação da varredura da superfície pelo sistema de digitalização vai retornar milhares de pontos chamados de nuvem de pontos, caracterizando a forma do objeto apresentando a geometria externa da peça digitalizada, esta informação servirá de

base para a manipulação por ferramentas de desenho em *software* de CAD que permitirá a geração de curvas, malhas, superfícies e sólidos tridimensionais compatíveis com sistemas CAD/CAE/CAM. A técnica de “obtenção de modelos virtuais a partir de modelos físicos ocorre em um processo inverso à engenharia convencional (onde a partir do virtual obtém-se o físico) e por isso esta técnica é conhecida como Engenharia Reversa” (SILVA, 2006).

Para Chikofsky apud Araujo (2010) o conceito da Engenharia Reversa “tem sua origem na análise de produtos - onde a prática de se decifrar os projetos através do produto finalizado é senso comum. A Engenharia Reversa é regularmente utilizada para se melhorar os próprios produtos, bem como analisar os produtos do concorrente”. Os autores em YE et al. (2008), apresentaram uma metodologia com uso de tecnologias 3D, onde as superfícies externas do produto foram digitalizadas por um scanner 3D e manipuladas em *software* de CAD, inicialmente identificando as características do objeto digitalizado e posteriormente partindo para uma nova definição de produto mais atualizado.

A técnica da Engenharia Reversa tem como processo de estudo dos princípios de geração do produto, objeto ou sistema através da análise de sua estrutura, função e operação. Isso envolve selecionar uma peça mecânica, um componente eletrônico ou um programa de computador, com a intenção de analisar seu funcionamento em detalhes. O resultado é referência para confecção de uma peça similar com função de substituir a original em casos de manutenção. Esta técnica poderá servir como inspiração para o conceito de um novo produto ou programa que tenha as mesmas funções sem ser apenas uma cópia do original. O propósito é o de deduzir as decisões de projeto a partir do produto final fazendo uma leitura apurada de geometria, material e comportamento do processo de fabricação (RAJA, 2007). Neste trabalho, pretende-se aplicar este conceito não em peças mecânicas propriamente ditas, mas em elementos naturais, seguindo a metodologia de Biônica.

Em Ouamer-Ali et al. (2014) os autores apresentam a metodologia para a Engenharia Reversa, por meio de quatro ações principais definidas inicialmente pela exploração do produto por intermédio da aquisição dos dados por digitalização; seguindo pela segmentação destes dados adquiridos; extração do conhecimento e

finalizando com a parametrização do resultado digitalizado através da reconstrução do modelo 3D em *software* de CAD. Ye et al. (2008) complementam que a abrangência do domínio da metodologia de Engenharia Reversa está na dependência do conhecimento e prática definida pelo profissional que vai modelar o produto em um determinado *software* de CAD. Apesar desta abrangência, algumas etapas são comuns nos métodos os procedimentos que definem o fluxo de trabalho para a Engenharia Reversa estão definidos por cinco etapas:

> "1. Importação de dados digitalizados em forma de nuvem de pontos (por exemplo, IGES, ASCII, OBJ) ou malha (por exemplo, STL, WRL, 3DS, OBJ).
> 2. Pré-processar os dados importados, incluindo o registro.
> 3. Criar um modelo de malha da nuvem de pontos digitalizada.
> 4. Criar superfícies com base na malha.
> 5. Exportar as superfícies reconstruídas no sistema CAD 3D" (YE et al., 2008).

Após a importação e o pré-processamento dos pontos da nuvem, que incluem as tarefas de remoção do ruído, o registro, o alinhamento, fechamento dos furos, reparos e definição do tamanho da amostragem, resulta numa malha que servirá de base para reconstrução do objeto digitalizado (YE et al., 2008).

A malha tridimensional é formada por inúmeros triângulos orientados pela união dos pontos, três a três. Segundo Silva (2006), este "arquivo com dados de malhas de triângulos planos, os quais também contém informações sobre os vértices e as normais de cada triângulo, podem ser salvos no formato STL (de *STereoLithography*)". Sendo o formato padrão STL, o mais conhecido para o uso nos sistemas de Prototipagem Rápida e um dos formatos mais utilizados para a manipulação em *software* de CAD pela técnica de engenharia reversa.

Após a digitalização e manipulação da nuvem de pontos em malha triangular, a sequência para engenharia reversa está na modelagem tridimensional por curvas e superfícies. Galantucci (2008) apresenta uma técnica dividida em duas etapas: "a primeira etapa é uma individuação lógica da forma do objeto 3D digitalizada e o segundo passo é da conversão da informação adquirida de nível lógico para um modelo de gráfico". O autor apresenta que a malha triangular é o

elemento base de construção do modelo gráfico. Partindo da nuvem de pontos é possível detectar a forma lógica da superfície. O autor recomenda que para a reconstrução das superfícies e criação automática de malhas de qualidade é necessária a identificação das arestas vivas do objeto. É importante, ainda, uma densidade mínima de pontos para a definição das superfícies a serem modeladas. Todos estes são requisitos básicos para a aplicação da técnica de reconstrução e geração das superfícies que vão formar o novo produto (GALANTUCCI, 2008).

A reconstrução da superfície é a etapa central da modelagem por Engenharia Reversa, para tanto os autores Ye et al. (2008) propõem três estratégias. A primeira forma é a auto geração das superfícies pela modelagem automática em formas livres; a segunda estratégia é utilizar o recurso da modelagem paramétrica baseada em sólidos; e a terceira é a modelagem das superfícies à base de Curvas. O procedimento para escolha de uma das três estratégias de modelação baseia-se na análise das características do modelo.

A estratégia da modelagem de superfície de forma livre automática é usada, principalmente, na modelagem de geometrias orgânicas, incluindo produtos de consumo, brinquedos e médicos. Neste processo, as superfícies resultantes se aproximam do modelo original, mas devido à falta de recurso no *software* de CAD 3D estas modelagens não apresentam o histórico de projeto, não possibilitando fazer modificações e correções dos modelos (YE et al., 2008).

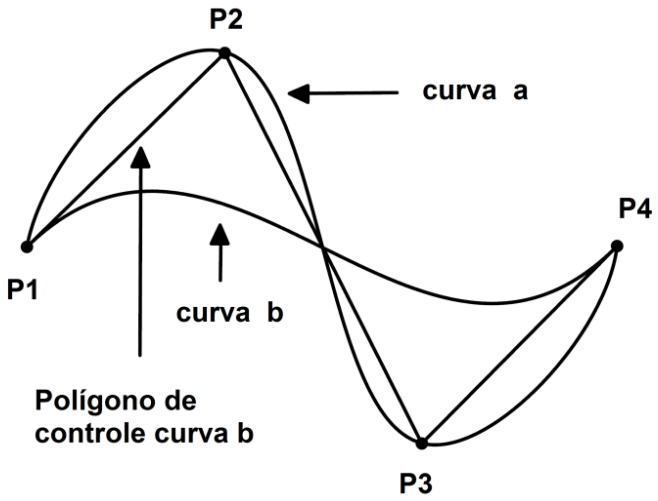
O recurso baseado em modelagem de sólidos paramétricos é amplamente utilizado em *softwares* de CAD 3D. Inicialmente devem ser definidas as regiões com malhas geométricas, com características de objetos cilíndricos, cúbicos, superfícies planas, regiões simétricas, bordas e raios. Os parâmetros de definição das superfícies são extraídos da divisão da malha, aplicadas em superfícies planas que após serem estendidas, aparadas e costuradas formam o modelo no ambiente CAD. A partir deste objeto sólido, ações como parametrização, filetagem, definição de espessura de parede e separação das faces são de fácil composição para o projeto do produto (YE et al., 2008).

No caso da modelagem das superfícies baseadas em curvas, este método visa atender os requisitos de projeto quando se quer maior controle da geometria e é aplicada em superfícies com formato misto (orgânica e geométrica). As curvas

geradas a partir da malha podem ser curvas de seção, curvas de contorno, curvas de silhueta e as linhas de recurso. Elas podem ser extraídas automaticamente ou com ferramentas de esboço. As superfícies geradas a partir das curvas dependem muito da qualidade entre o desenho e a matemática destas curvas. A experiência em CAD 3D do designer irá afetar significativamente no resultado do modelo, na qualidade e na precisão das superfícies reconstruídas (YE et al., 2008).

Conforme apresentado por De Aviz (2010), o equacionamento matemático capaz de representar uma curva em um sistema computacional é denominado por uma função spline. Uma curva spline é definida matematicamente por dois ou mais pontos de controle. Os pontos de controle da curva são chamados de nós. Há diversos modelos matemáticos aplicados para este fim, um dos mais utilizados é a curva de Bézier que é uma curva polinomial paramétrica que é expressa pela interpolação linear entre os pontos de controle. A partir dos pontos selecionados, dois métodos matemáticos podem ser empregados para a construção da curva, conforme apresentado na Figura 6: o método de interpolação (consiste em criar uma curva que passa pelos pontos fornecidos); e o método de aproximação (cria uma curva através de um polígono de controle).

Figura 6. Método de Interpolação e Aproximação.



A curva “a” é a representação gráfica da curva Bézier utilizando o método de interpolação caracterizada pela curva passando pelos pontos fornecidos; a curva “b” é a representação gráfica da curva Bézier utilizando o método de aproximação com as linhas que formam o polígono de controle da curva.

Fonte: DE AVIZ (2010).

Para avaliar a qualidade de uma curva ou de uma superfície é verificado o nível da sua continuidade, propriedade que está relacionada com o grau de suavização da curva ao longo da sua extensão. “Uma geometria descontínua pode representar implicações estéticas ou funcionais em um produto” (DE AVIZ, 2010).

Conforme as restrições das continuidades forem definidas nos *softwares* de CAD 3D, as ligações dos segmentos das curvas apresentarão variações de posição, de tangência ou de curvatura. A continuidade de posição (equação matemática com grau 0) ocorre quando dois segmentos possuem em comum apenas o ponto de contato. A continuidade de posição e tangência (equação matemática com grau 1) ocorre quando os segmentos possuem em comum vetores de tangência, além do contato. A continuidade de posição, tangência e curvatura (equação matemática com grau 2) ocorre quando os segmentos possuem em comum, no ponto de contato, o mesmo vetor de tangência e o mesmo grau de curvatura (DE AVIZ, 2010). Os sistemas CAD com recursos computacionais mais avançados para modelamento de superfícies, segundo De Aviz (2010), disponibilizam ferramentas gráficas para auxiliar na identificação do grau de continuidade de regiões de uma curva, união entre curvas e superfícies. Essa ferramenta é denominada pente de curvatura (curve comb) e é muito utilizada no processo de Engenharia Reversa.

No presente trabalho foram utilizados dados de digitalização 3D como base para a geração do design de superfícies em CAD. Extraindo por meio de ferramentas computacionais os parâmetros das malhas obtidas, para gerar curvas e superfícies paramétricas.

2.4 Design Paramétrico

O Design Paramétrico parte da base do modelo com parâmetros de controle das geometrias, gerando diversas possibilidades de projeto por meio de um conjunto de regras e algoritmos planejados. O modelo paramétrico é um processo de modelação, que pode ser simulado no computador, possibilitando a modelagem semelhante ao desenvolvimento morfogenético estudado na Biologia (COSTA, 2008).

O Design Paramétrico é compreendido por um método computacional que se destina à projetação de um sistema de funções definidas que vão desenhar o objeto, permitindo gerar uma série de variações geométricas. Este processo permite trabalhar a genética de um objeto ou sistema ao invés de modular apenas a sua forma final (ESTEVEZ, 2003).

Por intermédio da interação do *software Rhinoceros* com o *Grasshopper* é possível aplicar a metodologia projetual descrita por Santos (2014), aplicando o método de seleção natural ao produto a partir da “criação do código genético do produto”, e a possibilidade de alteração conforme necessidade do projetista. Este método permite análise de várias formas e compreender quais as que melhor respondem à função do objeto estudado. A interação com o *software Grasshopper*, possibilita a modelação paramétrica, isto é, o seu funcionamento é realizado por parâmetros, que podem ser modelados e alterados durante o processo de criação, permitindo obter infindáveis soluções ao projeto (SANTOS, 2014).

Visando facilitar o processo de modelagem, controle dos elementos e parametrização das geometrias, a solução apresentada é do uso da aplicação do *Grasshopper* que é executado dentro do *software* de CAD *Rhinoceros* 3D. O aplicativo *Grasshopper* é uma linguagem de programação visual desenvolvida por David Rutten e Robert McNeel & Associates. A interface principal para o projeto de algoritmo em *Grasshopper* é o editor baseado em nó. Os dados são passados de componente a componente ao longo do conector por fio que liga um ponto de saída com um ponto de entrada. Os dados podem ser definidos localmente como constantes, ou podem ser importados a partir de um documento do *Rhinoceros* ou um arquivo no computador. Os dados são sempre armazenados em parâmetros, que pode ser de livre flutuação ou ligados a um componente de entrada e saídas de objetos. Desta forma o *Grasshopper* permite aos usuários combinar a programação visual e textual (*C#, VB.NET, Python*) dentro do mesmo ambiente. O *Grasshopper* permite a criação de algoritmos paramétricos, que são transformados em geometrias tridimensionais (SANTOS, 2014). O *Grasshopper* apresenta uma interface gráfica bastante avançada com uma série de características que são raramente encontrados em *software* de desenho CAD (KHABAZI, 2012).

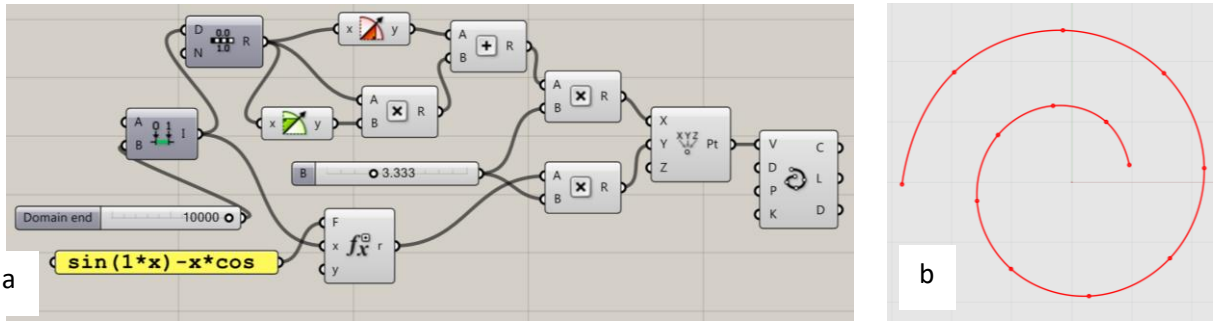
Buscando mapear a produção acadêmica realizada na área, bem como observar aspectos que vem sendo destacados e métodos empregados, a seguir apresenta-se um tópico sobre o estado da arte relacionado ao presente trabalho.

2.5 Estado da Arte

Yiwei & Xia (2010), apresentam o resultado do projeto digital realizado com o auxílio dos *softwares* de modelagem no *Grasshopper* e *Rhinoceros* para o desenho arquitetônico paramétrico baseado na espiral áurea da série de Fibonacci. O projeto arquitetônico desenvolvido por Yiwei & Xia (2010) apresenta o método paramétrico utilizado na programação para a construção do edifício realizado. A variação arquitetônica geométrica e dimensional foi integrada ao gerenciamento digital pelo *software* relacionado com as condições necessárias para o projeto arquitetônico. O método utilizado possibilitou alternância do projeto de maneira paramétrica visando atender as atribuições arquitetônicas relacionadas com a forma digital possibilitando o controle flexível das alterações de projeto.

Na Figura 7 está representada a espiral apresentada na forma gráfica e paramétrica desenvolvida no *Rhinoceros* e *Grasshopper*.

Figura 7. Programação da Espiral Desenvolvida por Yiwei & Xia.



Na figura: a) funções da programação no *Grasshopper*; b) resultado gráfico da Espiral.

Fonte: Yiwei & Xia, (2010).

Um projeto inovador de destaque é a torre Warka Water que foi apresentada na Bienal de Arquitetura de Veneza, em 2012 (Figura 8). O projeto desenvolvido pelo arquiteto italiano Arturo Vittori capta a água da atmosfera e disponibiliza para o consumo. A idéia do Warka surgiu da necessidade de ajudar a comunidade da Etiópia que sofre com a falta de água. O projeto consiste em uma torre construída em forma de cesta, com 8,5 metros de altura, com formato de estrutura triangular estável e base modular, semelhante à estrutura de um abacaxi. O projeto é feito em bambu ou talos de juncos (espécie de gramínea) e tem o interior forrado com uma malha plástica, similar aos sacos usados no transporte de frutas e legumes. A umidade presente na atmosfera condensa na tela de polietileno de alta densidade, para depois ser acumulada em água para o consumo. O processo acontece por condensação da água do orvalho, como consequência da diferença de temperaturas entre a alta temperatura do dia para a queda de temperatura da noite. “A estimativa é de que a Warka Water seja capaz de coletar 100 litros de água por dia quando instalado em um clima como o da Etiópia” (VITTORI, 2012).

Figura 8. *Warka Water* Mostra Internazionale di Architettura.

Protótipo do *Warka Water* na Biennale Di Architettura 2012 di Venezia.

Fonte: Vittori (2012).

Allgayer (2009) destaca o projeto do edifício sede da *Swiss Re*, Figura 9, em Londres, projeto do escritório de arquitetura *Foster & Partners*. A forma orgânica estruturada em diagonais foi uma alternativa para melhor suportar as cargas incidentes por ventos da região, compartilhando as características orgânicas aos requisitos tecnológicos. Os vidros de vedação são planos e circundam o edifício de maneira crescente semelhante as microscópicas cadeias de DNA. Segundo Caneparo (2014) o projeto apresenta elementos construtivos semelhantes a geometrias orgânicas, desde a forma externa definida por uma ogiva até a distribuição estrutural do processo de criação do edifício. Os andares foram projetados a partir da vinculação em torno de um cubo e na sequência estes andares foram rotacionados a cada 5 graus, criando assim uma distribuição dos andares em forma de espiral. Essa configuração possibilitou o giro do poço de luz facilitando a entrada da iluminação natural e melhorando a renovação do ar com movimento do fluxo no interior, fatores que proporcionaram conforto para os freqüentadores do edifício.

Figura 9. Projeto do Edifício *Swiss Re*.

Projeto *Swiss Re HQ, 30 St Mary Axe, London, UK,* 1997-2004.
Fonte: Escritório dos Arquitetos *Foster and Partners*. Disponível em: <http://www.fosterandpartners.com>.

Browne (2008) destaca a obra do arquiteto espanhol Antoni Gaudí (1852-1926), que trabalhou principalmente em Barcelona, famoso por seu estilo de arte original e formato orgânico. Em busca da verdade absoluta e da beleza autêntica, Gaudí encontrou resposta na forma das flores, na geometria das ondas e das montanhas, na curva imperfeita das plantas e na estrutura das árvores, com seus troncos, galhos, frutos e animais que habitavam o mesmo espaço. Gaudí se inspirou em curvas naturais, formas e padrões de crescimento, e incorporou esses princípios em seus projetos usando um processo conhecido como a construção orgânica em que uma idéia estrutural é transformada por elementos construtivos que crescem numa espiral. Segundo Rian & Sassone (2014) e Ramzy (2015), a obra mais conhecida de Gaudí, referência da moderna arquitetura catalã, é o Templo Expiatório da Sagrada Família, uma grande obra arquitetônica da cidade de Barcelona. A construção foi projetada para ter três grandes fachadas: a Fachada da Natividade, a Fachada da Paixão e a Fachada da Glória. Quando finalizado, o templo terá 18 torres distribuídas nas entradas portais. O interior formado por colunas arborescentes inclinadas e abóbadas baseadas em hiperbolóides e parabolóides. Outro projeto da arquitetura de Gaudí é o *Güell Palace*, o edifício se destaca por sua concepção inovadora de espaço e luz. A Figura 10 apresenta o projeto arquitetônico no estilo mosaico de uma das vinte chaminés do telhado do *Güell Palace*.

Figura 10. Chaminé do *Güell Palace*.

Fonte: Browne (2008).

Segundo Ekins (2016), é possível utilizar o design paramétrico no software Autodesk Inventor por meio do uso da linguagem de programação em script, por ferramentas de programação orientada a objetos com o uso do Visual Basic Script (VBScript). De modo flexível, semelhante às rotinas de uma Macro, o design paramétrico é editado no programa do VBScript, este arquivo é arrastado e solto na barra de ferramentas criando assim a ferramenta de desenho, em seguida esta ferramenta estará disponível para criar o desenho segundo o script.

Diversas pesquisas apresentam a utilização da programação orientada a objetos associadas ao design paramétrico com script de programação, recursos que agregam desempenho aos projetos em CAD e CAM. Uma das primeiras pesquisas nesta área foi realizada por Adamczyk e Kociolek (2001) quando da criação de banco de dados tecnológicos para a programação em máquinas ferramentas. Biard (2010) apresentou equações para criação de superfícies orientadas por vetores normais às linhas de construção. Os autores Sendra e Sevilha (2013) apresentaram algorítmos para parametrização de superfícies radicais de alta complexidade. Adsul et al. (2014) apresentaram a estrutura matemática para o cálculo de curvas de contato entre faces, limitando as fronteiras para aparar curvas de forma paramétrica. Kassabov (2014) apresenta o teorema que permite equacionar os parâmetros para construção de superfícies canônicas polinomiais de forma paramétrica. Maier (2014) aborda o equacionamento e a tolerância para construção de curvas com número mínimo de segmentos aplicado em arcos, círculos e linhas. Alcázar et al. (2015) apresentam o método para identificar o plano de simetria de uma curva racional. Beccari e Neamtu (2016) abordaram o equacionamento de como construir splines paramétricas associadas a geometrias homogêneas e parametrização na forma de superfícies livres. Bo et al. (2016) apresentaram o método para aproximar e suavizar localmente as superfícies estriadas. Machchhar e Elber (2016) apresentaram algoritmo para calcular a interpolação dos pontos de controle para construção de curvas de Bézier. Lee et al. (2016) utilizaram algoritmos para calcular diagramas Voronoi apartir de curvas. Nguyen e Peters (2016) desenvolveram método para suavização da superfície isogeométrica.

O presente trabalho insere-se neste contexto, com a contribuição científica da geração de algoritmos para modelagem de superfícies da natureza com padrão de crescimento expiral.

3. MATERIAIS E MÉTODOS

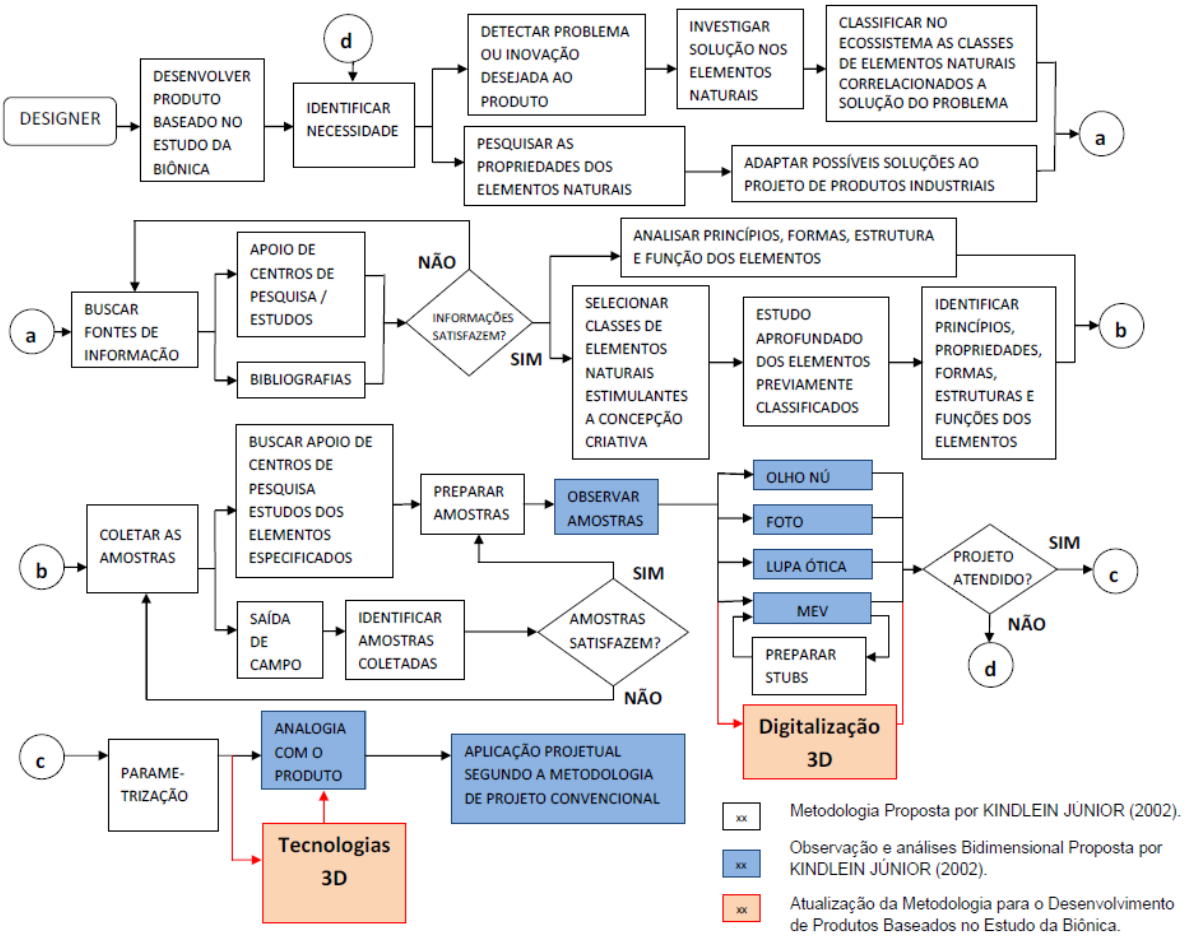
A primeira etapa do estudo consistiu em uma revisão bibliográfica para compreensão da metodologia de desenvolvimento de produtos baseados na Biônica; do uso das tecnologias 3D para obtenção de geometrias por meio da digitalização tridimensional; e de ferramentas de parametrização para aplicação da Engenharia Reversa. Para atingir o objetivo proposto, constatou-se a necessidade de uma metodologia de modelagem em *software* de CAD 3D paramétrico para gerar uma curva espiral logarítmica, associada à sequência de Fibonacci, e a partir desta curva modelar de forma automática as superfícies do objeto digitalizado. Neste sentido, este trabalho é apoiado no método apresentado por Kindlein Júnior et al. (2002), com o acréscimo das tecnologias de digitalização tridimensional e de processamento de nuvem de pontos utilizada por Silva (2006), complementado pela parametrização de superfícies à base de curvas descrita em Ye et al. (2008).

Na Figura 11, está representada a proposta de atualização da Metodologia para o Desenvolvimento de Produtos Baseados no Estudo da Biônica apresentada por Kindlein Júnior et al. (2002). Propõe-se adicionar a etapa “Digitalização 3D” (módulos em destaque) no item “observar amostras”, este método de observação e de mapeamento tridimensional foi utilizado por Cargnin et al. (2010). Também propõe-se adicionar uma nova modalidade para tarefa de “parametrização” auxiliada por recursos das “Tecnologias 3D”, tais como a Engenharia Reversa e o Design Paramétrico.

Para aplicação da metodologia proposta, foi realizado um estudo piloto com uma forma da natureza representativa do padrão de crescimento espiral. Foi escolhido como modelo de estudo o abacaxi do tipo pérola, pertence à família das bromeliáceas, o qual é oriundo da América do Sul e cultivado em qualquer região quente do mundo. No Brasil, são cultivadas várias espécies, como o abacaxi amarelo, porém, a que se destaca é a variedade Pérola, de polpa amarelo-pálida, sabor bastante doce, casca esverdeada, mesmo quando maduro e pouca acidez (LOPES, 2014). Comercializado como fruta, o abacaxi tem normalmente geometria cilíndrica ou ligeiramente cônica, constituído na parte interna pela infrutescência

(comestível) envolvida por uma casca com pequenos frutilhos ou frutículos (ANDRADE, 2015).

Figura 11. Proposta de atualização da Metodologia para o Desenvolvimento de Produtos Baseados no Estudo da Biônica.

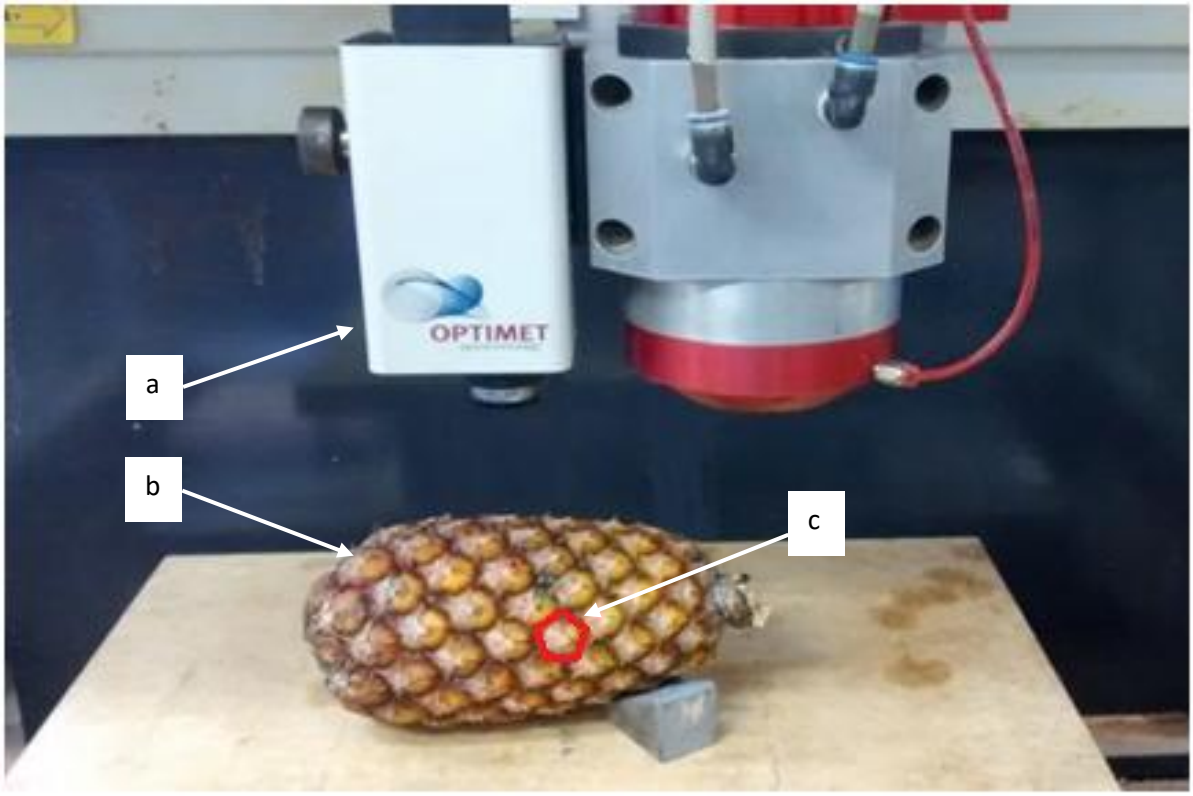


Fonte: Adaptado de Kindlein Júnior *et al. (*2002).

A etapa de digitalização 3D foi baseada no método utilizado por Silva (2006) e Cargnin et al. (2010). Assim, para esta fase foi utilizado um *scanner* tridimensional a Laser de marca Tecnodrill, modelo Digimill 3D, instalado nas dependências do Laboratório de Design e Seleção de Materiais (LdSM) da Universidade Federal do Rio Grande do Sul (UFRGS). O Digimill 3D é um equipamento híbrido fresadora CNC (controlado numericamente por computador) e *scanner* tridimensional a Laser, ou seja possui dois cabeçotes, um para a usinagem e outro para a digitalização. O funcionamento do *scanner* é determinado pelo movimento do cabeçote de digitalização sobre determinada peça no plano dos eixos X e Y, enquanto o Laser vai medindo a altura no eixo Z. Como, “resultado da varredura são obtidos arquivos

de texto com os pontos da superfície do objeto descrito em coordenadas (x,y,z)" formando uma nuvem de pontos com a forma da superfície digitalizada (SILVA, 2006). Na Figura 12 é possível visualizar o posicionamento entre o abacaxi e o cabeçote de digitalização.

Figura 12. Digitalização 3D com o Scanner Digimill 3D.



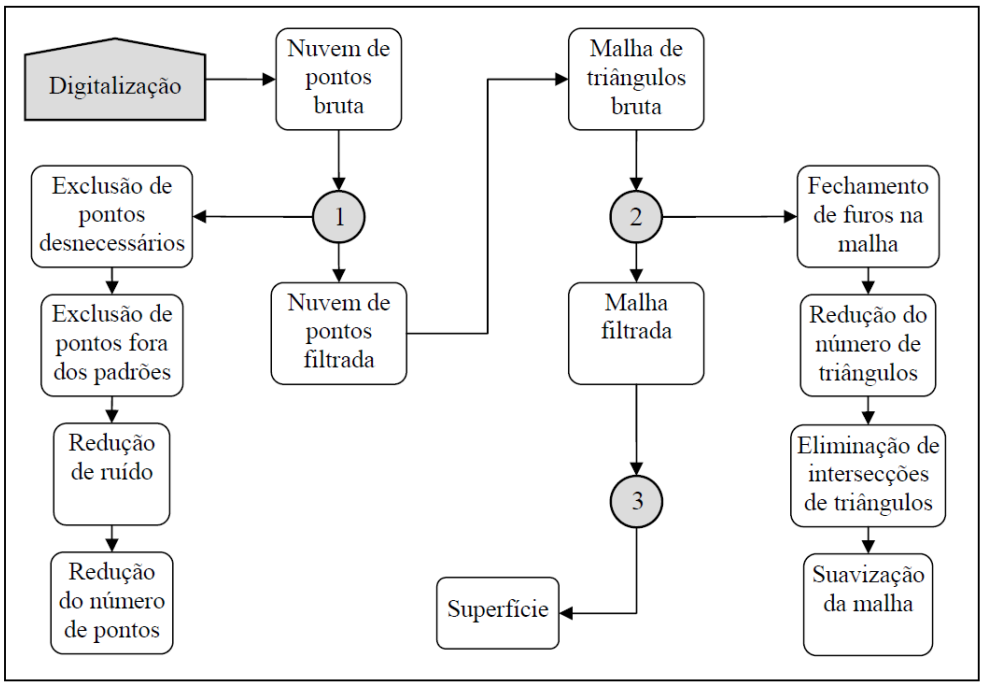
Sistema de Digitalização utilizado: a) cabeçote com sensor conoscópico a Laser 3D OPTIMET, instalado na máquina de usinagem com deslocamento controlado Digimill 3D; b) Abacaxi amostra para digitalização; c) Frutículo.
Fonte: LdSM/UFRGS.

O procedimento iniciou com a seleção da lente de 150 mm que possui precisão máxima de 35 µm. O objeto a ser digitalizado foi posicionado dentro da faixa de operação da lente que é de aproximadamente 70 mm. Foi utilizada uma resolução entre pontos e linhas de 0,2 mm. O processo de medição foi repetido girando o objeto, quatro vezes, visando varrer por inteiro o abacaxi, formando assim o volume do objeto. Os quatro arquivos gerados na digitalização tridimensional foram alinhados e posicionados no *software Geomagic Studio* de maneira que o

resultado disponibilizado apresente uma única malha com a geometria externa do abacaxi, salva no formato padrão STL.

Após a realização da digitalização, Silva (2006) recomenda o processamento da nuvem de pontos conforme apresentado na Figura 13. Dessa forma, a primeira etapa do processamento da nuvem de pontos foi a exclusão dos pontos desnecessários (mesa do *scanner*, dispositivos de apoio e fixação da peça). As etapas seguintes necessitam de *software* específico para filtragem dos pontos que estejam fora dos padrões, redução do ruído e redução do número de pontos. Em Ye et al. (2008), os autores recomendam reduzir o número total de pontos pelo uso da filtragem suavizando a nuvem e possibilitando construir uma malha de melhor qualidade que servirá posteriormente para a etapa de desenvolvimento de produto pelo método da Engenharia Reversa. Neste sentido, esses procedimentos de tratamento das nuvens de pontos e posterior geração de malha foram realizados no *software Geomagic Studio.*

Figura 13. Processamento da Nuvem de Pontos Digitalizada.



Fonte: Silva (2006)

A malha de triângulos é o resultado da digitalização, sendo que uma malha completa e com pontos suficientes otimizará a posterior etapa de desenvolvimento

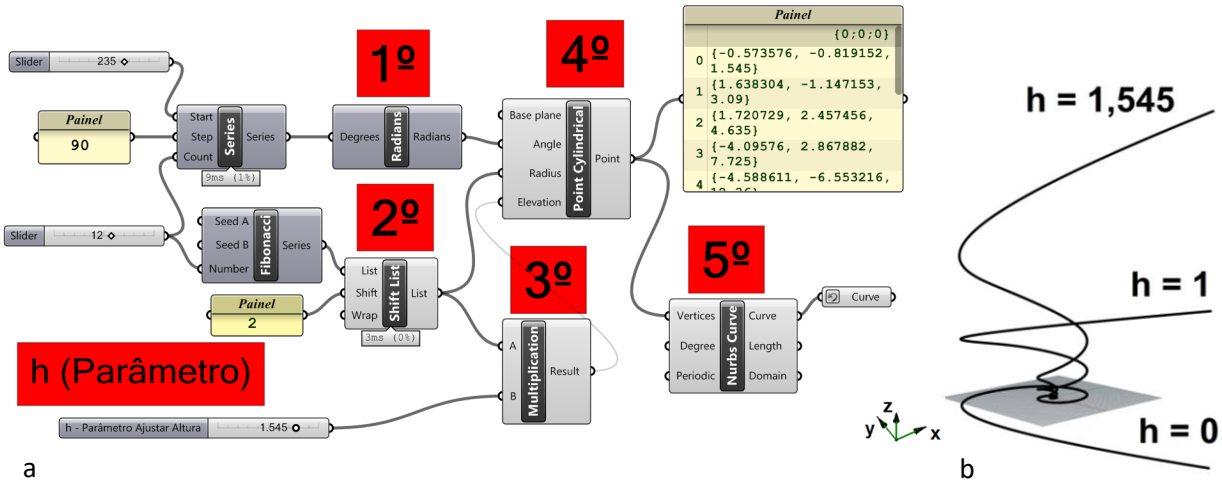
de produto pelo método da Engenharia Reversa. O processamento da malha também foi realizado no *software Geomagic Studio*, conforme (SILVA, 2006).

> "A construção de uma malha de triângulos planos é realizada através da união, três a três, dos pontos constituintes da nuvem. Normalmente na malha gerada há a presença de algumas descontinuidades (furos) que precisam ser corrigidas. O fechamento de furos ocorre pela interpolação de pontos (vértices dos triângulos) seguindo a curvatura de suas regiões adjacentes. Pode-se aplicar um filtro para redução do número de triângulos com vistas a obter uma malha mais leve e fácil de manipular. Eventuais intersecções entre triângulos também devem ser removidas, o que é realizado excluindo os triângulos em intersecção e realizando uma operação de fechamento de furos. Por fim pode-se realizar uma suavização na malha, ou seja, a re-orientação de alguns triângulos tornando suas arestas mais tangentes à vizinhança, operação que melhora o aspecto da superfície" (SILVA, 2006).

O método para análise e construção da "Curva Geométrica" parte do princípio de que é possível identificar na malha de triângulos os pontos que formam os frutículos característicos do Abacaxi. Ligando estes pontos por uma "polilinha" (ferramenta de desenho do CAD *Rhinoceros*) é possível desenvolver uma curva na forma de espiral 3D, essa curva será chamada de "Curva Polilinha". Utilizando o método de observação descrito em Kindlein Júnior et al. (2002) é possível identificar nesta curva as características da espiral áurea da série de Fibonacci e a partir desta dedução, com auxilio da plataforma paramétrica (*Rhinoceros* com o *Grasshopper*), modelar com exatidão as curvas livres e também as curvas definidas por equações paramétricas conforme descritas por Yiwei & Xia (2010).

O método para modelagem da curva definida por equações paramétricas resultará na "Curva Paramétrica" da espiral 3D. Para tanto, utilizou-se dos recursos disponíveis no *Grasshopper,* Figura 14. Assim, o método parte do princípio da seleção de parâmetros de programação que são disponibilizados em listas de números conectados a função chave "*Point Cylindrical*" resultando em uma nova lista com as coordenadas dos pontos que vão formar a "Curva Paramétrica" da espiral 3D.

Figura 14. Programação *Rhinoceros* e *Grasshopper* da Espiral Áurea.



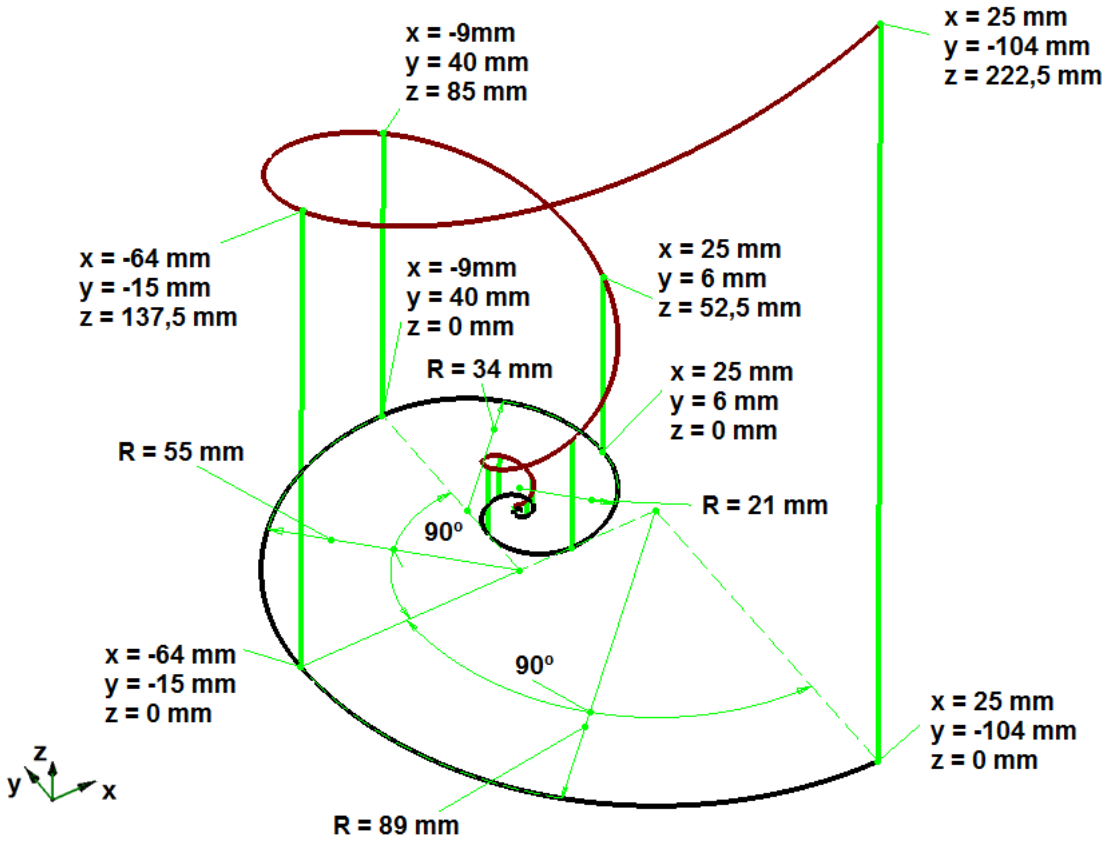
Programação da espiral Áurea: a) fases da programação com as funções aplicadas em: 1º *"Radians"*, 2º *"Shift List"*, 3º *"Multiplication"*, 4º *"Point Cylindrical"* e 5º *"Nurbs Curve"*; b) resultado da Programação das curvas com diferentes alturas "h - Parâmetro Ajustar Altura".

Detalhando o procedimento de modelagem paramétrica via *Grasshopper*, o 1º passo foi a criação de uma série de números (Fibonacci) e converte-los em radianos. Em paralelo, o 2º e o 3º passos consistem em selecionar os valores da série de Fibonacci e multiplicar pelo valor "h - Parâmetro Ajustar Altura", que é definido manualmente pelo usuário. A partir das listas, o 4º passo foi da conexão com a ferramenta de função "*Point Cylindrical*", a qual recebe os parâmetros das séries de números das funções anteriores e projeta uma lista de pontos variando no espaço em função do ângulo, raio e elevação (linear). Estes pontos servem para programação da espiral logarítmica no 5º passo, pela função "*NURBS Curve*" que vai definir a "Curva Paramétrica". O ajuste paramétrico da curva é definido variando o valor de "h - Parâmetro Ajustar Altura", para h=0 o desenho resultante será a espiral de Fibonacci 2D (plano XY), variando o valor para h>0 o resultado será o desenho de uma espiral 3D (espaço XYZ). Essa sequência de comandos é detalhada no Apêndice A (página 94).

Para que a curva tenha a mesma orientação de crescimento definida na série de Fibonacci, é necessário fazer o controle dimensional, avaliar a posição e a exatidão da "Curva Paramétrica". Conforme apresentado na Figura 15, o

procedimento consiste em traçar linhas verticais passando nos pontos de ligação dos raios e verificar a posição de intersecção destas linhas com as curvas. No caso de discordâncias deve-se corrigir os parâmetros ou incrementar funções de programação.

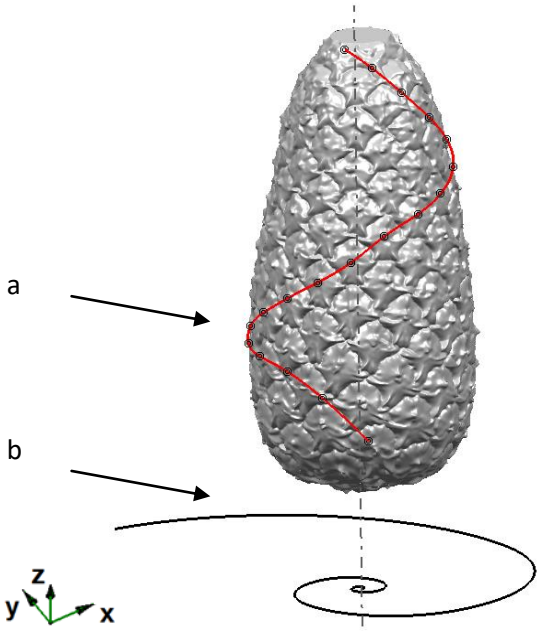
Figura 15. Avaliação da "Curva Paramétrica".



Avaliação da "Curva Paramétrica" por projeção de linhas verticais nos pontos finais (*end points)* dos arcos.

Na sequência, o procedimento para orientação espacial dos objetos Modelo 3D Digitalizado (Malha) e "Curva Polilinha" com a "Curva Paramétrica" servirá de referência para construção do Modelo 3D Final (Superfícies Paramétricas). Na Figura 16, o procedimento inicial do alinhamento é definir o eixo de simetria da malha do abacaxi, posicionando-o com o centro da "Curva Paramétrica", em seguida mover e rotacionar no eixo "Z" da malha de tal forma que se tenha a sobreposição das curvas "Polilinha" e "Paramétrica".

Figura 16. Alinhamento Entre a Malha e as Curvas.



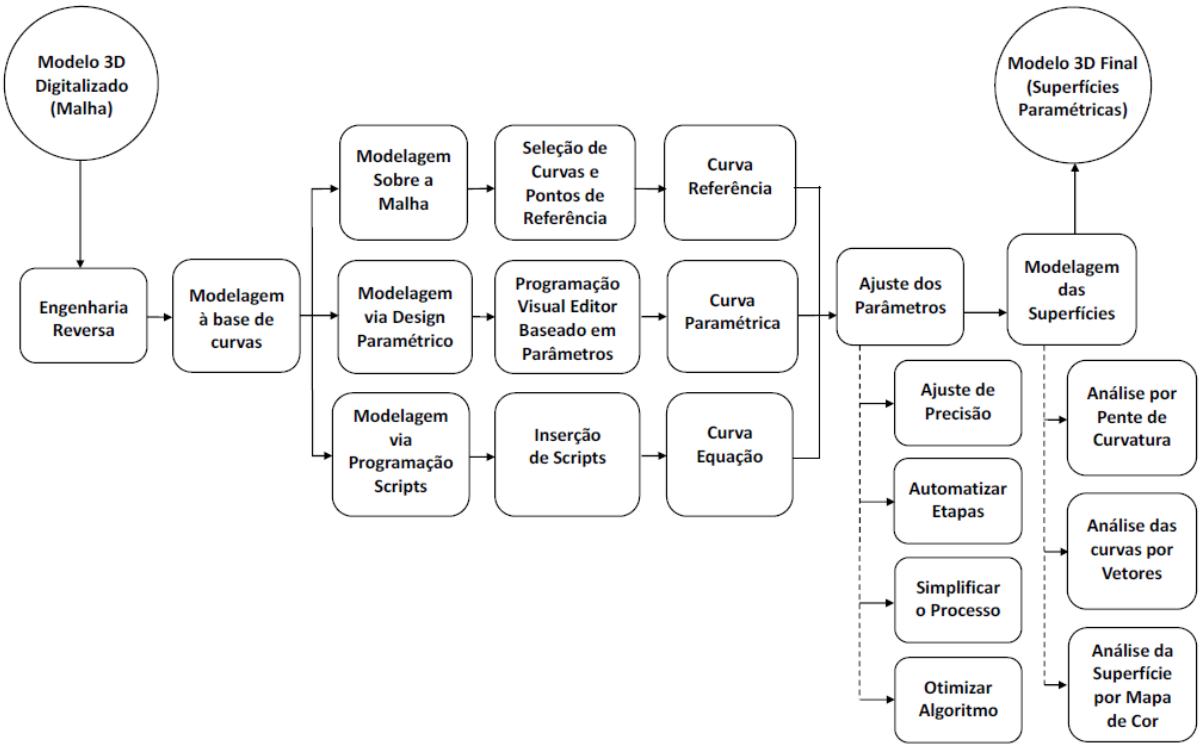
Alinhamento: a) malha de triângulos, “Curva Polilinha” projetada sobre os frutículos; b) “Curva Paramétrica” da espiral de Fibonacci (h=0).

Para a inspeção da curva, aplicou-se a ferramenta pente de curvas, conforme procedimento utilizado por De Aviz (2010). Foi realizado um estudo e análise da principal curva geométrica que define a orientação espacial da malha e a sequência das tarefas para a geração do modelo final. A modelagem da curva foi realizada por três formas distintas. A primeira é a modelagem sobre a malha, por meio do *software Rhinoceros,* selecionando pontos de referência e modelando por uma curva polilinha. A segunda modelagem foi via programação visual baseada em funções e parâmetros no *software Grasshopper*, resultando uma “Curva Paramétrica”. A terceira modelagem se deu via programação por *Scripts*, em editor de texto de função por declaração de variáveis com a linguagem de programação (*Python*), resultando numa curva definida por uma equação matemática. Independente do método utilizado, o controle e ajuste dos parâmetros que formam as curvas são importantes para orientar o designer no processo de criação. Para o controle e validação da geometria deve ser possível ajustar a espiral de maneira que proporcione qualidade e precisão ao processo de modelagem das curvas que servirão de base para a modelagem das superfícies paramétricas que, por sua vez, vão formar o modelo 3D.

Para inspeção das superfícies, foram utilizadas as ferramentas de análise por mapa de cores e vetor de distância entre secções para avaliar a aproximação dos modelos 3D (malha e superfícies paramétricas). Em Minetola et al. (2015) os autores apresentam o método da inspeção por comparação entre a malha e o modelo CAD, o resultado da inspeção indica o desvio da superfície por um mapa 3D de cores e por vetores do desvio das secções 2D das geometrias. No presente trabalho, este método foi aplicado utilizando o *software Geomagic Qualify*.

Por intermédio das ferramentas de análise é possível verificar a qualidade do projeto e a aproximação entre o Modelo 3D Digitalizado (Malha) e o Modelo 3D Final (Superfícies Paramétricas). Esta avaliação proporcionará a validação de todo o método proposto. A Figura 17 apresenta o fluxo de desenvolvimento proposto neste trabalho para a geração de superfícies paramétricas a partir da digitalização 3D de geometrias da natureza com padrão de crescimento espiral.

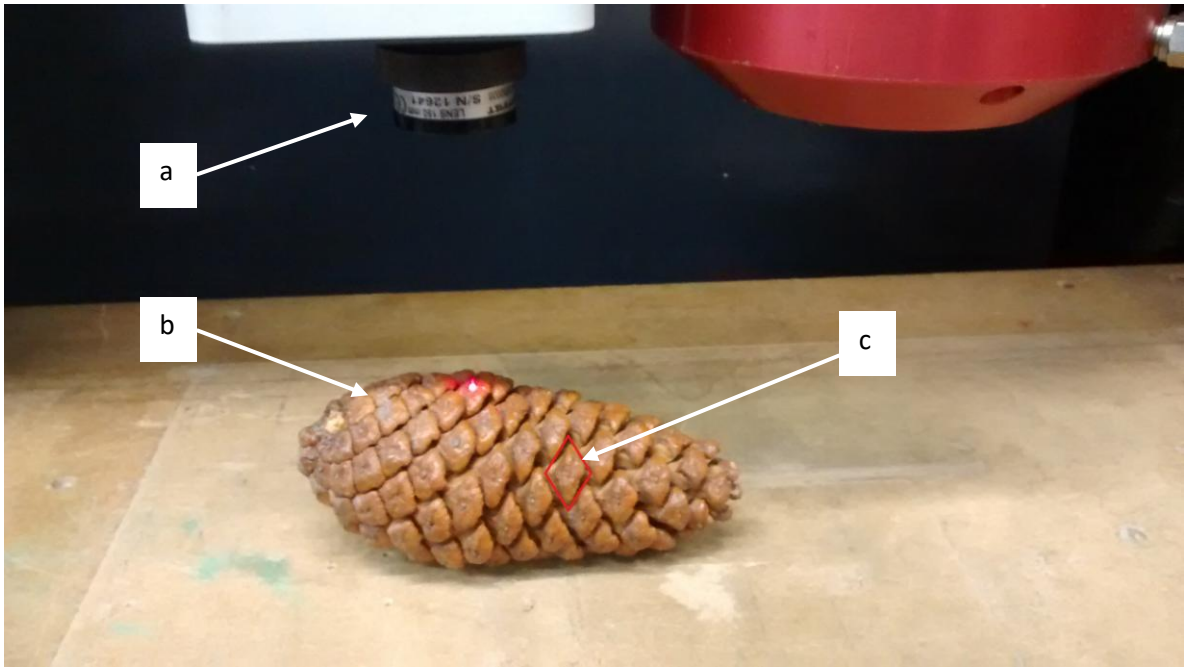
Figura 17. Fluxo de Desenvolvimento.



Fluxo de Desenvolvimento da fase de Engenharia Reversa. Entrada do Modelo 3D Digitalizado e a Malha da digitalização até a saída do Modelo 3D Final com as Superfícies Paramétricas.

Após as análises e determinação das curvas mais exatas e do melhor método de geração, o fluxo de desenvolvimento apresentado foi otimizado por programação das funções de desenho e alinhamento por programação via script em Python. O método da modelagem da curva com padrão de crescimento de Fibonacci foi repetido para três amostras de abacaxi e mais três amostras de pinha (Figura 18). Dessa forma, foi possível verificar a repetibilidade do método proposto.

Figura 18. Digitalização 3D da Pinha com o Scanner Digimill 3D.



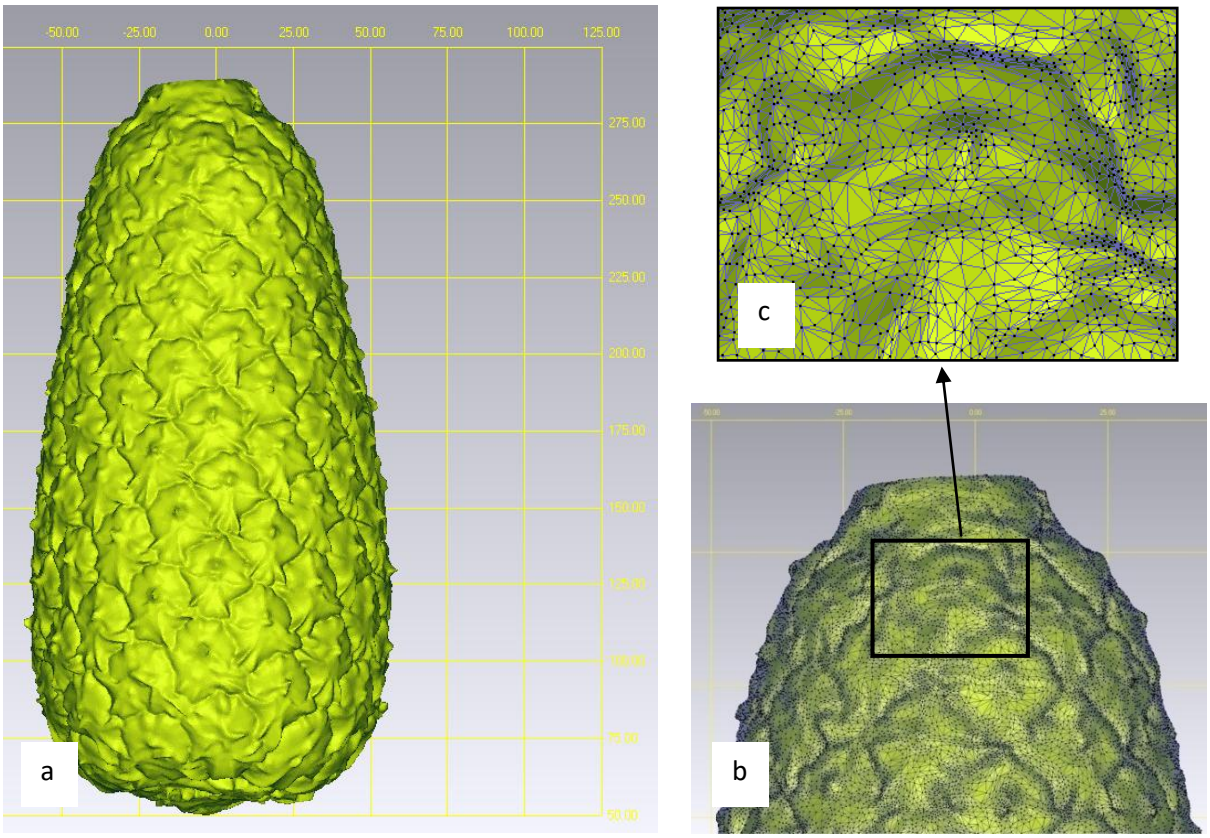
Sistema de Digitalização utilizado: a) cabeçote com sensor conoscópico a Laser 3D OPTIMET, instalado na máquina de usinagem com deslocamento controlado Digimill 3D; b) Pinha amostra para digitalização; c) Escamas da Pinha.

Fonte: LdSM/UFRGS.

4. RESULTADOS

Os resultados obtidos indicam a viabilidade técnica da aplicação do método proposto. Na imagem ampliada, Figura 19 (c), observa-se que cada triângulo é formado pela ligação de três pontos de vértices que foram originados do *scanner* a Laser.

Figura 19. Malha de Triângulos.

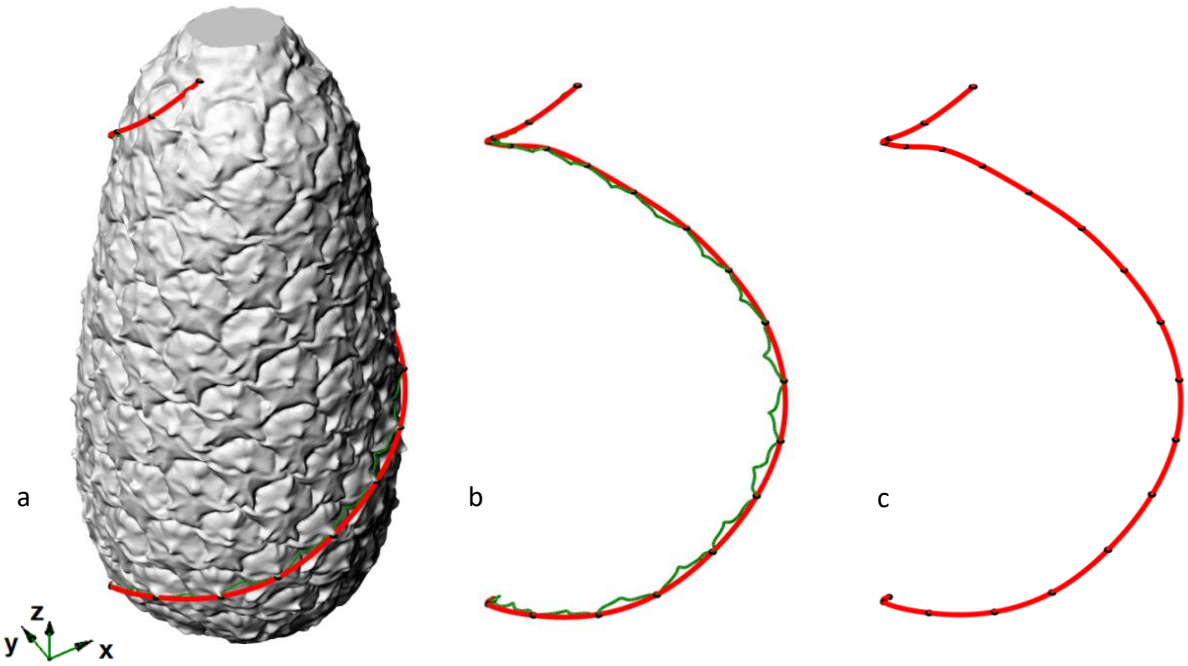


Malha de triângulos do Abacaxi: a) malha completa com o *grid* da escala com espaçamento de 25 mm; b) ampliação da região superior da amostra; c) destaque dos triângulos planos formados pela ligação dos pontos.

Após a digitalização da amostra foi realizado o procedimentos de tratamento da nuvem de pontos e geração da malha no *software Geomagic Studio*. O resultado foi armazenado em um arquivo digital na extensão ".stl". Por meio da importação do arquivo no *Software* de CAD *Rhinoceros* foi iniciado o estudo e a modelagem com a

ferramenta de curva sobre a malha (*Polyline OnMesh*), definindo o relevo dos frutículos. Na sequência, utilizando o comando "*Divide*" foi segmentada a polilinha em pontos distantes de 1 mm. Esta segmentação serviu para modelagem de uma curva utilizando o comando "*Curve: interpolate points*" com a função "*Osnap point*" habilitado. Para modelar a curva contínua chamada de "Curva Polilinha", bastou selecionar manualmente o ponto sobre o relevo do frutículo, conforme resultado apresentado na Figura 20.

Figura 20. Modelagem da Curva sobre Malha.

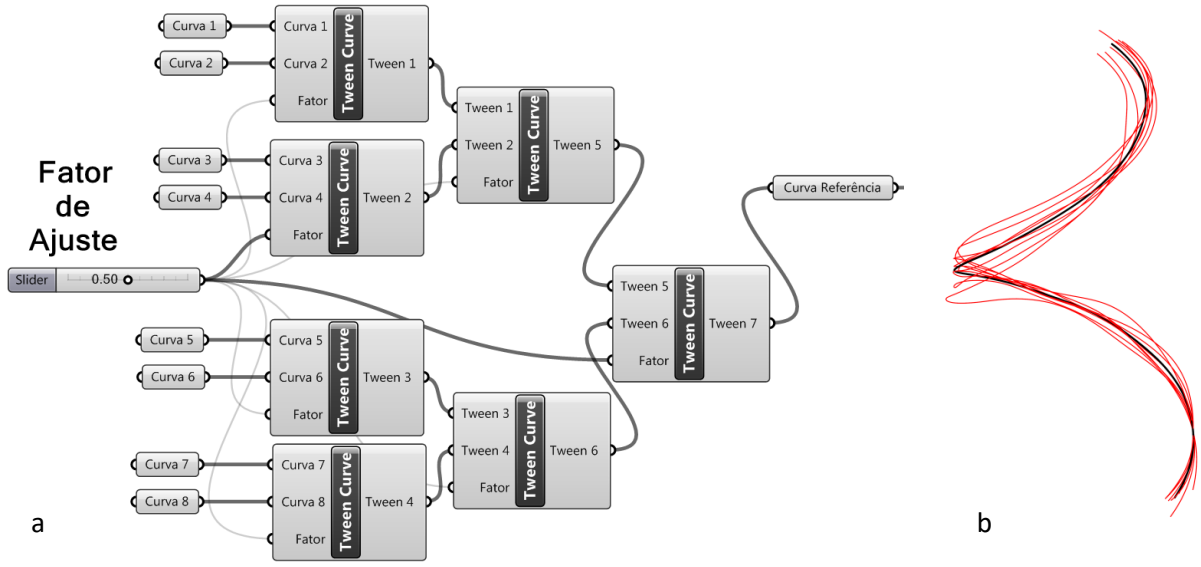


Curva projetada sobre a malha de triângulos: a) malha completa com a curva projetada; b) curva projetada sobre a malha e a "Curva Polilinha" projetada sobre os relevos dos frutículos; c) destaque da "Curva Polilinha".

O procedimento da modelagem da "curva polilinha" foi repetido para as 8 sequências de padrões espirais formados pelos frutículos de um abacaxi. As curvas foram alinhadas no *Rhinoceros* com o comando "*Orient*" por "3 *points*". São apresentadas na Figura 21 (b) as 8 curvas polilinhas e em cor diferente a curva média chamada de "Curva Referência". Dessa forma, por meio do *Grasshopper* foi realizada seleção "*Set On Curve*" das 8 curvas (*curve* 1 até *curve* 8) em seguida foi realizada a interpolação das 8 curvas de duas em duas até resultar uma única "Curva Referência". Foi utilizado a função "*Tween Curve*" com fator de ajuste de 0,5, conforme observa-se na Figura 21 (a). A função "*Tween Curve*" vai receber a

entrada de duas curvas e interpolar uma curva Média conforme o valor definido no "*Slider*" para determinar o fator de aproximação.

Figura 21. Programação Visual da "Curva Referência".

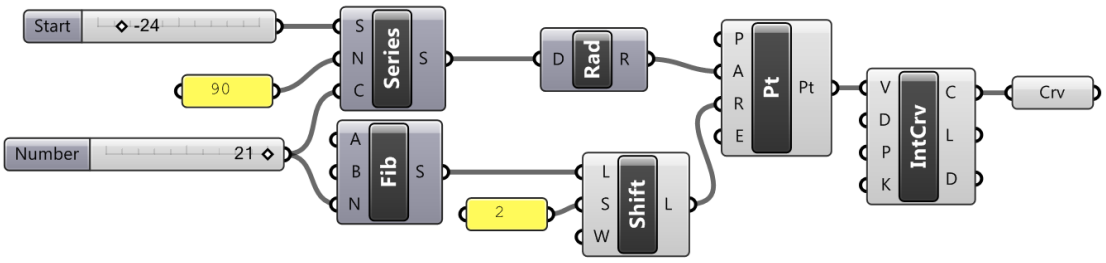


Programação no *Grasshopper* da "Curva Referência": a) sequência das funções da programação no *Grasshopper* (fator de ajuste, seleção das curvas, comando "*Tween Curve"* com função de fundir duas curvas e retornar uma curva corrigida pelo fator definido; b) visualização da sobreposição das 8 curvas com a "Curva Referência".

4.1 Modelagem Paramétrica

Após a construção da "Curva Referência", o passo seguinte foi de desenhar na plataforma *Rhinoceros* a curva de Fibonacci 2D e com o *Grasshopper* a "Curva Paramétrica". O principal comando "*Point Cylindrical*" projetou os pontos em um cilindro variando o ângulo, o raio e a elevação. A sequência da programação é detalhada na Figura 22.

Figura 22. Programação Visual da Curva Espiral "*Point Cylindrical"*.

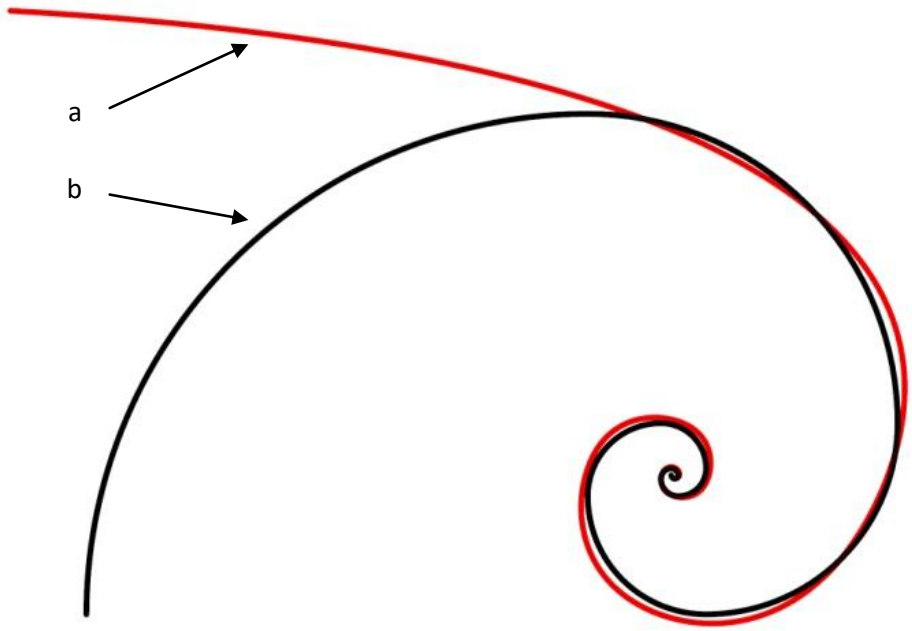


Programação da curva espiral com a função *"Point Cylindrical"*.

De acordo com a Figura 22, a programação parte de um número (*Number*) para seleção do início, passo e sequência de números da série de Fibonacci (*Fib*). A função *Radians* (*Rad*) converte a lista de números (*Series*) de graus de rotação em Radianos. A função *Shift List* (*Shift*) seleciona os itens da lista que serão projetados em um cilindro variando o ângulo, raio e elevação pelo comando "*Point Cylindrical*" (*Pt*). O processo é finalizado com a função *Interpolate Curve* (*IntCrv*) para a representação da geometria em forma de curva.

O resultado da programação (Figura 23) via comando "*Point Cylindrical*" foi uma curva, que quando sobreposta à curva de Fibonacci 2D apresentou variação, curvatura e afastamento tangencial no final.

Figura 23. Sobreposição da Curva Espiral "*Point Cylindrical*".



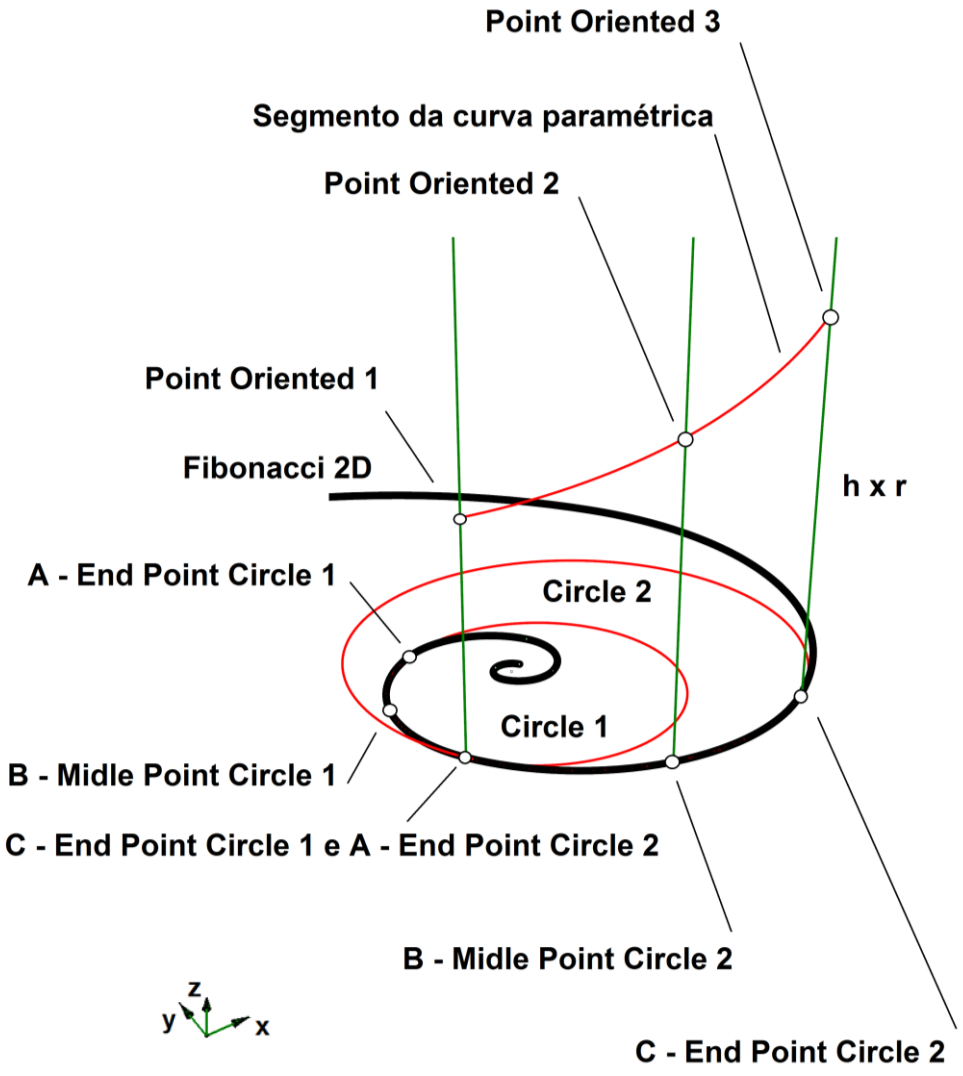
Sobreposição das curvas: a) curva projetada sobre o plano XY resultante da programação "*Point Cylindrical*"; b) curva formada pela espiral 2D de Fibonacci.

A diferença entre as curvas, Figura 23, desencadeou a necessidade da programação de uma nova alternativa de curva. Optou-se por manter o conceito de projeção, variando o ângulo, o raio e a elevação. Nesse sentido, o método adotado foi de traçar linhas verticais saindo dos vértices dos raios da espiral de Fibonacci, elevando estes pontos na proporção do fator de correção da altura "h". Assim, foram

gerados pontos no espaço, que ligados por uma curva interpolada formaram a nova “Curva Paramétrica”.

A base da programação partiu do desenho do *Rhinoceros,* posicionando pontos nos “*End point*” e “*Midle Point*” de cada arco da espiral de Fibonacci 2D, conforme Figura 24 (A - *End Point Circle* 2, B - *Midle Point Circle 2 e* C - *End Point Circle* 2). Estes pontos foram transladados no eixo “z” gerando novos pontos (Point Oriented 1, Point Oriented 2 e Point Oriented 3). A união destes novos pontos formará um arco que é o segmento da “Curva Paramétrica”, conforme a programação visual apresentada na Figura 25.

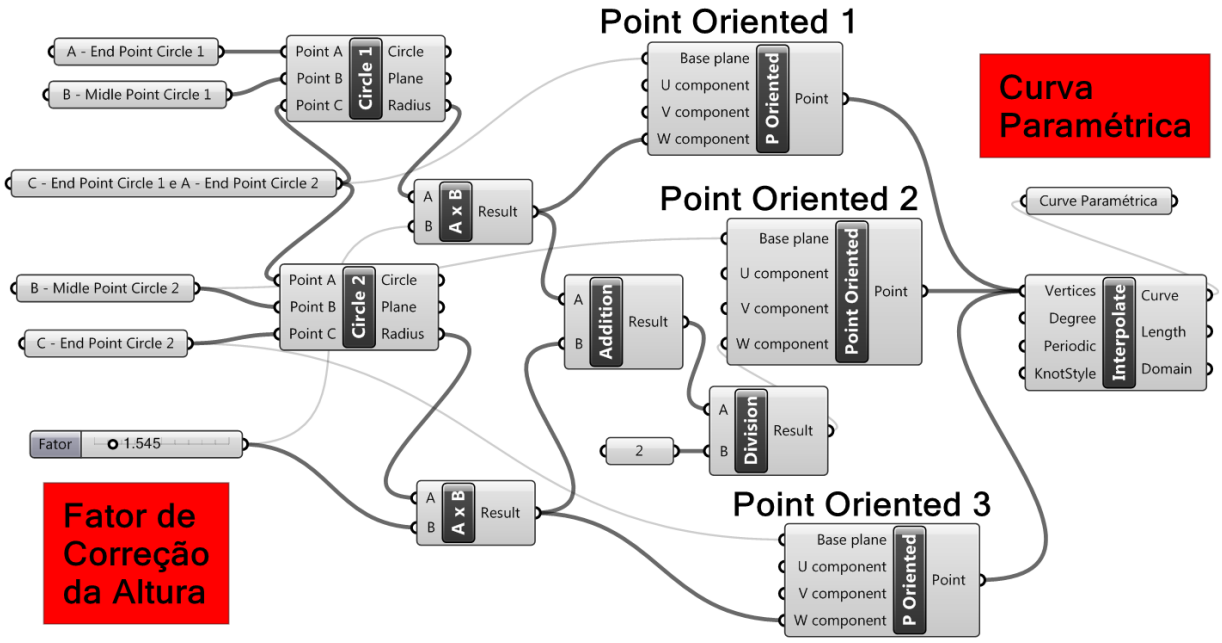
Figura 24. Representação de um Arco da “Curva Paramétrica”.



Representação gráfica do segmento da “Curva Paramétrica”.

O parâmetro da coordenada “z” (h x r) dos pontos (*Point Oriented 1 e Point Oriented 3*), apresentado na Figura 25, foi definido pela multiplicação do “Fator de Correção da Altura” pelo raio do círculo correspondente ao ponto sobre a Fibonacci 2D. A descrição dos comandos utilizados encontra-se no Apêndice A (página 96).

Figura 25. Programação Visual de um Arco Paramétrico.

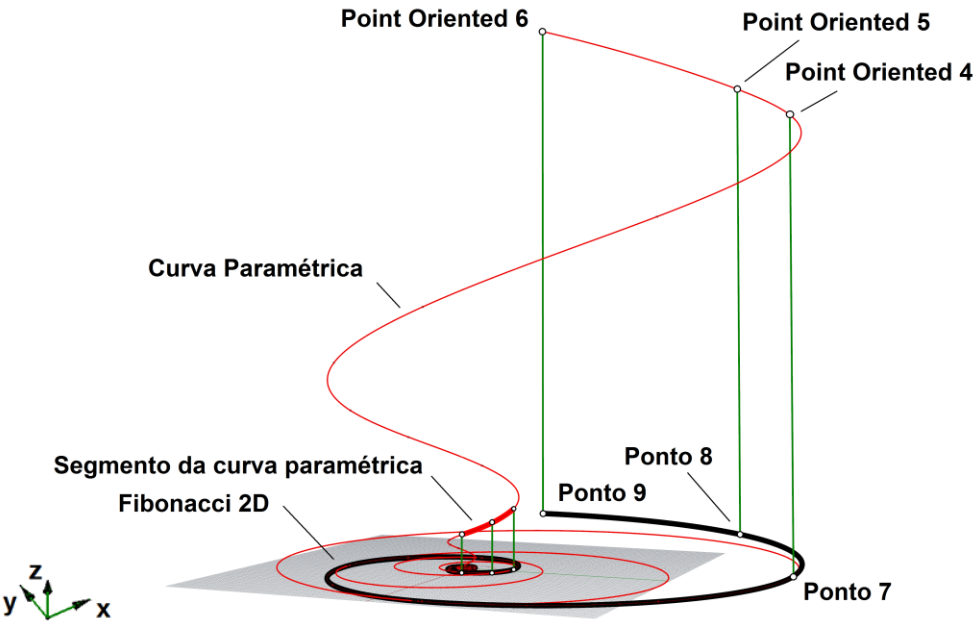


Programação *Grasshopper* do segmento da “Curva Paramétrica”.

Para definição da coordenada “z” do ponto “B - *Midle Point Circle 2*” foi utilizado uma sequência *matemática para equacionar a média entre os raios dos círculos (Circle* 1 e Circle 2) multiplicado pelo “Fator de Correção da Altura” e, assim, atribuir o resultado na função “w” do comando “*Point Oriented 2*”.

O procedimento foi repetido até o arco de raio 144 mm da série de Fibonacci, conforme a Figura 26, que apresenta os pontos (7, 8 e 9) onde estes foram transladados no eixo z com o valor do fator de correção da Altura multiplicado pelo raio, originando, assim, novos pontos (*Point Oriented 4, 5 e 6*).

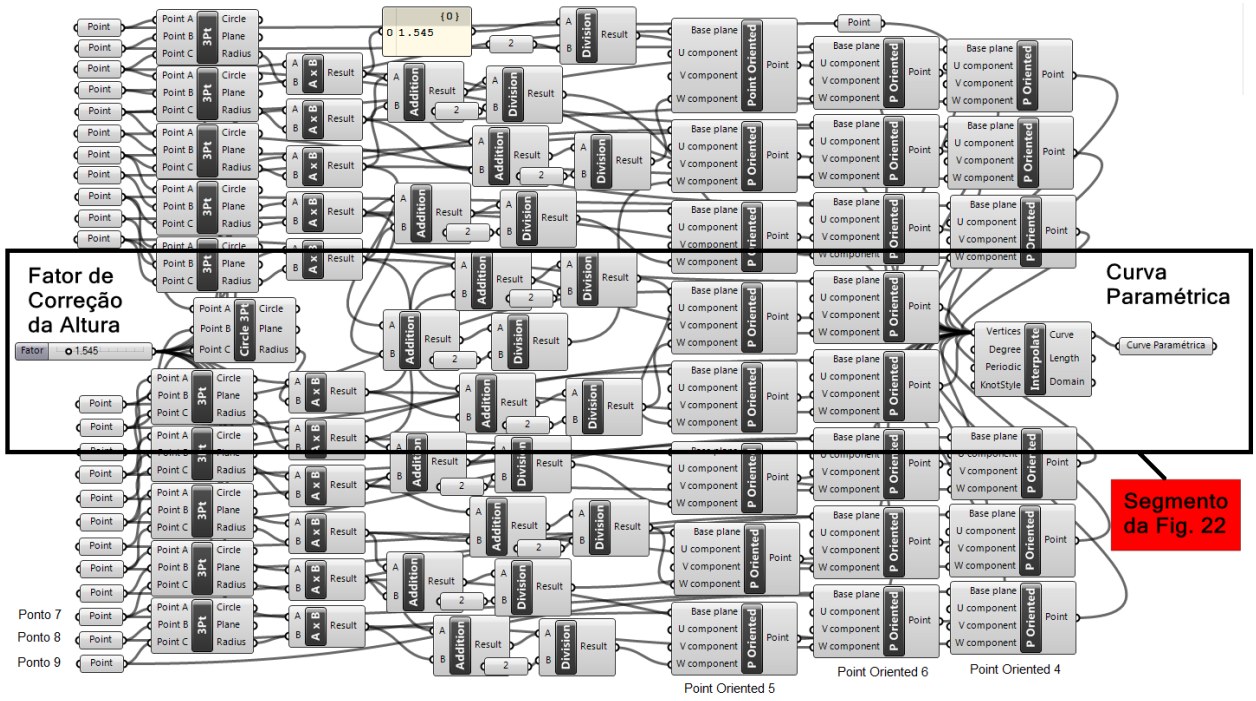
Figura 26. Representação da "Curva Paramétrica" com *Grasshopper*.



Representação Gráfica da programação completa da "Curva Paramétrica".

Após a programação da posição dos diversos pontos no espaço, conforme apresentado na Figura 27, com a função "*Interpolate*", foi definida a nova "Curva Paramétrica". A descrição dos comandos utilizados encontra-se no Apêndice A (página 97).

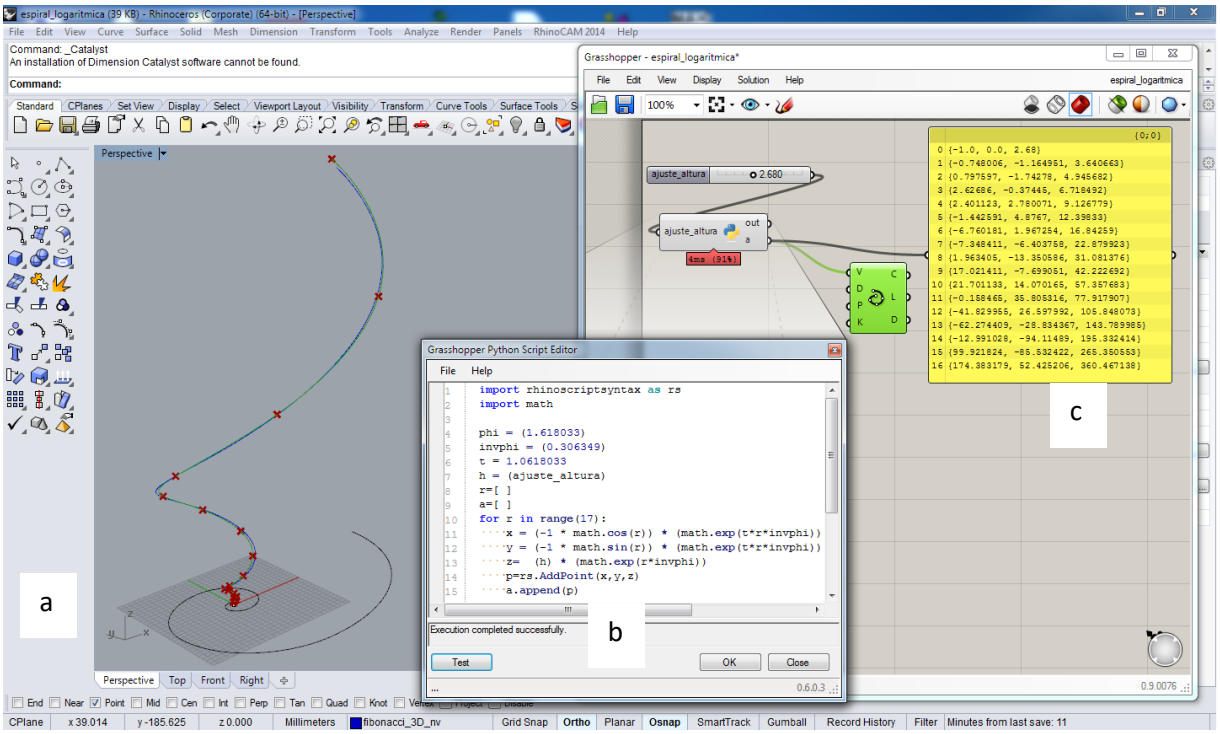
Figura 27. Programação Visual Completa da "Curva Paramétrica".



Programação completa da "Curva Paramétrica" no *Grasshopper*.

Uma terceira modelagem pelo método de Design Paramétrico foi realizada utilizando a programação por *Scripts*, com inserção de código de programação em editor de função por declaração de variáveis, via linguagem de programação *Python*, que resultou numa curva definida por uma equação matemática. Na Figura 27, está representada a "Curva Equação" modelada também parametricamente via programação por *Scripts* no *Grasshopper* com a linguagem *Python Script*. A Figura 28 (a) apresenta a janela do *software Rhinoceros* com a sobreposição das curvas que formam a espiral 2D de Fibonacci, a "Curva Paramétrica" e a "Curva Equação". Inicialmente, foi identificada uma aproximação refinada entre as curvas 3D. A Figura 28 (b) apresenta a janela do Editor *Python Script* com as linhas de programação. A Figura 28 (c) apresenta a janela do *Grasshopper* com o comando "*Slider*", Editor "*Python Script*", "*Curve Interpolate*" e o painel de informações.

Figura 28. Programação no *Grasshopper* com *Python Script*.



Programação em *Python Script* da "Curva Equação": a) Visualização Gráfica da sobreposição do resultado das curvas Fibonacci 2D, "Curva Paramétrica" e "Curva Equação"; b) Janela do Editor *Python Script;* c) janela com as funções de programação paramétrica do *Grasshopper.*

Conforme realizado no Editor *Python Script*, a Tabela 1 apresenta as linhas de programação da "Curva Equação". A referência da programação está baseada na

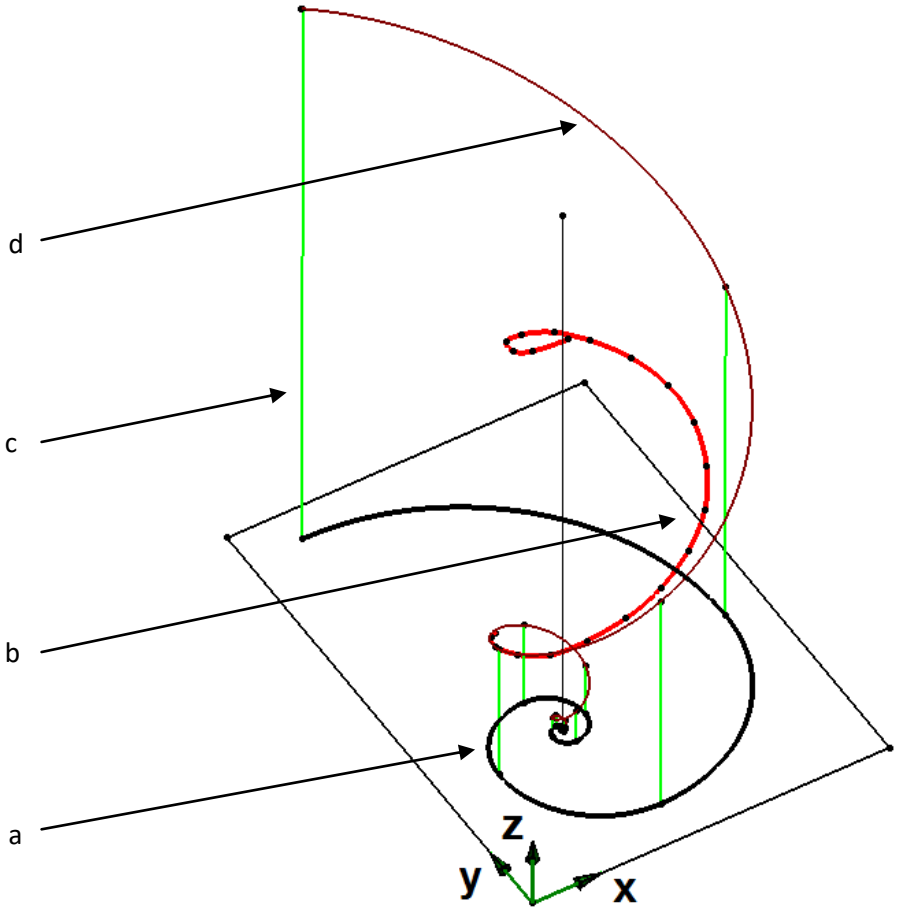
Equação 11 (pag. 26). Para esta modelagem, inicialmente foram definidas as variáveis globais e, a partir do valor de Phi, as equações de x, y e z da programação. Essas equações (Tabela 1) resultam na sequência de pontos distribuídos no espaço, formando uma espiral 3D com a forma desejada.

Tabela 1. Programação de uma Curva em *Python Script.*

COMANDOS	COMENTÁRIOS
import ***rhinoscriptsyntax*** *as* ***rs***	Chamada das funções específicas da
import ***math***	biblioteca do *Rhinoceros*
# Phi = (1.618033)	Comentários: Phi = número áureo
	Variáveis Globais:
invPhi = (0.306349)	InvPhi = resultado da equação logarítmica
t = 1.0618033	t = aproximação
h = (ajuste_altura)	h = parâmetro de altura
r = [ ]	r = raio
a = [ ]	a = saída
for r in range(17):	Número de pontos para representação da curva
	Equação:
x = (-1 * math.cos (r)) * (math.exp (t*r*invPhi))	Funções para Declaração das Variáveis (x,y,z)
y = (-1 * math.sin (r)) * (math.exp (t*r*invPhi))	Equação: $r(t) = r_0 * e^{(t * 0,30635)}$
z = (h) * (math.exp (r*invPhi))	
p = **rs.AddPoint**(x,y,z)	Função da adição dos Pontos com coordenadas (x,y,z)
a.**append**(p)	Saída Gráfica dos Pontos

Após a obtenção da “Curva Referência” e da “Curva Paramétrica”, segue o alinhamento das curvas, cujo método utilizado foi de mover e rotacionar no eixo Z (eixo normal saindo da origem) a “Curva Referência” até o momento da sobreposição, Figura 29.

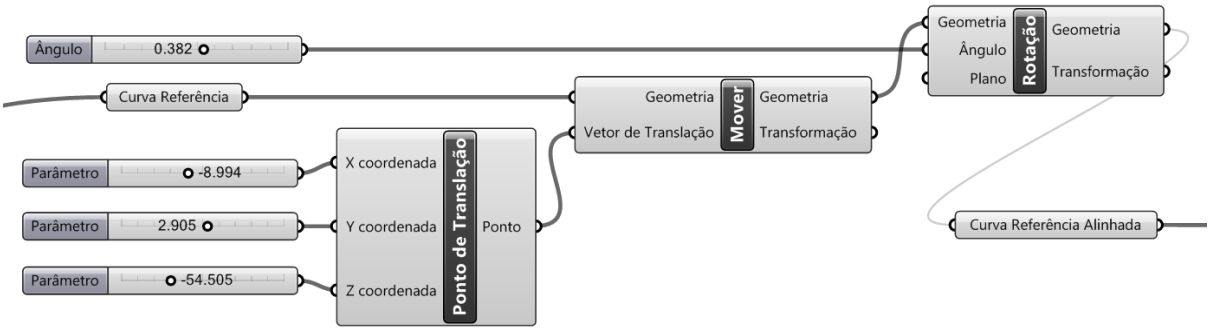
Figura 29. Aproximação da "Curva Referência".



Alinhamento das curvas em: a) Curva Fibonacci 2D; b) "Curva Referência" alinhada; c) vetor do eixo Z para projeção dos pontos; d) "Curva Paramétrica".

Na Figura 30, é apresentada a programação no *Grasshopper* para o alinhamento da "Curva Referência". A descrição dos comandos utilizados encontra-se no Apêndice A (página 98).

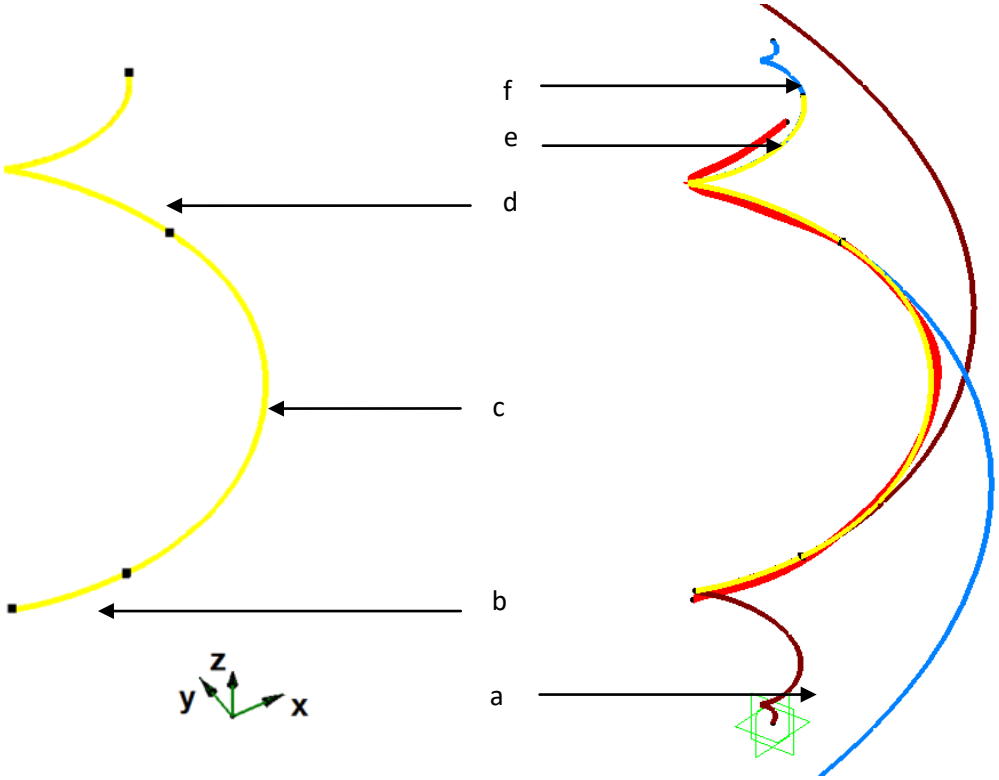
Figura 30. Programação Visual para os Parâmetros do Alinhamento.



Programação *Grasshopper* para alinhamento da "Curva Referência".

O passo seguinte foi a conclusão da “Curva Paramétrica” que consiste na tarefa de espelhar, cortar e unir as curvas obtidas. A primeira etapa é espelhar a curva nos planos XY e YZ e elevar no eixo z, posicionando de modo que a “Curva Paramétrica Espelhada” fique tangenciando a “Curva Referência”. A segunda fase é cortar e segmentar as curvas com a dimensão escolhida. A terceira fase é ligar os seguimentos através de uma “Curva Tangente” com curvatura ajustada. A montagem das curvas após a sequência de passos está representada na Figura 31.

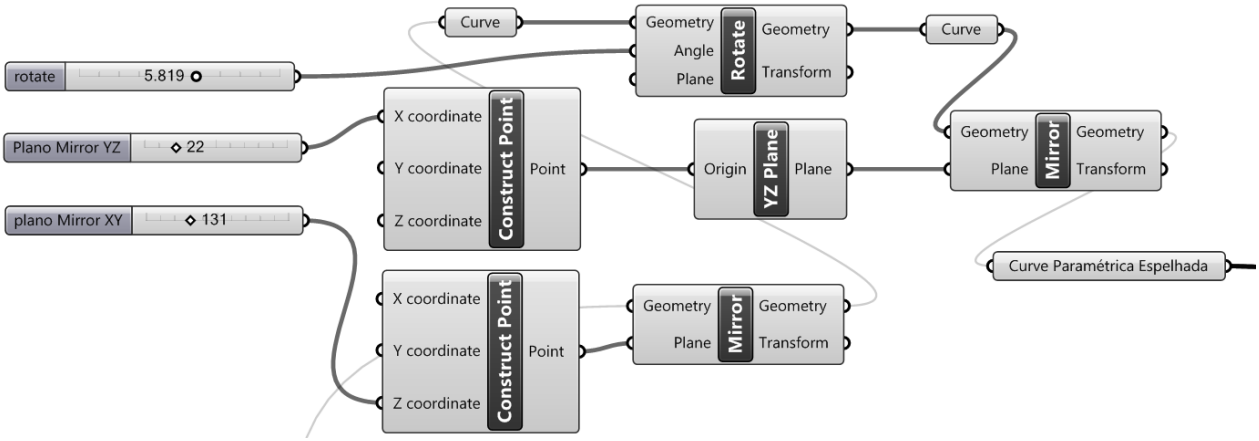
Figura 31. Montagem das Curvas.



Alinhamento das curvas em: a) “Curva Paramétrica”; b) segmento da “Curva Paramétrica”; c) “Curva Tangente”; d) segmento da “Curva Paramétrica Espelhada”; e) “Curva Referência”; f) “Curva Paramétrica Espelhada”.

Conforme observado no topo da “Curva Referência”, há uma curvatura invertida, que é definida por uma “Curva Paramétrica Espelhada” apresentada pelas funções da Figura 32.

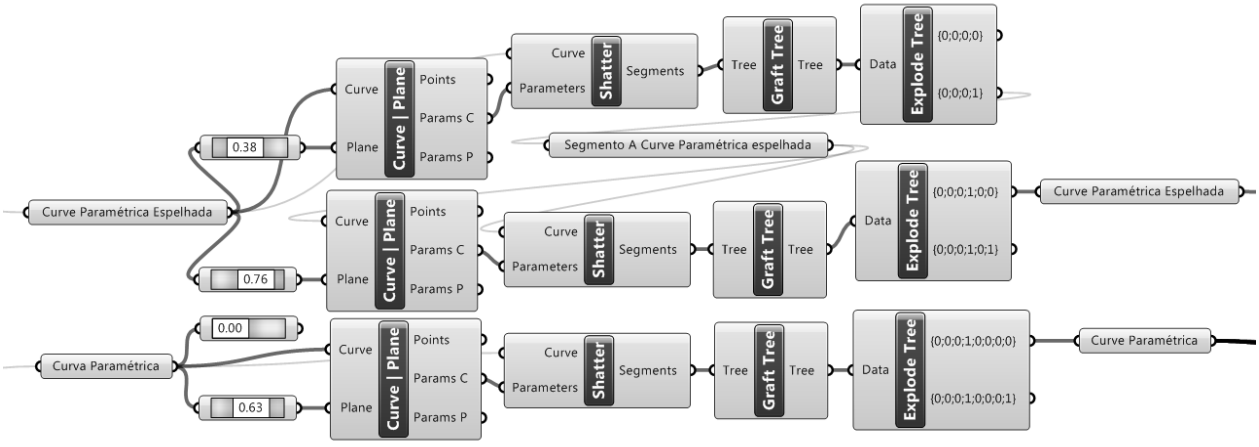
Figura 32. Programação Visual do Posicionamento da “Curva Paramétrica Espelhada”.



Programação *Grasshopper* para posicionamento da “Curva Paramétrica Espelhada”.

A programação de cortar, conforme a Figura 33, apresenta os parâmetros do posicionamento dos pontos de corte das curvas de maneira que os segmentos fiquem proporcionais e alinhados com a “Curva Referência”. A descrição dos comandos utilizados encontra-se no Apêndice A (página 99).

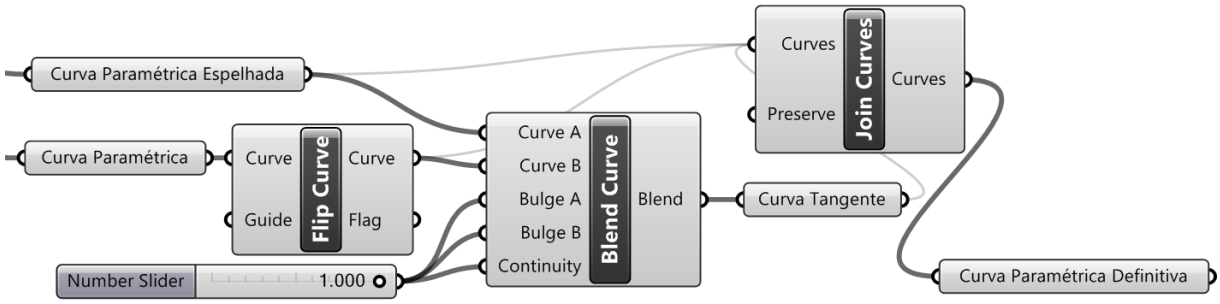
Figura 33. Programação Visual do Corte das Curvas em Segmentos.



Programação do corte em segmentos da “Curva Paramétrica Espelhada” e da “Curva Paramétrica”.

Para obtenção da “Curva Tangente”, Figura 34, foi utilizada a função “*Blend Curve*”, que é a programação de uma curva com raio definido pela tangencia e continuidade a partir das extremidades dos segmentos das duas curvas cortadas. Finaliza-se pela união dos segmentos pela função “*Join Curves*” que formou a “Curva Paramétrica Definitiva” da modelagem. A exemplo das outras figuras, a descrição dos comandos utilizados encontra-se no Apêndice A (página 100).

Figura 34. Programação Visual da “Curva Tangente”.

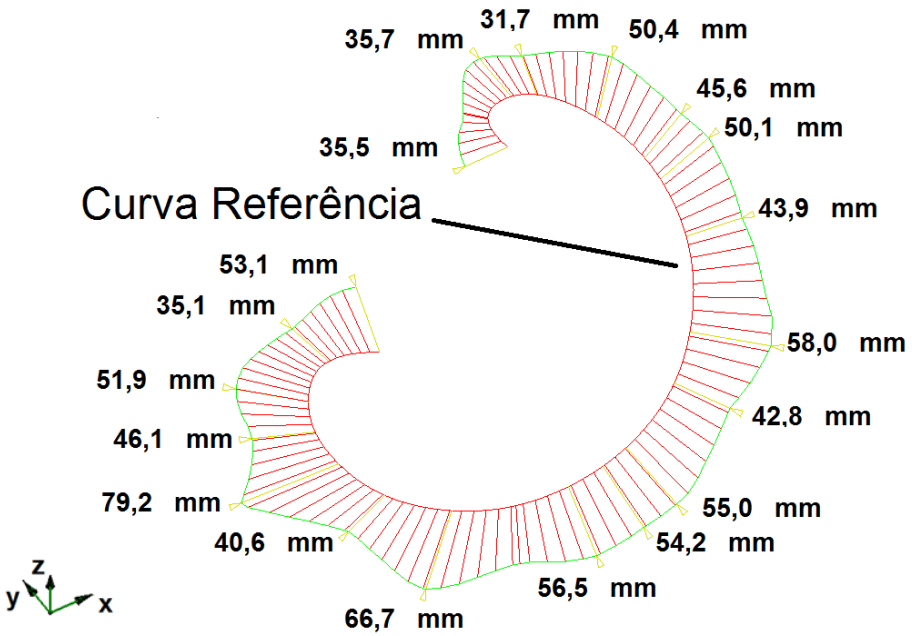


Programação *Grasshopper* para o acabamento da “Curva Paramétrica Definitiva”.

4.2 Análise dos Primeiros Modelos

Pela análise da qualidade das curvas obtidas, com a ferramenta pente de curvatura (*curve comb*) é possível identificar os vetores normais que indicam a continuidade de posição, tangência e curvatura em toda a extensão das curvas. Na Figura 35 é apresentada a análise por Pente de Curvatura da “Curva Referência” que é a média das curvas construídas no *Rhinoceros* pelo método de interpolação das oito curvas Polilinhas passando pelos Frutículos da malha do abacaxi.

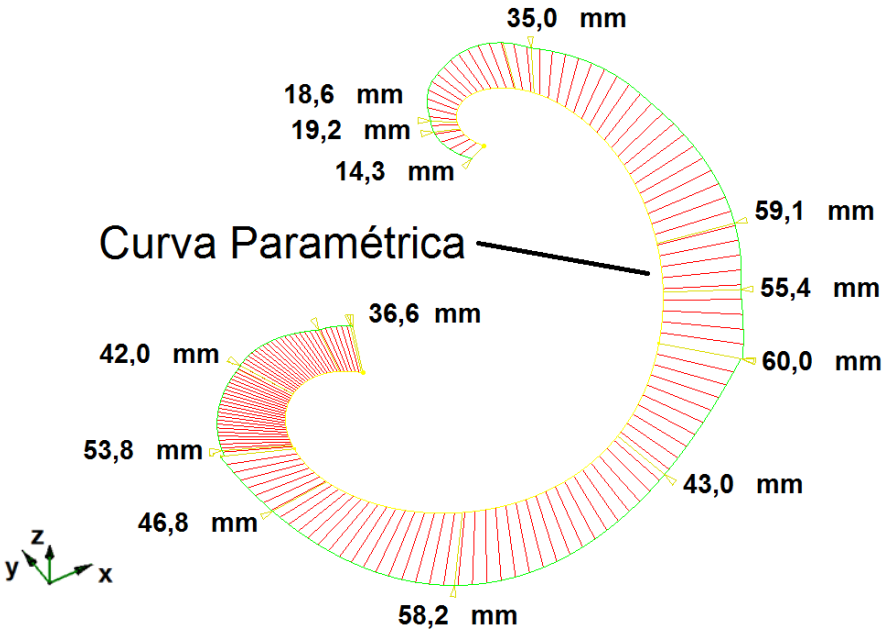
Figura 35. Análise por Pente de Curvatura da “Curva Referência”.



Ferramenta pente de curvatura para análise da continuidade de posição, tangência e curvatura da “Curva Referência”. Os valores em milímetros indicam o raio na posição indicada na seta.

Na Figura 36 é apresentada a “Curva Paramétrica” resultante da programação paramétrica *Grasshopper*.

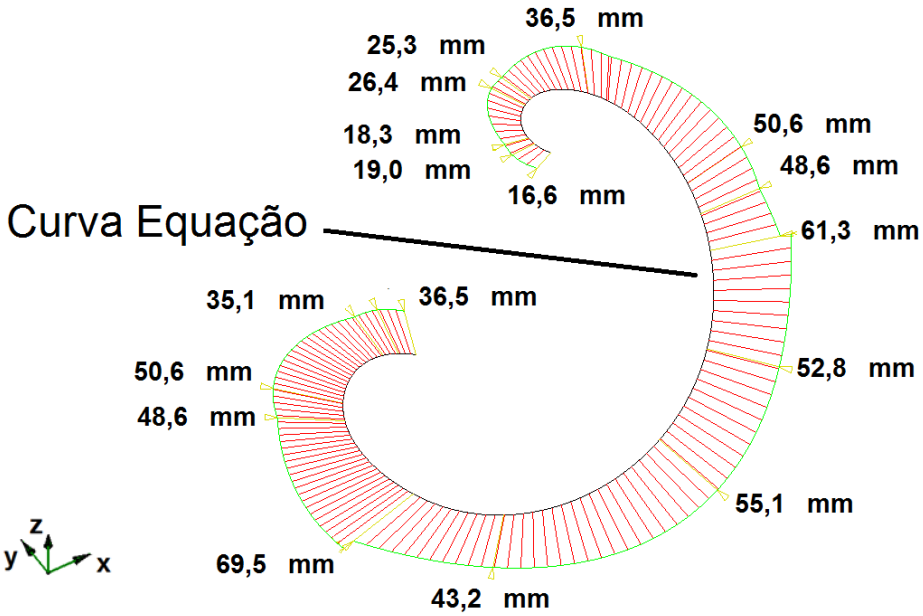
Figura 36. Análise por Pente de Curvatura da “Curva Paramétrica”.



Ferramenta pente de curvatura para análise da continuidade de posição, tangência e curvatura da “Curva Paramétrica”. Os valores em milímetros indicam o raio na posição indicada na seta.

A Figura 37 apresenta a análise da “Curva Equação” calculada por equação em *Python Script*.

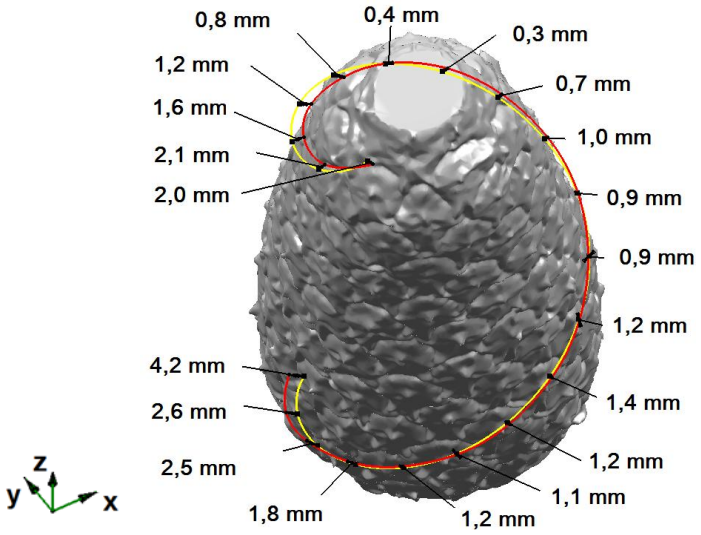
Figura 37. Análise por Pente de Curvatura da “Curva Equação”.



Ferramenta pente de curvatura para análise da continuidade de posição, tangência e curvatura da “Curva Equação”. Os valores em milímetros indicam o raio na posição indicada na seta.

Na Figura 38, é apresentada a “Curva Paramétrica” sobreposta à “Curva Referência” com as medidas das diferenças entre as curvas na posição da altura de cada frutículo. Nesta análise é avaliado o módulo dos vetores gerados partindo dos pontos de referência (Frutículos) até os pontos correspondentes normais ao eixo do Abacaxi.

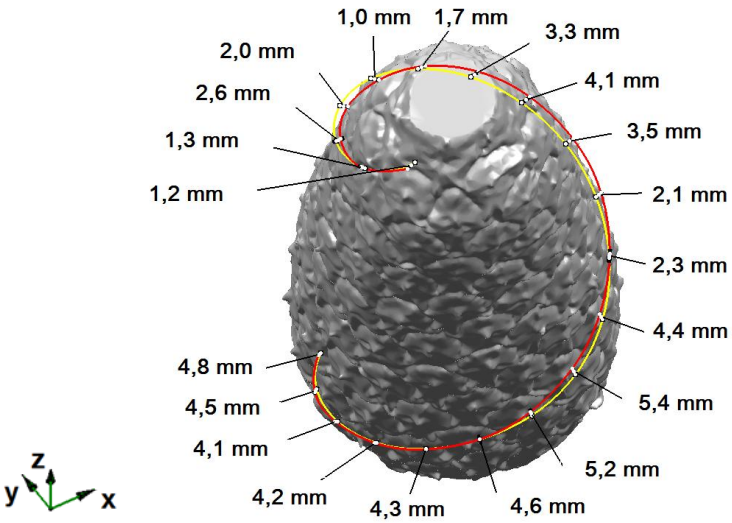
Figura 38. Análise por Vetores da “Curva Paramétrica”.



Análise entre a “Curva Referência” e “Curva Paramétrica”, por vetores de distância entre as curvas.

Na Figura 39, é apresentada a “Curvas Equação” sobreposta à “Curva Referência” com as medidas das diferenças entre as curvas na posição da altura de cada frutículo.

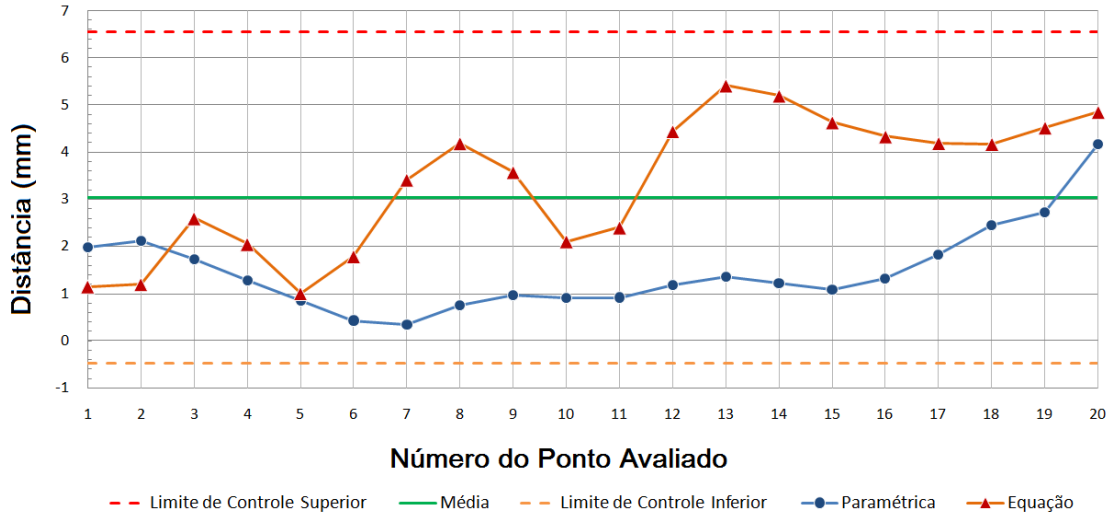
Figura 39. Análise por Vetores da Curva Equação.



Análise entre a “Curvas Referência” e “Curva Equação”, por vetores de distância entre as curvas.

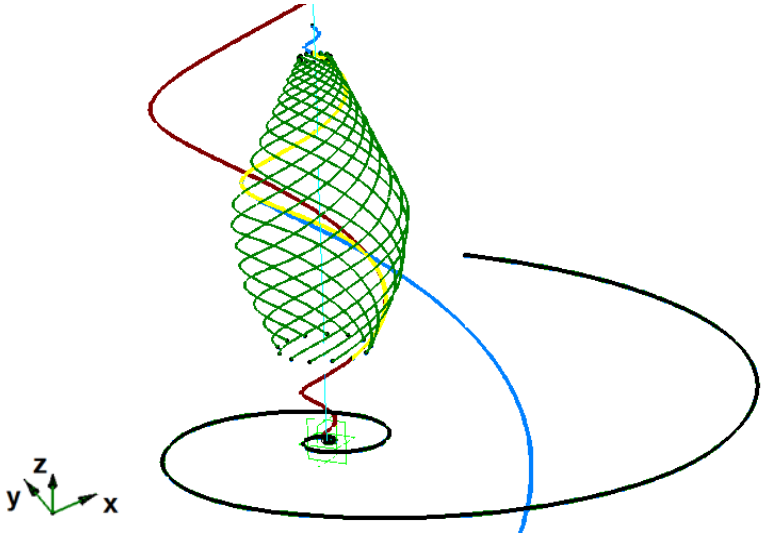
O gráfico da Figura 40 mostra as 20 medidas tomadas para avaliação dos valores das distâncias entre os pontos projetados nas curvas. O valor médio da distância entre as curvas (Paramétrica e Equação) é de 3,04 mm. Para determinar os limites de controle das médias foi considerada a extensão de seis desvios-padrões (três para cada lado da curva) que segundo a distribuição Normal compreende 99,73% dos valores de médias amostrais. Isso resultou no limite de controle superior de 6,56 mm e no limite de controle inferior de -0,48 mm.

Figura 40. Média dos Pontos de Medição 01.



A Figura 41 apresenta o resultado da modelagem da “Curva Paramétrica” copiada e rotacionada em torno do eixo de alinhamento, com o comando “Matriz Circular”, formando uma estrutura triangular semelhante a estrutura do abacaxi.

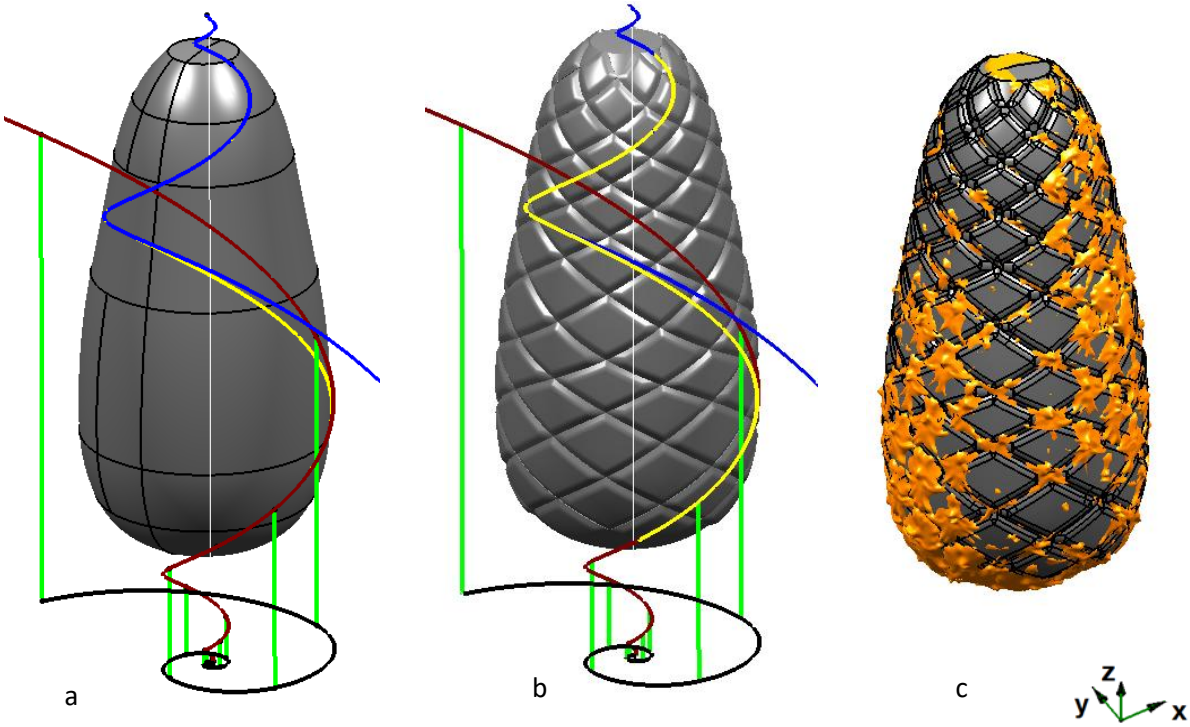
Figura 41. Design Paramétrico das Curvas de Construção.



Curvas de construção copiadas e rotacionadas pelo comando “Matriz Circular”.

O resultado do Modelo 3D Final (Superfícies Paramétricas), apresentado na figura 42 (a) representa a proposta de desenho 3D da engenharia reversa do abacaxi. Para visualização, as áreas formadas pela intersecção das curvas foram extrudadas, conforme observa-se na Figura 42 (b). Este modelo foi ainda sobreposto com a malha do abacaxi para verificação da semelhança, Figura 42 (c).

Figura 42. Modelo 3D Final (Superfícies Paramétricas).



Modelo 3D Final (Superfícies Paramétricas): a) sobreposições das curvas de construção e o Modelo 3D; b) modelo gerado com extrusão das áreas de intersecção das curvas; c) sobreposição do padrão de crescimento e da malha do abacaxi.

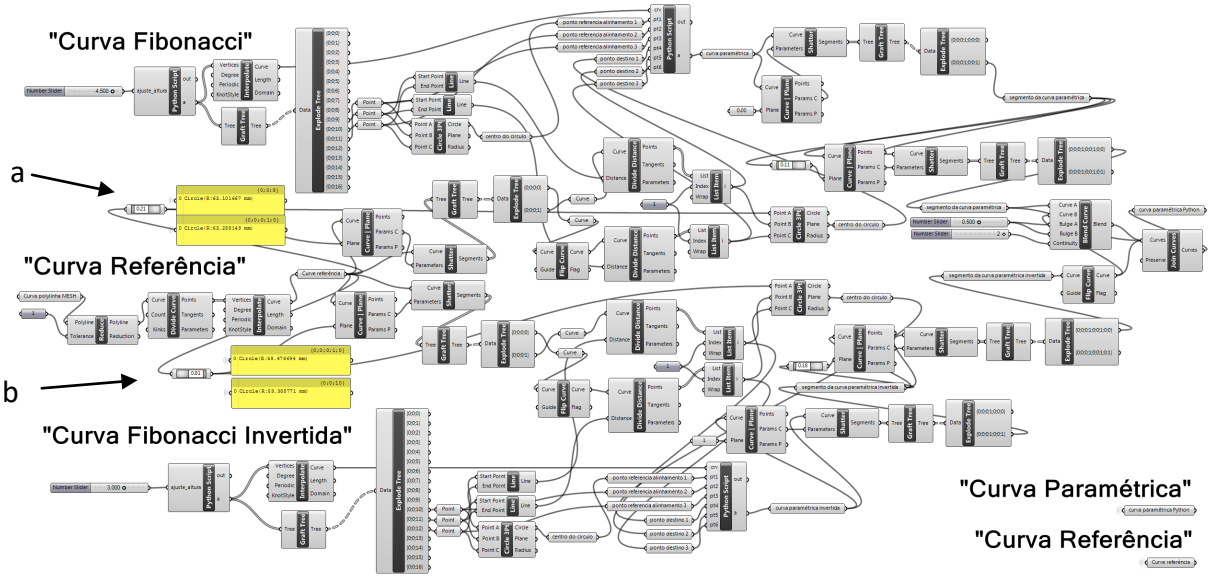
Através da aplicação do método de digitalização tridimensional e parametrização por *software* de CAD foi possível modelar o padrão de crescimento do abacaxi com base na espiral áurea da série de Fibonacci. Assim, foi demonstrada a possibilidade do uso da engenharia reversa como complemento ao método da Biônica para auxiliar a geração de alternativas de desenvolvimento do produto.

4.3 Otimização da Programação

A seguir, buscou-se sistematizar o fluxo de desenvolvimento para a modelagem tridimensional, de maneira que seja possível replicar as etapas de

Engenharia Reversa e Design Paramétrico para outros elementos da natureza com o padrão de crescimento espiral. Buscou-se automatizar o processo ao máximo, para que haja a menor intervenção manual possível. Utilizando o recurso da programação no *Grasshopper,* Figura 43*,* algumas etapas foram otimizadas, exemplo da seleção de uma “Curva Polilinha” que através da função de suavização modela uma “Curva Referência”. Através da função de programação por Python Script foi possível construir a “Curva *Fibonacci*”, e também separar os pontos, projetados no eixo Z, que definem o círculo referência para o alinhamento destas curvas. A ampliação da imagem encontra-se no Apêndice A (página 101).

Figura 43. Modelagem Paramétrica Otimizada.

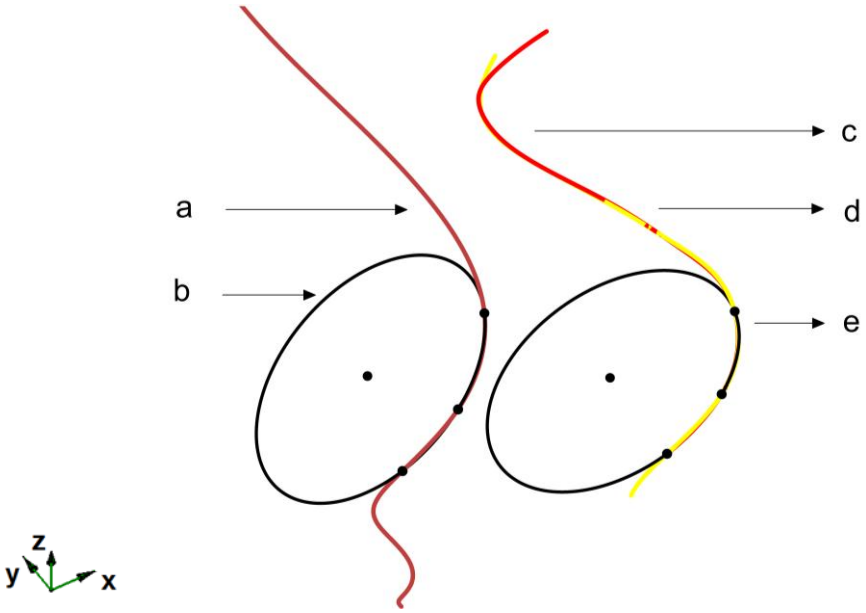


Programação otimizada no *Grasshopper*: a) cursor para parametrização do círculo de alinhamento, possição do ponto tangente entre a “Curva Referência” e a “Curva Fibonacci”; b) cursor para parametrização do círculo de alinhamento superior, posição do ponto tangente entre a “Curva Referência” e a “Curva Fibonacci Invertida”.

O critério chave para modelagem paramétrica é a movimentação do cursor que determina a posição dos pontos sobre as curvas. Esse é o critério utilizado para definir o raio do círculo projetado sobre a “Curva referência” e que deverá ter o mesmo raio do círculo projetado sobre a “Curva Fibonacci”. No momento em que os dois raios ficam iguais, através de nova programação por scripts em Py*thon*, é possível fazer o alinhamento (mover e rotacionar) orientando os objetos por três pontos de referência e três pontos de destino (*target)*, Figura 44. O recurso deverá ser repetido para a construção de uma “Curva Fibonacci Invertida” que vai compor a parte superior da “Curva referência”. Para finalizar a construção da “Curva

Paramétrica” será necessário cortar o excesso das curvas e construir uma “Curva Tangente” que vai ligar a “Curva Fibonacci” e a “Curva Fibonacci Invertida”.

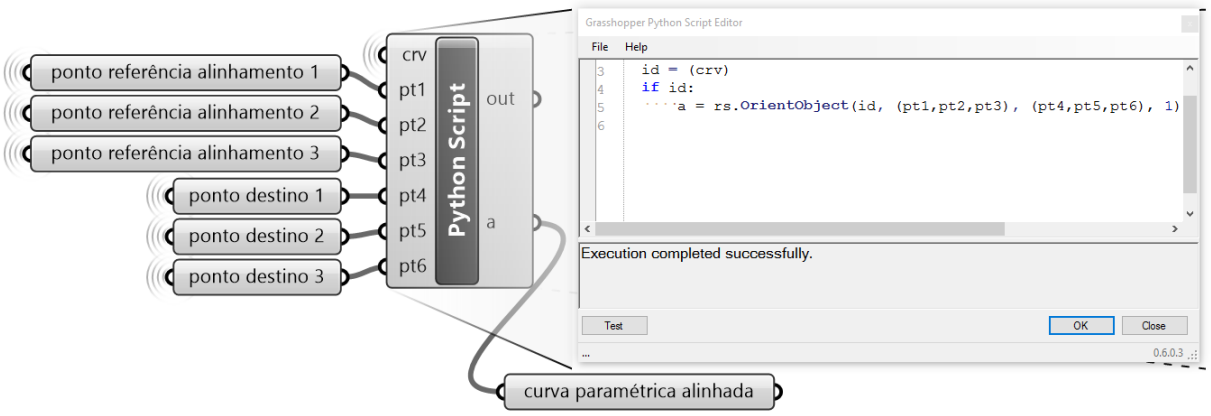
Figura 44. Alinhamento por Círculos com *Script Python*.



Alinhamento das curvas por script: a) “Curva Fibonacci”; b) círculo e pontos para referência; c) “Curva Referência” (amarelo); d) “Curva Paramétrica” (vermelha); e) círculo e pontos para destino.

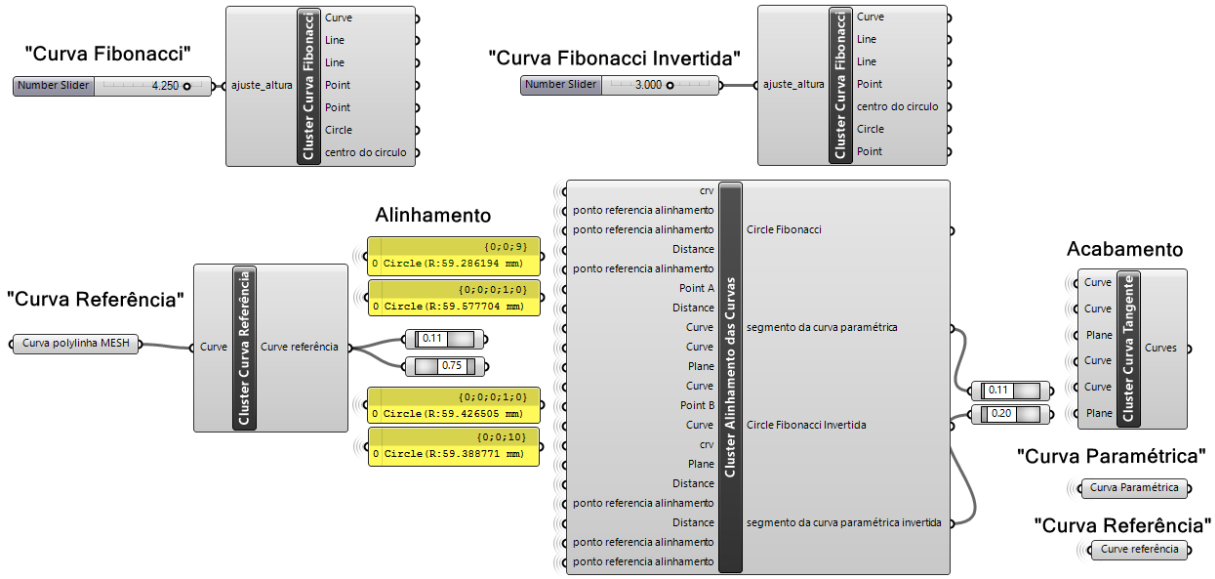
Através da programação por *Grasshopper Python Script Editor* (Figura 45) é possível realizar o alinhamento da “Curva Fibonacci” com a “Curva Referência”. A operação consiste na seleção do objeto, dos três pontos de referência e dos três pontos de destino. Automaticamente, o objeto selecionado vai transladar e rotacionar posicionando-se de maneira que os pontos de referência fiquem sobrepostos aos pontos de destino.

Figura 45. Programação para Alinhamento das Curvas.



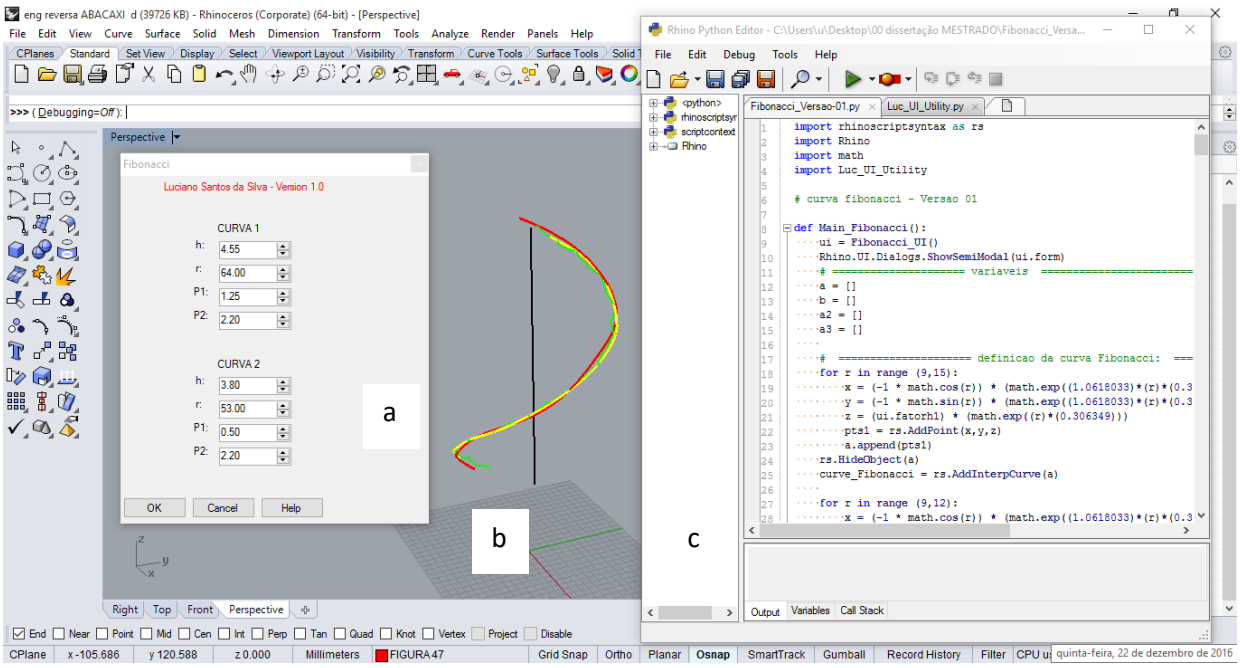
Visando facilitar a interação do usuário com a modelagem via “Design Paramétrico Otimizado”, a opção da função *Cluster* do *Grasshopper* possibilita agrupar as funções em caixas editáveis. Esta caixa mantém as opções de rotulação e ligação por fios das funções de modelagem, além de acrescer a possibilidade de ser exportada e protegida por senha. A Figura 46 apresenta a organização das funções em *Clusters* (“Curva Fibonacci”, “Curva Fibonacci Invertida”, “Curva Referência”, Alinhamento das Curvas e a “Curva Tangente”).

Figura 46*. Grasshopper* Função *Cluster.*



Conforme apresentado na Figura 47, o fluxo de desenvolvimento realizado no *Grasshopper* foi analisado e estruturado por Scripts no *Rhino Python Editor*. Através da Modelagem via Programação *Scripts* foi possível construir a “Curva Equação” substituindo a “Curva Paramétrica”. No *Rhinoceros* opção Tools/ PythonScript/Run ou Edit, foi compilado o arquivo “Fibonacci_Versao-01.py”, o resultado é a abertura de uma janela de interação com o designer. Através da janela é possível escolher os valores para construção das curvas, selecionar a “Curva Referência” e o eixo. A modelagem da “Curva Fibonacci”, esferas, círculos e pontos para alinhamento são realizados automaticamente e o resultado é a “Curva Equação” posicionada sobre a “Curva Referência”. O recurso pode ser repetido para outros elementos da natureza bastando ajustar os parâmetros na janela de interação. O detalhamento do *Script* encontra-se no Apêndice B (página 103).

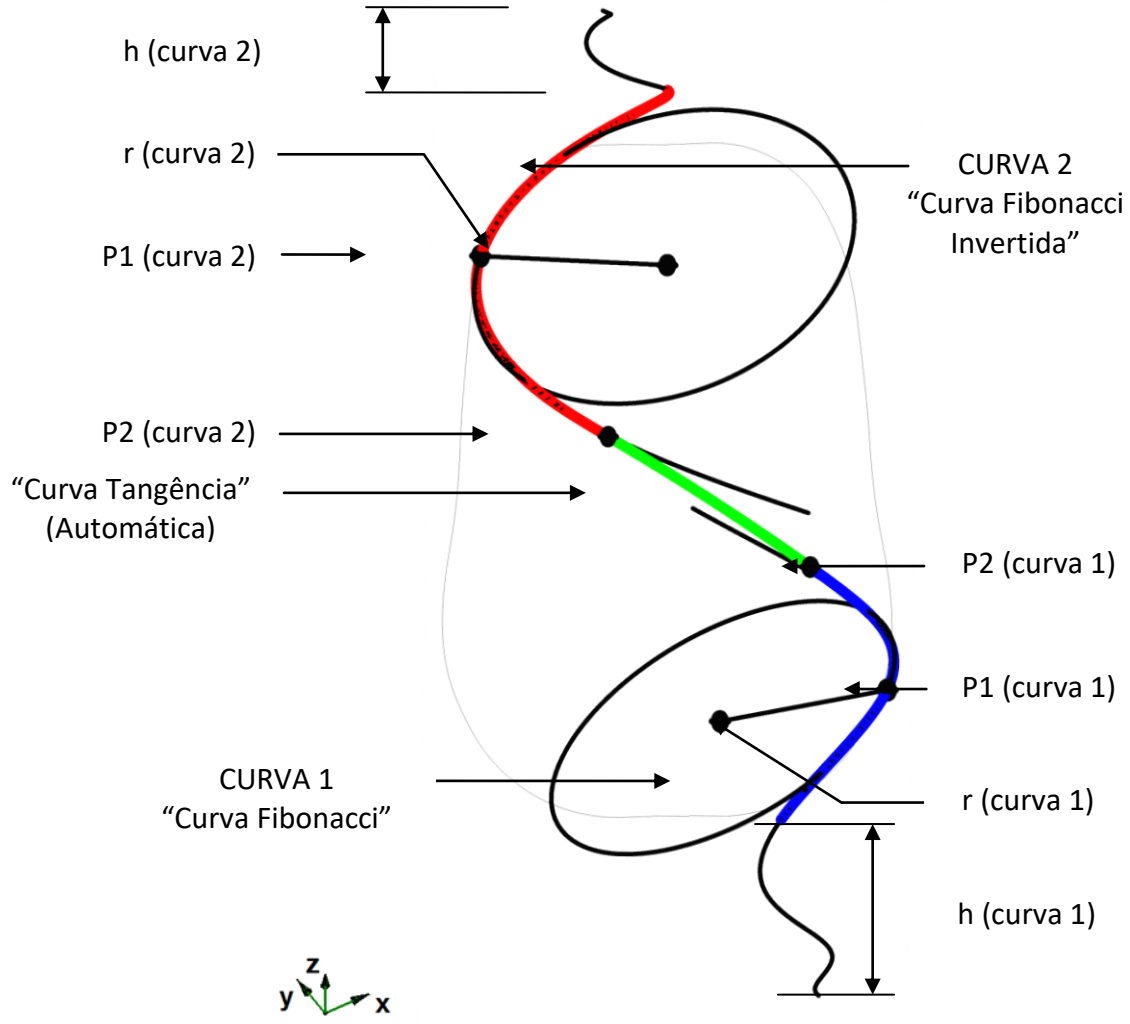
Figura 47. Interface e programação no *Rhino Python Editor.*



Programação em *Rhino Python Script* da “Curva Equação”: a) Janela de interação que permite a visualização gráfica dos parâmetros para construção da “Curva Equação”; b) Janela do *software Rhinoceros*; c) Janela com os *scripts* da programação paramétrica no *Rhino Python Editor.*

Para a construção da “Curva Equação” é necessário o ajuste de alguns parâmetros, os quais são ilustrados na Figura 48. Os parâmetros apresentados são para construção da “Curva Fibonacci” (curva 1), da “Curva Fibonacci Invertida” (curva 2) e para a “Curva Tangência” (ajustada automaticamente pelo *script*). No caso das curvas 1 e 2 o parâmetro “h” corresponde ao Fator da Altura (Ajuste da Altura da Espiral Áurea). O parâmetro “P1” corresponde ao ponto de tangência do círculo de alinhamento com a “Curva Equação” e o “P2” corresponde ao ponto final da curva. O parâmetro “r” corresponde ao valor do raio para construção do círculo e dos pontos de referência para alinhamento da “Curva Equação” (1) ou “Curva Equação Invertida” (2) com a “Curva Referência”. Conforme estabelecido na programação, a “Curva Tangência” será construída automaticamente interpolada de modo contínuo e tangente, ligando a “Curva Fibonacci” e a “Curva Fibonacci Invertida” passando pelos pontos “P2” (curva 1) e “P2” (curva 2).

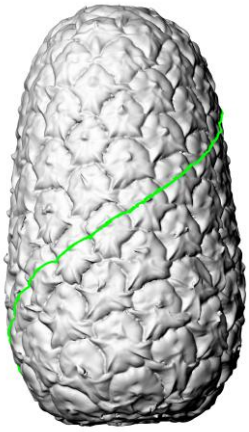
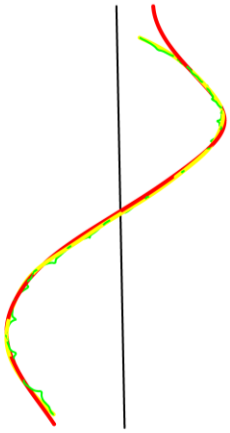
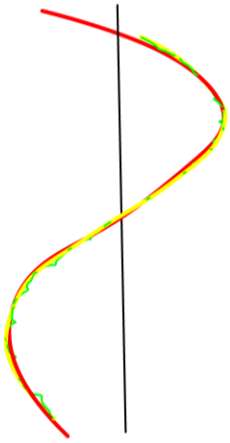
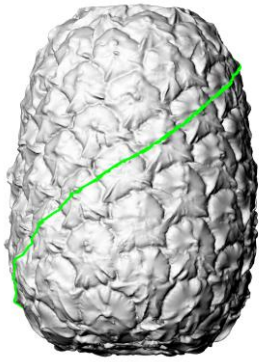
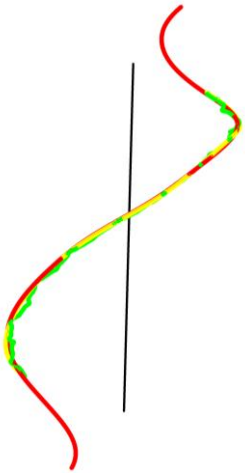
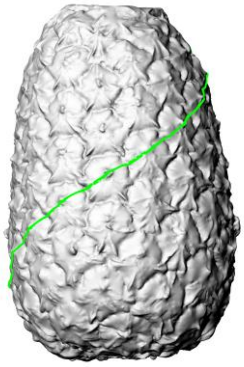
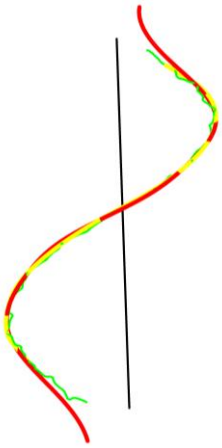
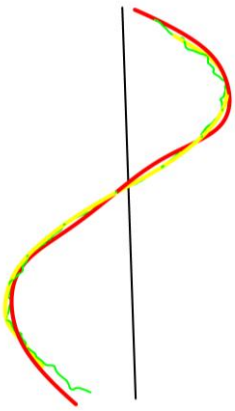
Figura 48. Parâmetros para Programação da "Curva Equação"



4.4 Análise da Modelagem Proposta

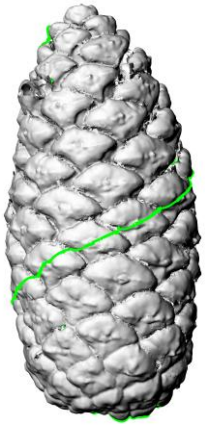
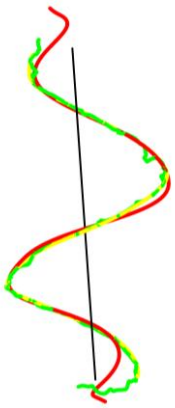
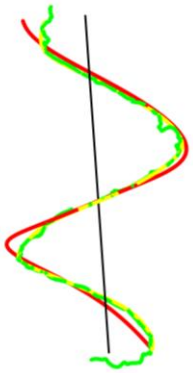
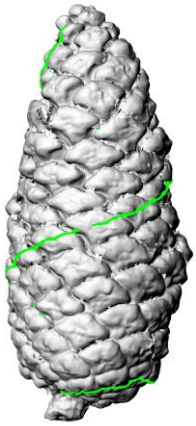
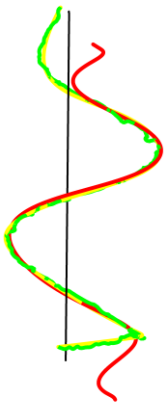
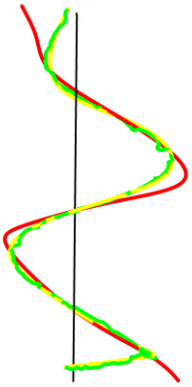
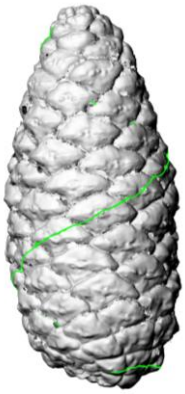
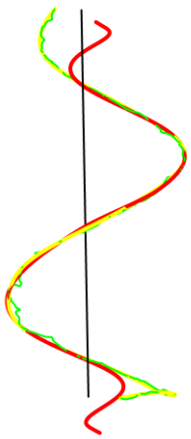
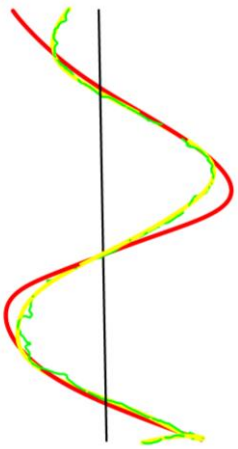
Visando validar a programação, foram realizadas novas digitalizações e modelagens de três modelos de abacaxi e três modelos de pinha, conforme descrito na metodologia proposta. Os procedimentos foram repetidos pelo método da “Modelagem via Design Paramétrico” e da “Modelagem via Programação Scripts”. A Figura 49 e a Figura 50 apresentam os estudos realizados respectivamente para os Abacaxis e para as Pinhas. Na coluna *Grasshopper*, observa-se o resultado da “Curva Referência” e da “Curva Paramétrica” modeladas pela programação otimizada em *Grasshopper*. Na coluna *Python*, observa-se o resultado da “Curva Referência” e da “Curva Equação” modelada pela opção do *Rhino Python Editor*.

Figura 49. Estudo de Caso: Abacaxis.

“Curva Polilinha” e Modelo 3D Digitalizado	“Curva Referência” e “Curva Paramétrica” modelada no *Grasshopper*	“Curva Referência” e “Curva Equação” modelada por Scripts em Python
 Abacaxi_a		
 Abacaxi_b		
 Abacaxi_c		

Verde: Curva Polilinha; Amarelo: Curva Referência; Vermelho: Curvas geradas por parametrização.

Figura 50. Estudo de Caso: Pinhas.

“Curva Polilinha” e Modelo 3D Digitalizado	“Curva Referência” e “Curva Paramétrica” modelada no *Grasshopper*	“Curva Referência” e “Curva Equação” modelada por Scripts em Python
 Pinha_a		
 Pinha_b		
 Pinha_c		

Verde: Curva Polilinha; Amarelo: Curva Referência; Vermelho: Curvas geradas por parametrização.

Para modelagem das curvas de cada elemento natural foi necessário fazer o ajuste dos valores dos raios dos círculos de alinhamento (r) e modificar o parâmetro ajustar altura (h), para a “Curva Fibonacci” e para a “curva Fibonacci Invertida”. Os valores utilizados são apresentados na Tabela 2.

Tabela 2. Parâmetros para Construção das Curvas.

Modelo	Parâmetro Ajustar Altura	Parâmetro Raio
Abacaxi_a	h (1) = 4,71 h (2) = 3,78	r (1) = 66,50 r (2) = 52,77
Abacaxi_b	h (1) = 4,00 h (2) = 3,50	r (1) = 55,50 r (2) = 49,50
Abacaxi_c	h (1) = 4,55 h (2) = 3,80	r (1) = 64,00 r (2) = 53,00
Pinha_a	h (1) = 2,45 h (2) = 2,70	r (1) = 20,35 r (2) = 21,65
Pinha_b	h (1) = 2,50 h (2) = 2,07	r (1) = 20,65 r (2) = 18,70
Pinha_c	h (1) = 2,00 h (2) = 2,00	r (1) = 18,45 r (2) = 18,45

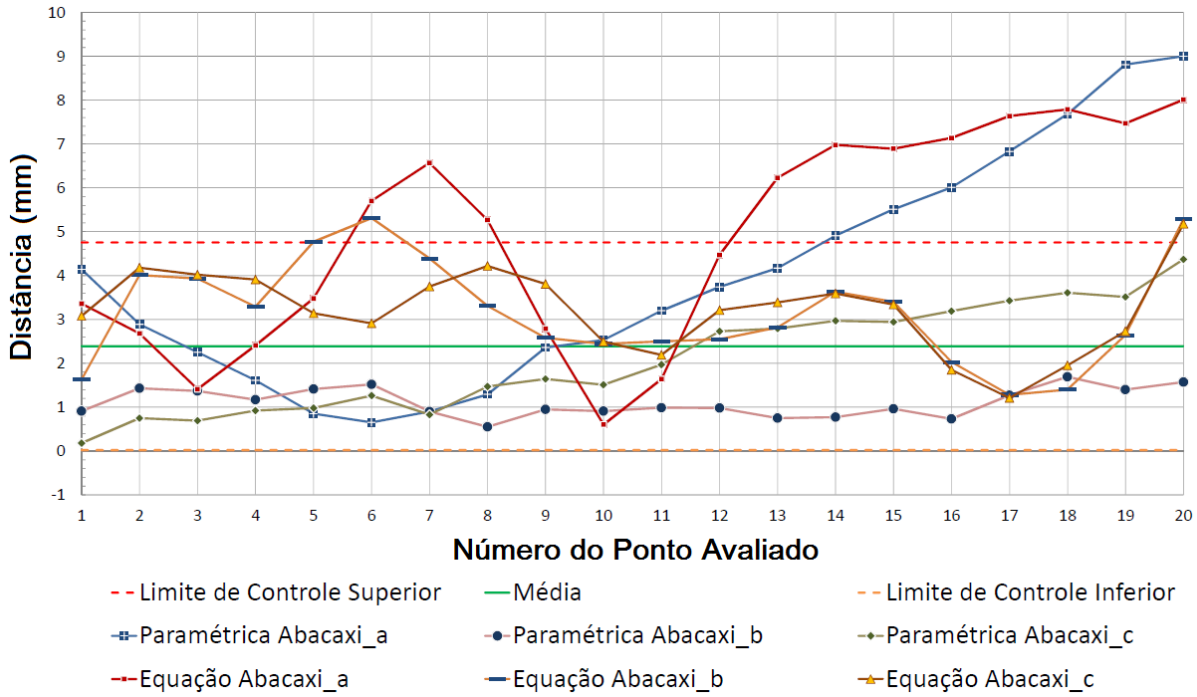
No caso dos abacaxis, foi observada a aproximação dos parâmetros de construção das curvas, em que os valores para os diferentes modelos variou para h(1) de 4,00 a 4,71 mm e para h(2) 3,50 a 3,80 mm. No modelo “Abacaxi_b”, que tem a forma mais arredondada, os parâmetros de ajuste h(1) e h(2) ficaram mais próximos que para os demais modelos (“Abacaxi_a” e “Abacaxi_c”) que apresentam uma forma mais alongada e cônica.

No caso dos modelos das Pinhas, a melhor região para aproximação da “Curva Paramétrica” é a região central, apresentando uma melhor definição da espiral Áurea em relação às extremidades. Foi observado que devido às severas deformações do modelo, para cada “Curva Referência” escolhida uma diferente “Curva Paramétrica” será modelada. Constatou-se que os parâmetros de raio aumentam ou diminuem na mesma proporção dos parâmetros de altura.

Foi realizada uma nova análise das curvas por vetores, a exemplo da já apresentada nas Figuras 38, 39 e 40 (Pág. 62). O gráfico da Figura 51 mostra as 20 medidas tomadas para avaliação das distâncias entre os pontos projetados nas curvas dos modelos dos Abacaxis. Foi possível verificar a exatidão e precisão das curvas geradas. O gráfico apresenta o resultado entre a “Curva Paramétrica”

(modelada na versão otimizada do *Grasshopper*) e a “Curva Referência”. Neste gráfico também é apresentado o resultado entre a “Curva Equação” (modelada no *Rhino Python Editor*) e a “Curva Referência”. O valor médio da distância entre as curvas (Paramétrica e Referência) dos três modelos ficou em 2,39 mm. Para determinar os limites de controle das médias foi considerada a extensão de seis desvios-padrões (três para cada lado da curva) que segundo a distribuição Normal compreende 99,73% dos valores de médias amostrais. Isso resultou no limite de controle superior de 4,76 mm e no limite de controle inferior de 0,02 mm.

Figura 51. Média dos Pontos de Medição dos Abacaxis.

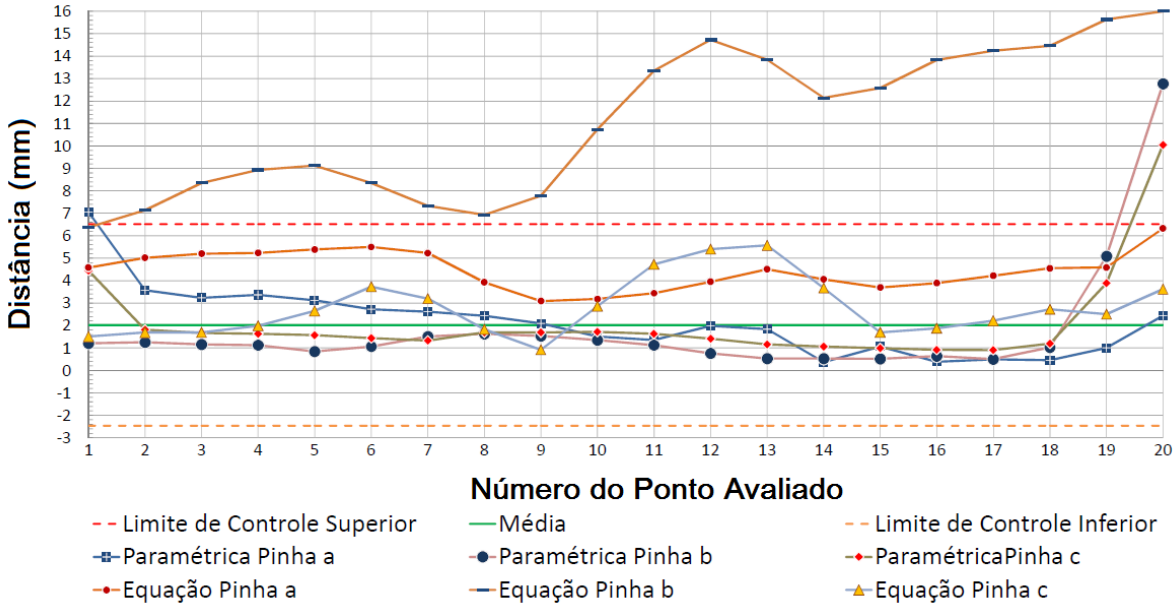


Interpretando o gráfico é possível identificar que as curvas “Paramétrica Abacaxi_b” e “Paramétrica Abacaxi_c” ficaram com os pontos de avaliação dentro dos limites de controle. Estas curvas modeladas na versão “Otimizada do *Grasshopper*” apresentaram o melhor resultado. A razão da qualidade está no ajuste dos parâmetros durante a modelagem das curvas, possível pela interação visual do designer com a área gráfica do *Rhinoceros.* A curva “Paramétrica Abacaxi_a” apresentou variação dimensional com vários pontos fora dos limites, a avaliação feita foi de que a forma acentuada de cone do abacaxi (provocada por questões naturais) não proporciona o melhor alinhamento da “curva de Fibonacci” na extremidade.

Para a “Modelagem via Programação Scripts”, apesar de ser mais prática e com parâmetros editáveis na janela de interação, o método apresentou as curvas “Equação Abacaxi_b” e “Equação Abacaxi_c” com variações praticamente dentro dos limites de controle. Acredita-se que através da execução de recursos de programação, seria possível a obtenção de curvas mais exatas. Para isso, seria necessária a programação de cálculos iterativos para convergir à exatidão máxima, ou a programação de funções gráficas para que seja possível modificar os parâmetros de construção e em tempo real visualizar as alterações das curvas.

O gráfico da Figura 52 mostra as 20 medidas tomadas para avaliação das distâncias entre os pontos projetados nas curvas dos modelos das Pinhas. O gráfico apresenta o resultado entre a “Curva Paramétrica” (modelada na versão otimizada do *Grasshopper*) e a “Curva Referência”. Neste gráfico também é apresentado o resultado entre a “Curva Equação” (modelada no *Rhino Python Editor*) e da “Curva Referência”. O valor médio da distância entre as curvas (Paramétrica e Referência) dos três modelos ficou em 2,03 mm. Para determinar os limites de controle das médias foi considerada novamente a extensão de seis desvios-padrões, o que resultou no limite de controle superior de 6,52 mm e no inferior de -2,46 mm.

Figura 52. Média dos Pontos de Medição das Pinhas.

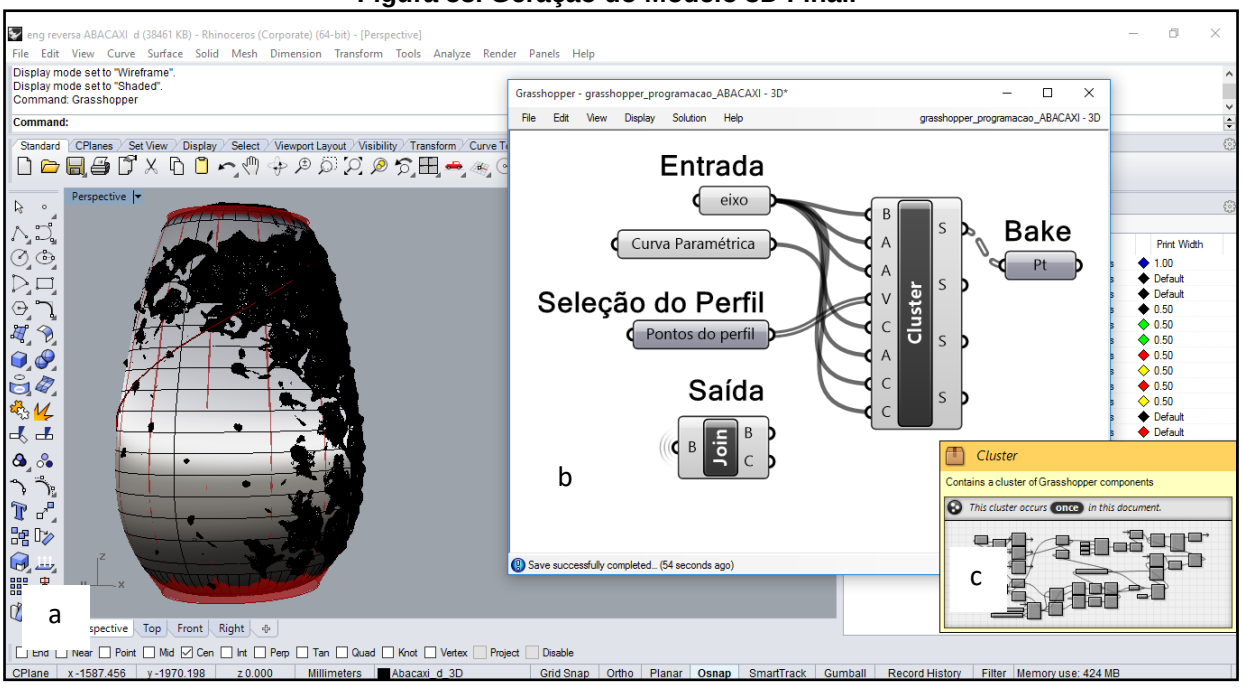


No caso da modelagem das Pinhas, praticamente os dois métodos (“Modelagem via Design Paramétrico” e “Modelagem via Programação Scripts”) apresentaram os pontos entre os limites de controle. A excessão foi a curva

“Equação Pinha b”, para a qual foi observada uma grande variação dimensional decorrente da forma natural assimétrica. Destaca-se que as extremidades dos objetos naturais apresentam uma maior irregulariedade, o que também é refletido por um afastamento das curvas nesses pontos. No caso da “Modelagem via Programação Scripts” as mesmas considerações feitas para os recursos de programação dos Abacaxis são feitas para as Pinhas.

Para o desenho do Modelo 3D Final (Superfícies Paramétricas) partiu-se da “Curva Paramétrica” a partir das funções disponíveis no *Grasshopper,* armazenadas dentro de um *cluster* (Figura 53). O primeiro passo foi a seleção do eixo e da “Curva Paramétrica” (informações de entrada). O segundo passo é feito automaticamente pelas funções do *Cluster,* que desenham os círculos, o plano e os pontos de intersecção (saída do *Cluster*). O terceiro passo é o desenho dos pontos na *layer* ativa do *Rhinoceros* através da função *Grasshopper Bake*. O passo final é a seleção dos pontos que vão formar o Perfil para construção da superfície de revolução através do comando *Revolve*. A descrição dos comandos utilizados no *cluster* encontra-se no Apêndice A (página 102).

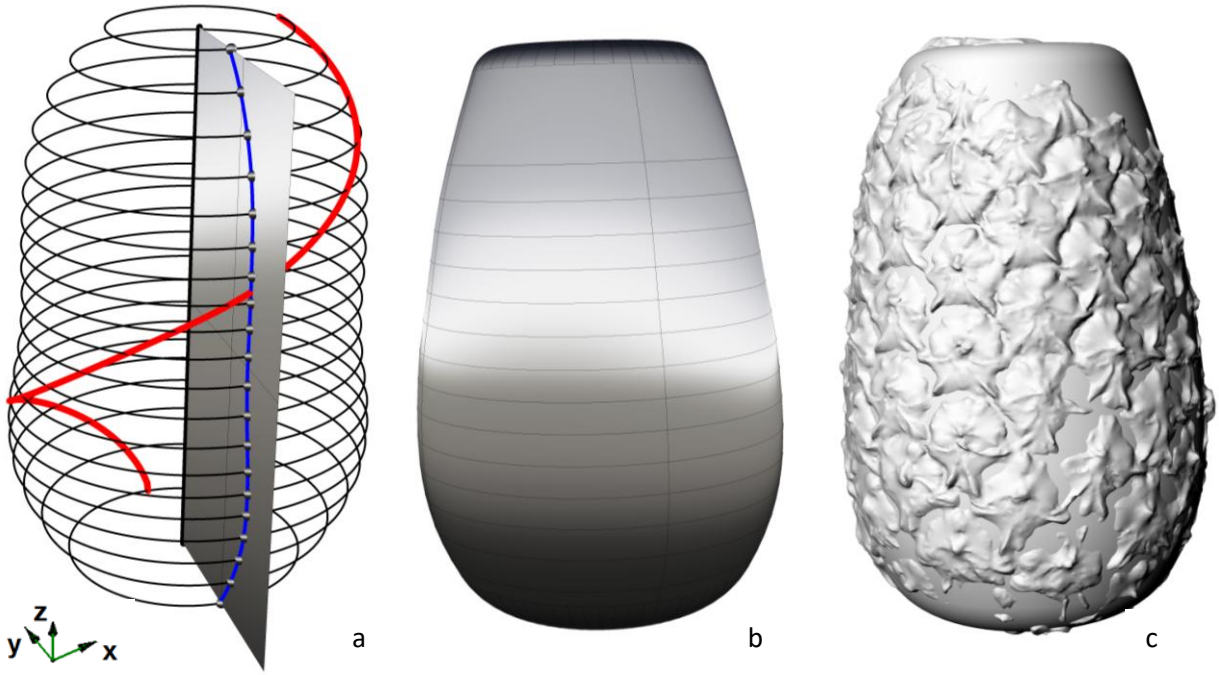
Figura 53. Geração do Modelo 3D Final.



Modelagem 3D: a) *Rhinoceros* resultado com o Modelo 3D Final; b) *Grasshopper* programação das superfícies 3D; c) *Cluster* da programação em *Grasshopper*.

A Figura 54 (a) apresenta a “Curva Paramétrica”, os círculos, o plano de intersecção e os pontos que vão formar o Perfil. A Figura 54 (b) apresenta o resultado do comando de revolução para o Modelo 3D Final que foi construído a partir da seleção do eixo e do Perfil. A Figura 54 (c) apresenta a sobreposição da Malha Digitalizada com o Modelo 3D Final.

Figura 54. Modelo 3D Final (Superfícies paramétricas).



Obtenção do Modelo 3D Final: a) intersecção das curvas de construção; b) superfícies modeladas no *Rhinoceros* e *Grasshopper*; c) sobreposição da malha Digitalizada e das Superfícies Paramétricas.

Para a avaliação tridimensional foi realizada a comparação entre o Modelo 3D Final e o Modelo 3D Digitalizado de cada exemplar estudado. Através da sobreposição dos modelos foi possível avaliar visualmente a diferença entre as secções das Superfícies Paramétricas geradas e da Malha original. Utilizando o software Geomagic Qualify foi possível analisar por Mapa de Cor as diferenças médias, máximas e mínimas entre as superfícies paramétricas e a malha. A Figura 55 apresenta os resultados da avaliação dos abacaxis e a Figura 56 o resultado da avaliação das Pinhas.

Figura 55. Avaliação Tridimensional dos Abacaxis.

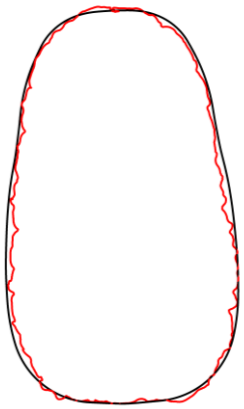
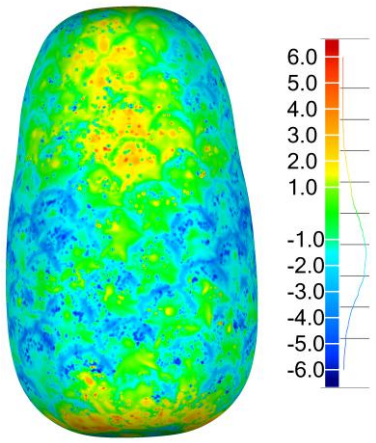
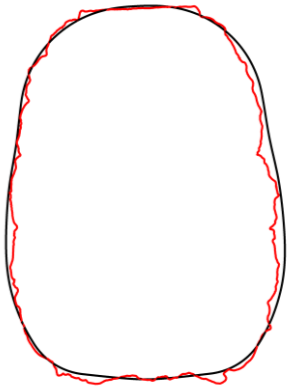
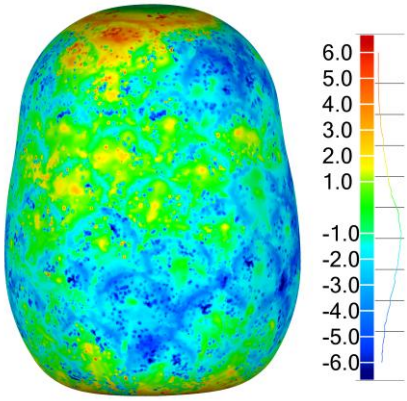
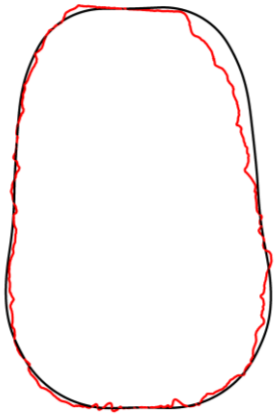
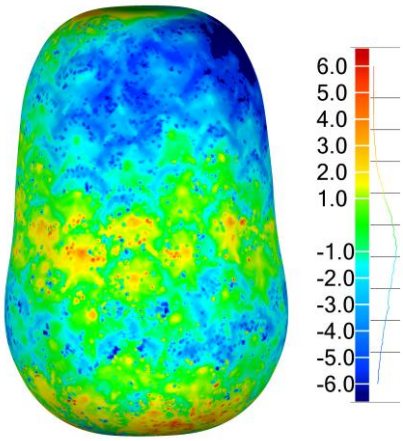
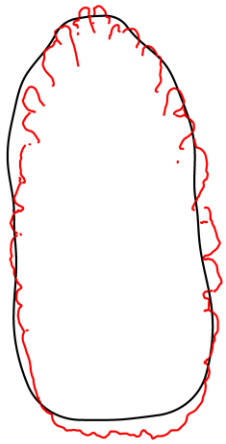
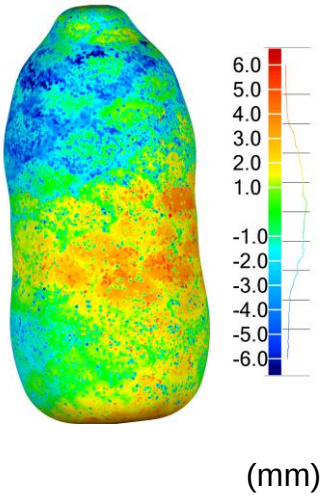
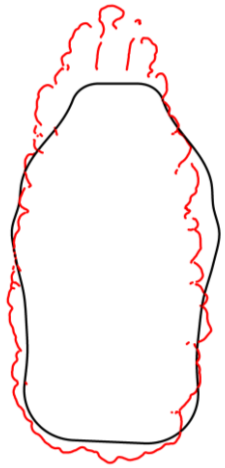
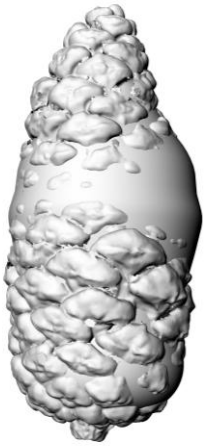
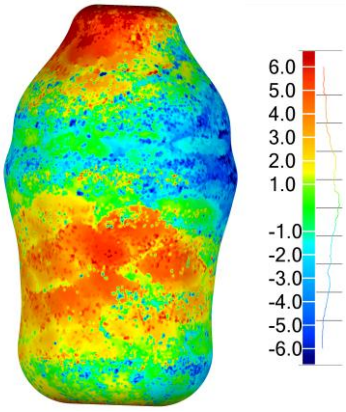
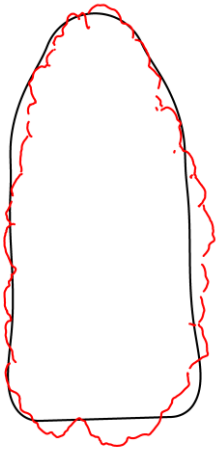
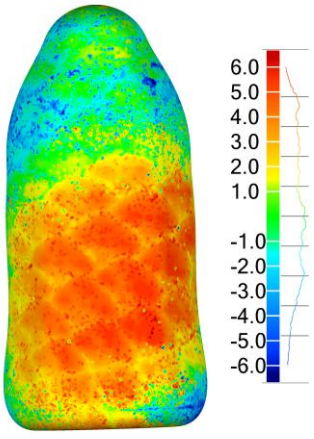
Sobreposição das seções do Modelo 3D Final e do Modelo 3D digitalizado.	Sobreposição do Modelo 3D Final (Superfícies) e do Modelo 3D digitalizado (Malha).	Comparação do Modelo 3D Final com o Modelo 3D digitalizado por Mapa de Cor.
 Abacaxi_a		
 Abacaxi_b		
 Abacaxi_c		

Figura 56. Avaliação Tridimensional das Pinhas.

Sobreposição das seções do Modelo 3D Final e do Modelo 3D digitalizado.	Sobreposição do Modelo 3D Final (Superfícies) e do Modelo 3D digitalizado (Malha).	Comparação do Modelo 3D Final com o Modelo 3D digitalizado por Mapa de Cor.
 Pinha_a		
 Pinha_b		
 Pinha_c		

Os resultados apresentados para os Abacaxis demonstram aproximação das secções entre o Modelo 3D Final e o Modelo 3D Digitalizado de cada exemplar estudado. Para as secções das Pinhas, semelhante ao ocorrido na análise das “Curvas Paramétricas”, o melhor perfil é da “Pinha_a” e a região central apresenta a maior aproximação. Já as extremidades, por serem muito cônicas, apresentaram o perfil muito distorcido, conforme observa-se nos modelos “Pinha_b” e “Pinha_c”.

Através do histograma (nas escalas das Figuras 55 e 56) é observada a distribuição da freqüência em faixas de aproximação entre a malha e as superfícies dos modelos, assim, avalia-se que o processo apresenta alta dispersão das medidas.

A Tabela 3 apresenta o resultado dos valores de Média das diferenças entre as superfícies (dos valores positivos e negativos) e do Desvio-Padrão na comparação realizada no *software Geomagic Qualify*.

Tabela 3. Desvio 3D.

Modelo	Média	Desvio-Padrão
Abacaxi_a	+1,0/-2,2 mm	1,9 mm
Abacaxi_b	+1,5/-2,6 mm	2,6 mm
Abacaxi_c	+1,2/-2,7 mm	2,5 mm
Pinha_a	+1,8/-2,5 mm	2,8 mm
Pinha_b	+3,1/-3,0 mm	3,9 mm
Pinha_c	+2,7/-2,9 mm	3,5 mm

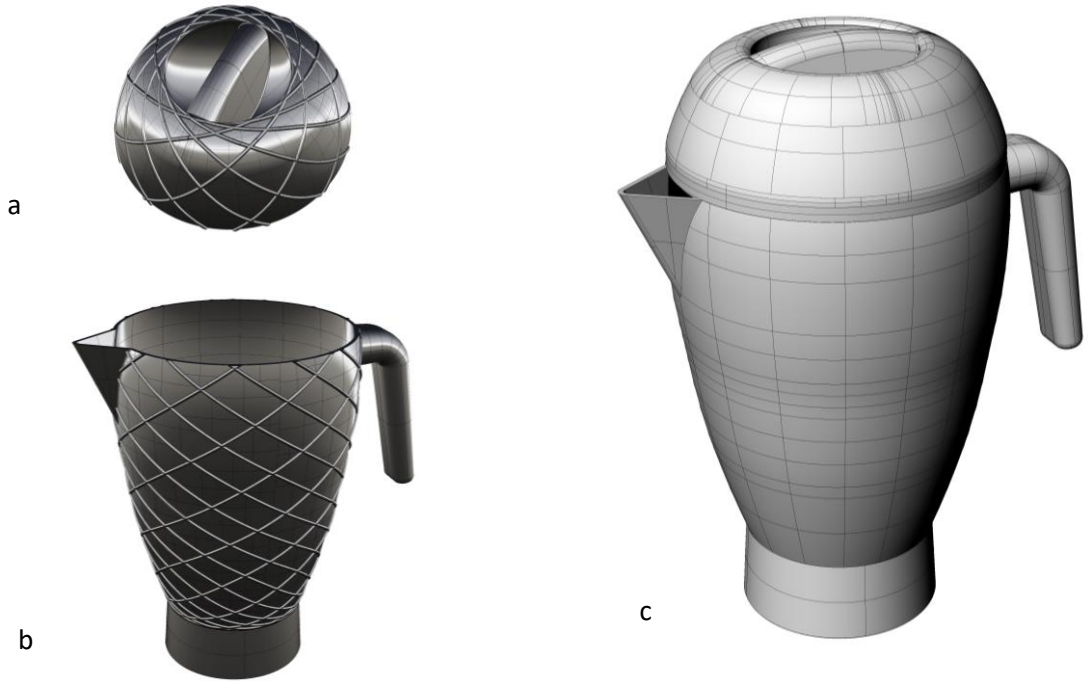
Nota-se que para os Abacaxis os valores das Médias são menores em relação aos valores das Médias das Pinhas. O Desvio-Padrão dos abacaxis ficou menor em relação ao Desvio-Padrão das Pinhas. Assim, para as amostras dos abacaxis o processo apresentou-se mais estável e com maior percentual de aproximação entre a Malha do Modelo 3D Digitalizado e o Modelo 3D Final. Já as Pinhas, por possuírem geometrias mais deformadas, apresentaram erros maiores nos modelos 3D Finais.

Grande parte dos erros pode ser atribuída à assimetria de um objeto da natureza (que não é geometricamente perfeito) ao ser parametrizado. Ainda, erros são gerados ao desconsiderar a textura superficial dos elementos naturais (em especial os Frutículos do Abacaxi ou as Escamas da Pinha).

4.5 Exemplos de Aplicações

Para exemplificar o potencial de uso dos modelos 3D Finais (Superfícies Paramétricas), foram propostos alguns produtos conceituais, sem a intenção de constituir um projeto de produto. Tais produtos de exemplo foram apenas bioinspirados, ou seja, inspirados nas formas da natureza obtidas durante este trabalho. O primeiro exemplo é um copo e tampa de liquidificador baseado no modelo do Abacaxi_a, conforme resultado apresentado na Figura 57. A modelagem foi realizada a partir das superfícies paramétricas com uso das ferramentas de CAD, onde foi possível a divisão do modelo, definição de espessura, desenho das pegas e do bico direcionador.

Figura 57. Copo e Tampa de Liquidificador Conceitual.



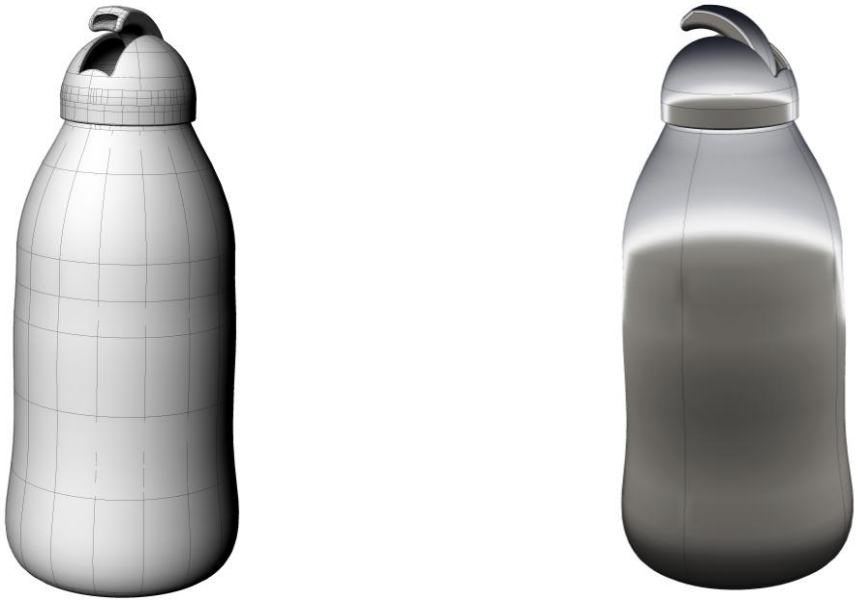
Produto: a)Tampa do liquidificador; b) Copo do liquidificador;
c) Conjunto Copo e Tampa do liquidificador liso.

O exemplo de um espremedor de frutas foi modelado baseado na curva Fibonacci invertida do modelo do Abacaxi_c. Através da modelagem em CAD, foi definido um perfil triangular que percorreu a curva Fibonacci para formar a aresta de contato tipo faca, definindo a função do espremedor. A aresta resultante foi copiada e rotacionada em torno do eixo, assim sendo possível finalizar o modelo através do desenho da base de encaixe universal. O resultado é apresentado na Figura 58.

Figura 58. Espremedor de Frutas Conceitual.

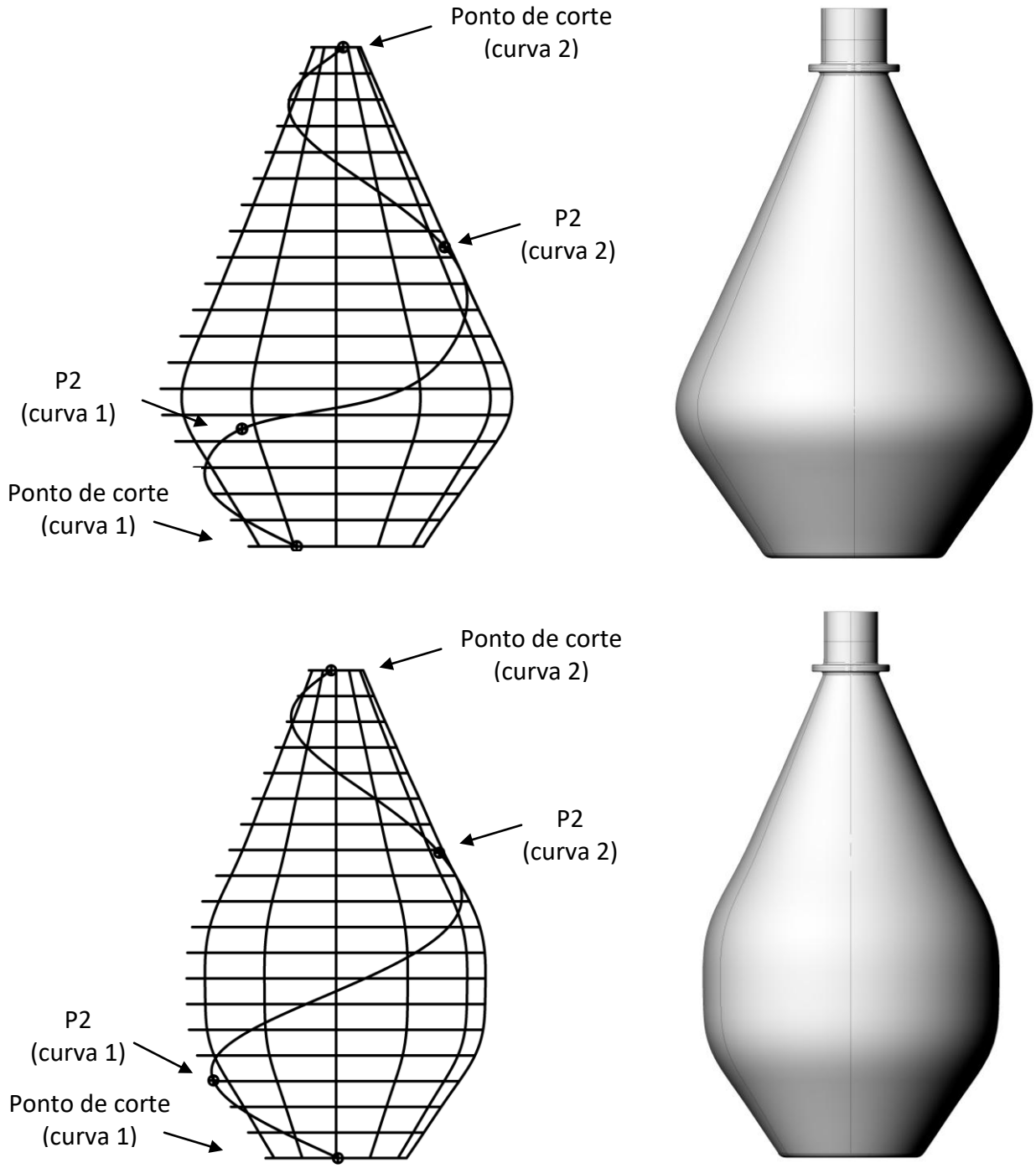
Também foi modelada uma garrafa *squeeze* conceitual baseada no modelo da Pinha_c. Para este modelo, foram utilizadas as superfícies paramétricas e a partir das ferramentas do CAD foi possível desenhar a garrafa e a tampa com bico direcionador, conforme resultado apresentado na Figura 59.

Figura 59. Garrafa *Squeeze* Conceitual.



Por fim, conceitualmente, foi proposta uma garrafa paramétrica de água a partir de uma “Curva Equação”. Utilizando o algoritmo de crescimento espiral de geometrias da natureza desenvolvido neste trabalho, pode-se variar parâmetros para gerar diferentes modelos, conforme observa-se na Figura 60.

Figura 60. Garrafa Paramétrica de Água Conceitual.



Esta modelagem foi orientada no plano cartesiano e com eixo normal ao plano XY. Este exemplo foi baseado no conceito da construção da “Curva Equação” e do ajuste de alguns parâmetros para construção da “Curva Fibonacci” (curva 1), da “Curva Fibonacci Invertida” (curva 2) e para a “Curva Tangência”. Para o “Ponto de corte” (curvas 1 e 2) foram definidos os parâmetros das distâncias entre os pontos e a origem do desenho, limitando assim as curvas Fibonacci. Os pontos “P2” (curvas 1 e 2) correspondem aos pontos de final de cada curva, movimentando estes pontos é possível definir o volume e o perfil da garrafa. Conforme estabelecido na programação, a “Curva Tangência” será construída automaticamente interpolada de modo contínuo e tangente, ligando a “Curva Fibonacci” e a “Curva Fibonacci Invertida” passando pelos pontos “P2” (curva 1) e “P2” (curva 2). Isso resulta em diferentes alternativas de modelos, conforme apresentado na Figura 60. Assim, demonstra-se que o *script* desenvolvido neste trabalho pode ser utilizado para a geração de alternativas e de modelos conceituais, os quais podem auxiliar nas etapas iniciais de criação em um projeto de design.

5. CONCLUSÃO

Este trabalho basicamente teve o objetivo de selecionar e modelar de modo paramétrico formas da natureza representativas do padrão de crescimento espiral, aplicando as tecnologias CAD (Rhinoceros e Grasshopper) a partir da digitalização 3D. Com isso, foi possível a sistematização do fluxo de desenvolvimento para modelagem tridimensional utilizando recursos do design paramétrico e da modelagem via programação por scripts.

Levando-se em conta o desenvolvimento da equação matemática para modelagem da espiral logarítmica, a partir da variável h (altura), foi possível representar graficamente a sequência de Fibonacci em forma de uma espiral “tridimensional”. Observado que a forma estabelecida pela curva espiral é próxima à curva identificada no Modelo da Digitalização (Malha), conclui-se que partindo do modelo digitalizado e da programação paramétrica é possível representar por software CAD as geometrias semelhantes às observadas nos elemento da natureza.

Por intermédio da interação do software Rhinoceros com o Grasshopper é possível aplicar o conceito do Design Paramétrico, que parte da base do modelo com parâmetros de controle das geometrias, gerando diversas possibilidades de projeto e por meio de um conjunto de regras e algoritmos planejados a modelagem dessas alternativas. A otimização da parametrização foi possível pelo desenvolvimento do algoritmo com funções em Python, que contribuíram para o aperfeiçoamento da modelagem. Um novo algoritmo, desenvolvido pelos scripts via Rhino Python Editor, resultou na modelagem de geometrias mais eficientes respeitando as restrições do projeto, modelando as curvas de forma iterativa e intantânea.

Pela observação dos aspectos analisados por Pente de Curvas e por vetores, os resultados dos abacaxis apresentaram similaridade entre as curvas “Referência” e “Paramétrica”. Através da análise das superfícies por Mapa de Cor foi possível identificar a aproximação das superfícies e do perfil entre o Modelo 3D digitalizado (Malha) e o Modelo 3D Final (Superfícies Paramétricas), concluindo-se

que a melhor modelagem foi da realizada pela parametrização otimizada com o algoritmo desenvolvido no Grasshopper com funções em Python. Foi observada nas Pinhas uma grande variação dimensional decorrente da forma natural assimétrica. Destaca-se que as extremidades dos objetos naturais apresentam uma maior irregularidade, o que também é refletido por um afastamento das curvas nesses pontos. Isso influencia diretamente na escolha da curva para definição do desenho do Modelo 3D Final.

Os exemplos de aplicação apresentados neste trabalho indicam uma maneira de auxiliar a concepção do projeto, inspirada nas formas complexas da natureza, para atender demandas por inovação e criatividade. Assim, o design paramétrico, auxiliado por softwares de programação, pode influenciar diretamente no desenvolvimento dos produtos.

Considerando que a sequência de Fibonacci é aplicada para explicar as principais características geométricas dos elementos naturais, recursos de modelagem 3D podem ser utilizados para geração de alternativas de formas no desenvolvimento de produtos. Aplicações poderão ser estudadas a exemplo dos arranjos das folhas no ramo de árvores (formação que facilita o escoamento da água), na espiral formada pela folha de uma bromélia, sementes de girassol que são organizadas em duas espirais, nas sementes das pinhas, no arranjo do cone da alcachofra, no desenrolar da samambaia ou na Concha do Nautillus. Neste sentido, ao facilitar a modelagem de superfícies paramétricas, a partir da digitalização 3D de elementos da natureza, contribui-se com a modelagem de curvas e superfícies para utilização em projetos de design.

5.1 Sugestões para Trabalhos Futuros

- Desenvolver uma metodologia, contemplando desde a aquisição da forma natural via digitalização 3D até a geração de um modelo paramétrico, para utilização como ferramenta de desenvolvimento na fase inicial de concepção do projeto de produtos baseados na Biônica.
- Desenvolver um sistema paramétrico que permita a geração automática de elementos naturais da paisagem urbana.
- Uso da equação da espiral “Phi” como equação para representação dos padrões de repetição “fractais” por meios computacionais. Equações para representar sistemas dinâmicos usadas para descrever fenômenos que, parecem aleatórios, mas obedecem a regras de dimensão fracionária.
- Automatizar e programar o algoritmo em outras linguagem de scripting para utilização em outros softwares CAD, exemplo do C#, VisualBasic.NET, além da utilizada Python.
- Utilizar a modelagem dos elementos da natureza na atividade de projeto no ensino, bem como contribuir para demonstrar a interdisciplinaridade da atividade de projetar com o conceito da Biônica.
- A partir da modelagem proposta, simular por elementos finitos os padrões encontrados na natureza com vistas ao entendimento das funções que podem auxiliar projetos de Biônica.
- Modelar parametricamente outras formas da natureza de grande complexidade.

REFERÊNCIAS

ADAMCZYK, Zbigniew; KOCIOLEK, Krzysztof. **CAD/CAM technological environment creation as an interactive application on the Web**. Journal of Materials Processing Technology, v. 109, n. 3, p. 222-228, 2001.

ADSUL, Bharat; MACHCHHAR, Jinesh; SOHONI, Milind. **Local and global analysis of parametric solid sweeps**. Computer Aided Geometric Design, v. 31, n. 6, p. 294-316, 2014.

ALCÁZAR, Juan Gerardo; HERMOSO, Carlos; MUNTINGH, Georg. **Symmetry detection of rational space curves from their curvature and torsion**. Computer Aided Geometric Design, v. 33, p. 51-65, 2015.

ALLGAYER, Rodrigo. **Formas Naturais e Estruturação de Superfícies Mínimas em Arquitetura.** Dissertação (Mestrado), Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Design, Porto Alegre, BR-RS, 2009.

ANDRADE, Maria das Graças dos Santos et al. **Aspectos da qualidade de infrutescências dos abacaxizeiros 'Pérola'e 'Vitória'.** Agropecuária Técnica, v. 36, n. 1, p. 96-102, 2015.

ARAUJO, Marcos Coutinho Monnerat. **Uso de Técnica de Engenharia Reversa para Reconstrução Tridimensional de Fósseis através de Fotografias.** Dissertação (Mestrado), UFRJ/ COPPE/ Programa de Engenharia Civil, Rio de Janeiro, 2010.

BARBA NETO, João. **Estudo de Elemento da Natureza para Aplicação em Design: Biomimetização da Estrutura de Ninhos de Cacicus Haemorrhous.** Dissertação (Mestrado). Universidade Federal do Paraná, Setor de Artes, Comunicação e Design, Curitiba, BR-PR, 2013.

BECCARI, Carolina Vittoria; NEAMTU, Marian. **On constructing RAGS via homogeneous splines**. Computer Aided Geometric Design, v. 43, p. 109-122, 2016.

BENTLEY, Peter J.; KUMAR, Sanjeev. **Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem.** In: GECCO. 1999. p. 35-43.

BIARD, Luc; FAROUKI, Rida T.; SZAFRAN, Nicolas. **Construction of rational surface patches bounded by lines of curvature**. Computer Aided Geometric Design, v. 27, n. 5, p. 359-371, 2010.

BO, Pengbo; LIU, Yang; TU, Changhe; ZHANG, Caiming; WANG, Wenping. **Surface fitting with cyclide splines**. Computer Aided Geometric Design, v. 43, p. 2-15, 2016.

BROWNE, Cameron. **Gaudí's organic geometry. Computers & graphics**, v. 32, n. 1, p. 105-115, 2008.

CANEPARO, Luca. **Digital fabrication in architecture, engineering and construction**. London: Sprinher, 2014.

CARGNIN, L. G.; ROCKENBACK, M.; SILVA, F. P.; IANNUZZI, R.; KINDLEIN JUNIOR, W. **Desenvolvimento de Geometrias para o Núcleo de Painéis Tipo Sanduíche Através da Digitalização Tridimensional de Conchas de Fósseis de Moluscos Cefalópodes.** In 5º Workshop Design & Materiais: Selação de Materiais e Processos de Fabricação. Lorena: FATEA, v. 1. p. 122-139, 2010.

CELANI, Gabriela; VAZ, Carlos Eduardo Verzola. **Scripts em CAD e Ambientes de Programação Visual para Modelagem Paramétrica: Uma Comparação do Ponto de Vista Pedagógico.** Encontro de Tecnologia de Informação e Comunicação na Construção, v. 5, 2011.

COELHO, Denis A.; VERSOS, Carlos A. M. **An approach to Validation of Technological Industrial Design Concepts With a Bionic Character.** In: Proceedings of the International Conference on Design and Product Development (ICDPD'10). p. 29-31. Covilhã, 2010.

COSTA, Mauro. **Analogias Biológicas en La Arquitectura – Del acercamiento biônico hacia lós paradigmas de ló biodigital.** Tese de Doutoramento. Barcelona : ESARQ - UIC, 2008.

CRUZ, André João Abrunhosa Barata. **Arquitectura [Bio]lógica - Uma Análise da Obra de Frei Otto.** Dissertação de Mestrado Integrado em Arquitectura. Faculdade de Ciências e Tecnologia da Universidade de Coimbra Departamento de Arquitectura, Coimbra, 2012.

DETANICO, F. B.; TEIXEIRA, F. G.; SILVA T. K. A. **Biomimética como Método Criativo para o Projeto de Produto.** Design & Tecnologia – 02 – 2010. Disponível em: <http://www.pgdesign.ufrgs.br/designetecnologia/index.php/det/article/viewFile/52/33>. Acesso em: 13 jun. 2014.

DE AVIZ, Daniel. **Estudo da Técnica de Engenharia Reversa para Construção de Geometrias Complexas Focando Erros de Forma e Métodos de Digitalização Geométrica.** Dissertação (Mestrado). Instituto Superior Tupy, Programa de Mestrado em Engenharia Mecânica, Joinville, BR-SC, 2010.

ESTEVEZ, A. T. **Arquitecturas Genéticas.** Barcelona: ESARQ - Universitat Internacional de Catalunya. 2003.

EKINS, Brian. **Unleashing Hidden Powers of Inventor with the API / Part 1. Getting Started with Inventor VBA – "Hello Inventor!".** Autodesk, Inc. Disponível em: <http://usa.autodesk.com/adsk/servlet/u/gsearch/results?x=10&y=9&siteID=123112&catID=123155&id=2088334&qt=Brian+Ekins+Unleashing+Hidden+Powers+of+Inventor+with+the+API>. Acesso em: 25 ago. 2016.

FLORIO, Wilson. **Modelagem Paramétrica, Criatividade e Projeto: Duas Experiências com Estudantes de Arquitetura.** Gestão & Tecnologia de Projetos, v. 6, n. 2, p. 43-66, 2012.

GALANTUCCI, L. M.; PERCOCOC, G.; DAL MASO, U. A. **Volumetric Approach for STL Generation from 3D Scanned Products.** ELSEVIER, Journal of Materials Processing Technology 204, 2008.

HELBIG, Susanne; HENKEL, Kareen; KRIENER, Jan. **Internet-Projekt fur Theoretische Mathematik.** Spiralen in Naturwissenschaft, Technik und Kunst. Beruflichen, 2001.

KASSABOV, Ognian. **Transition to canonical principal parameters on minimal surfaces**. Computer Aided Geometric Design, v. 31, n. 7, p. 441-450, 2014.

KHABAZI, Zubin. **Generative Algorithms (using Grasshopper)**. Morphogenesism. 2012.

KINDLEIN JÚNIOR, W.; GUANABARA, A. S.; SILVA, E. A. da.; PLATCHECK, E. R. **Proposta de Uma Metodologia para o Desenvolvimento de Produtos Baseados no Estudo da Biônica.** In: ANAIS DO P&D DESIGN 2002 – 1º Congresso Internacional de Pesquisa em Design. 5º Congresso Brasileiro de Pesquisa e Desenvolvimento em Design. Associação de Ensino de Design do Brasil (Textos referentes à sessão técnica Metodologias). Vol. 7. Rio de Janeiro: AEnD-BR, 2002.

KRISH, Sivam. **A practical Generative Design Method.** Computer-Aided Design, v. 43, n. 1, p. 88-100, 2011.

LEE, Jaewook; JUNKIM, Yong; SOOKIM, Myung; ELBER, Gershon. **Efficient Voronoi diagram construction for planar freeform spiral curves**. Computer Aided Geometric Design, v. 43, p. 131-142, 2016.

LOPES, Patrícia. **Equipe Brasil Escola.** Disponível em: <http://www.brasilescola. com/frutas/abacaxi. html>. Acesso em: 02 de dez. 2014.

MACHCHHAR, Jinesh; ELBER, Gershon. **Revisiting the problem of zeros of univariate scalar Béziers**. Computer Aided Geometric Design, v. 43, p. 16-26, 2016.

MAIER, Georg. **Optimal arc spline approximation**. Computer Aided Geometric Design, v. 31, n. 5, p. 211-226, 2014.

MARTINS, Thaís Michelão. **História da Matemática, Sólidos Platônicos, Razões e Proporções.** Instituto de Arquitetura e Urbanismo. USP. São Paulo. 2011.

MEZINI, Ledita; ERYILDIZ, Semih. **Bioarchitecture-Inspirations From Nature.** Gazi University Journal of Science, v. 25, n. 1, p. 263-268, 2012.

MINETOLA, Paolo; LULIANO, Luca; CALIGNANO, F. **A customer oriented methodology for reverse engineering software selection in the computer aided inspection scenario.** Computers in Industry, v. 67, p. 54-71, 2015.

NGUYEN, Thien; PETERS, Jörg. **Refinable C 1 spline elements for irregular quad layout**. Computer aided geometric design, v. 43, p. 123-130, 2016.

OUAMER-ALI, Mohamed-Islem; Laroche, Florent; Bernard, Alain; Remy, Sébastien. **Toward a Methodological Knowledge Based Approach for Partial Automation of Reverse Engineering.** Procedia CIRP, v. 21, p. 270-275, 2014.

RAJA, Vinesh; FERNANDES, Kiran J. **Reverse Engineering: An Industrial Perspective.** Springer Science & Business Media, 2007.

RAMZY, Nelly Shafik. **Biophilic qualities of historical architecture: In quest of the timeless terminologies of 'life'in architectural expression. Sustainable Cities and Society**, v. 15, p. 42-56, 2015.

RIAN, Iasef Md; SASSONE, Mario. Tree**-inspired dendriforms and fractal-like branching structures in architecture: A brief historical overview. Frontiers of Architectural Research**, v. 3, n. 3, p. 298-323, 2014.

SALGUEIRO, Bruno Miguel da Silva. **Projecto e Parametrização - Uma Torre Evolutiva.** Dissertação (Mestrado). Universidade Lusófona de Humanidades e Tecnologias, Escola de Comunicação, Arquitectura, Artes e Tecnologias da Informação, Lisboa, 2011.

SANTOS, Nuno Miguel. **Integração de Biónica em Design do Produto - Modelos de Design Generativo e Paramétrico em Estruturas Efêmeras.** Dissertação Mestrado em Design de Produto. Faculdade de Arquitectura da Universidade de Lisboa. Lisboa, 2014.

SENDRA, J. Rafael; SEVILLA, David. **First steps towards radical parametrization of algebraic surfaces**. Computer Aided Geometric Design, v. 30, n. 4, p. 374-388, 2013.

SILVA, Fabio Pinto da. **O uso da Digitalização Tridimensional a Laser no Desenvolvimento e Caracterização de Texturas Aplicadas ao Design de**

Produtos. Dissertação (Mestrado). Universidade Federal do Rio Grande do Sul, Escola de Engenharia, Programa de Pós-Graduação em Engenharia de Minas, Metalúrgica e de Materiais, Porto Alegre, BR-RS, 2006.

SILVA, Fabio Pinto da. **Usinagem de Espumas de Poliuretano e Digitalização Tridimensional para Fabricação de Assentos Personalizados para Pessoas com Deficiência.** Tese (Doutorado). Universidade Federal do Rio Grande do Sul, Escola de Engenharia, Programa de Pós-Graduação em Engenharia de Minas, Metalúrgica e de Materiais, Porto Alegre, BR-RS, 2011.

TINELLO, Daniel; JODIN, Dirk; WINKLER, Herwig. **Biomimetics Applied to Factory Layout Planning: Fibonacci Based Patterns, Spider Webs and Nautilus Shell as Bio-inspiration to Reduce Internal Transport Costs in Factories.** CIRP Journal of Manufacturing Science and Technology, 2016.

VITTORI, Arturo. **Architecture and Vision. Traces of Centuries & Future Steps.** 13 Mostra Internazionale di Architettura. Biennale di Architettura 2012, Venezia, 2012. Disponível em: <http://www.vittori-lab.com/team/arturo-vittori> e <http://www.warkawater.org/design>, Acesso em: 31 ago. 2015.

WOLF, Vicki. **Biomimicry: solutions hidden in plain sight.** Citizens League for environmental Action Now (CLEAN), 2005. Disponível em: <http://www.cleanhouston. org/living/better_world/biomimicry.htm>. Acesso em 10 fev. 2016.

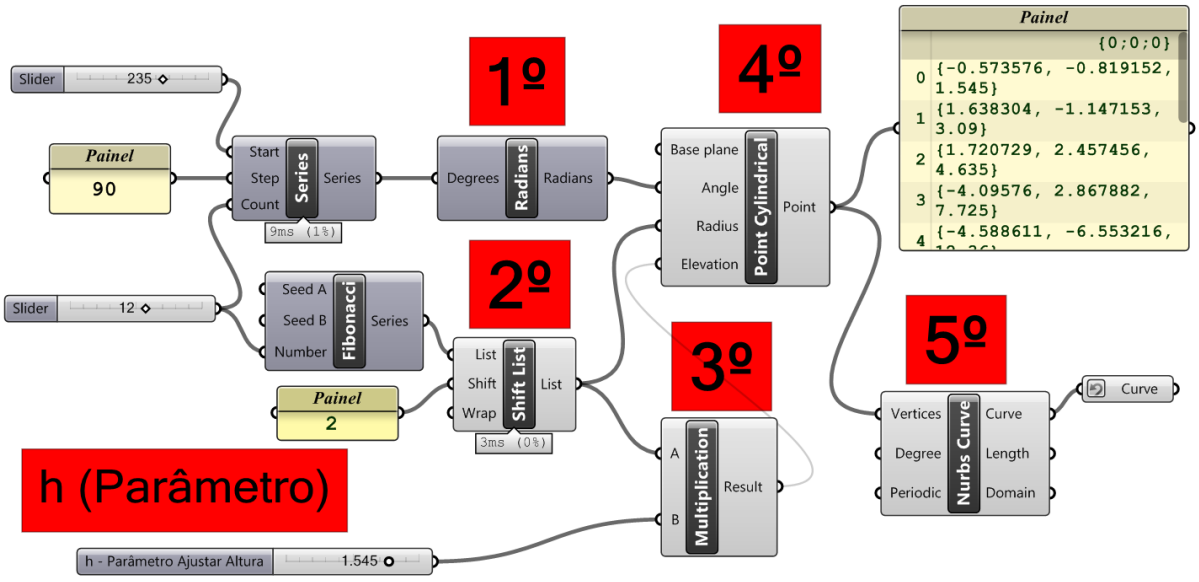
YE, Xiuzi; LIU, Hongzheng; CHEN, Lei; CHEN, Zhiyang; PAN, Xiang; ZHANG, Sanyuan. **Reverse Innovate Dessign - An Integrated Product Design Methodology.** Computer - Aided Design, v. 40, n.7, p.812-827, 2008.

YIWEI, Ren; XIA, Ren**. Interpretation of Parametric Design.** In: Computer-Aided Industrial Design & Conceptual Design (CAIDCD), IEEE 11th International Conference on. IEEE. p. 1458-1461. Beijing, 2010.

APÊNDICE A

Sequência de comandos utilizados no *Grasshopper*.

Detalhamento 1 (Figura 14): Programação *Rhinoceros* e *Grasshopper* da Espiral Áurea.

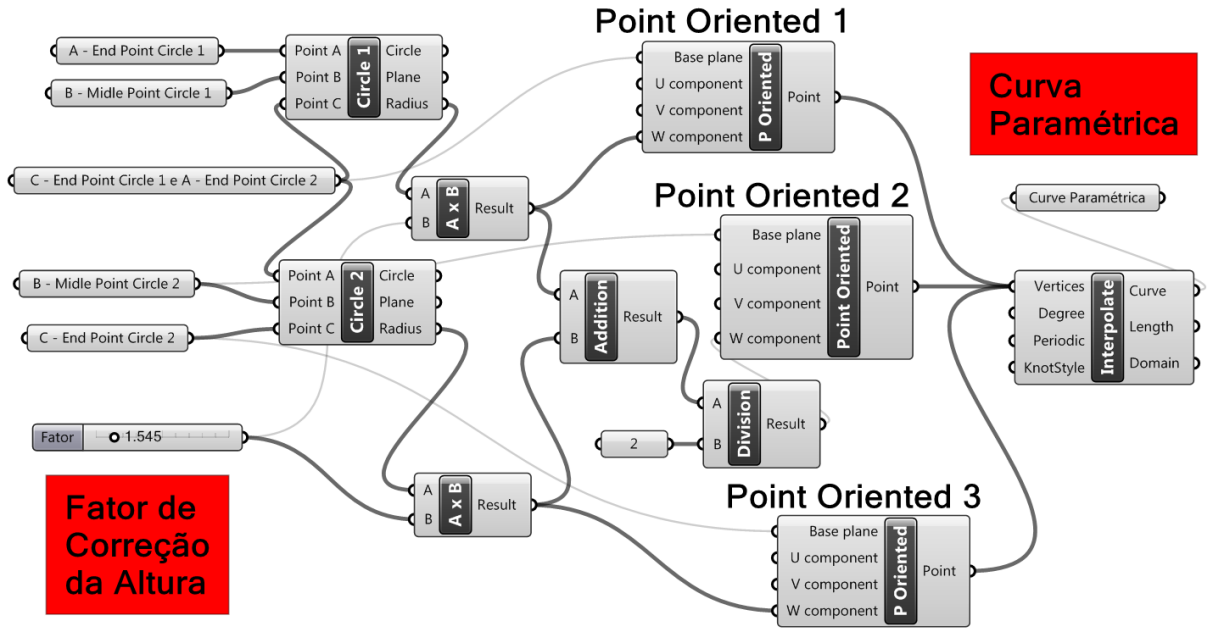


Para este método de programação via "*Grasshopper*" foi utilizado uma sequência de comandos descritos pelas seguintes funções:

- *Slider* - é a principal função de entrada da informação para a programação, ajustando o valor numérico através da ação sobre o controle deslizante.
- *Series* - Função com saída de uma série de números distribuídos de forma uniforme, com passo definido. O parâmetro do "*Slider*" atribuído para "start" define o valor de início da lista. O valor de "*step*" define o passo para a série e o parâmetro "*Count*" define o número de elementos da lista. Para a série de números o valor de início da curva selecionado foi na posição do ângulo "75°" somado ao valor de passo "90°".
- *Radians* - Converte os valores da lista de graus de rotação em Radianos.
- *Panel* - Painel é uma janela informativa. Considerado como um objeto inativo que permite adicionar comentários ou receber informações.

- Fibonacci - Lista de valores da sequência da série de Fibonacci. Na programação foi selecionado as 12 somas iniciais da série de Fibonacci.
- *Shift List* - Item da Lista, seleciona um item ou vários itens da lista. A função *Shift* divide a sequência em uma posição e retorna os valores da direita.
- *Multiplication* - A equação matemática de multiplicação do item “A” por “B”. Saída do valor em “R” (resultado). Na metodologia aplicada o valor de entrada em “A” é o valor 1,545 definido no “*Slider* = h - Parâmetro Ajustar Altura” em “B” entra a lista dos valores selecionados pela função “*Shift List*”.
- *Point Cylindrical* - Ponto cilíndrico, programação para criar pontos projetados em um cilindro variando o ângulo, raio e elevação. O “Plane base” mantido plano XY para definição das coordenadas espaciais do cilindro. O valor para “*Angle*” foi definido em “*Radians*” quando da conversão da lista de números para orientar o ângulo em radianos por ponto para rotação. O parâmetro “*Radius*” configuração do raio do cilindro a partir da lista de valores da sequência da série de Fibonacci . E o parâmetro “*Elevation*” determinado pelo valor da elevação do ponto que é o resultado da equação matemática entre a lista de números e o fator de correção da altura. Na saída teremos uma lista com as coordenadas (x,y,z) dos pontos de controle que estarão distribuídos no espaço formando uma espiral.
- *Nurbs* Curve - Curva NURBS, função de construir uma curva NURBS a partir da ligação dos pontos de controle.
- *Curve* - Parâmetro curva, representa uma geometria em forma de curva.

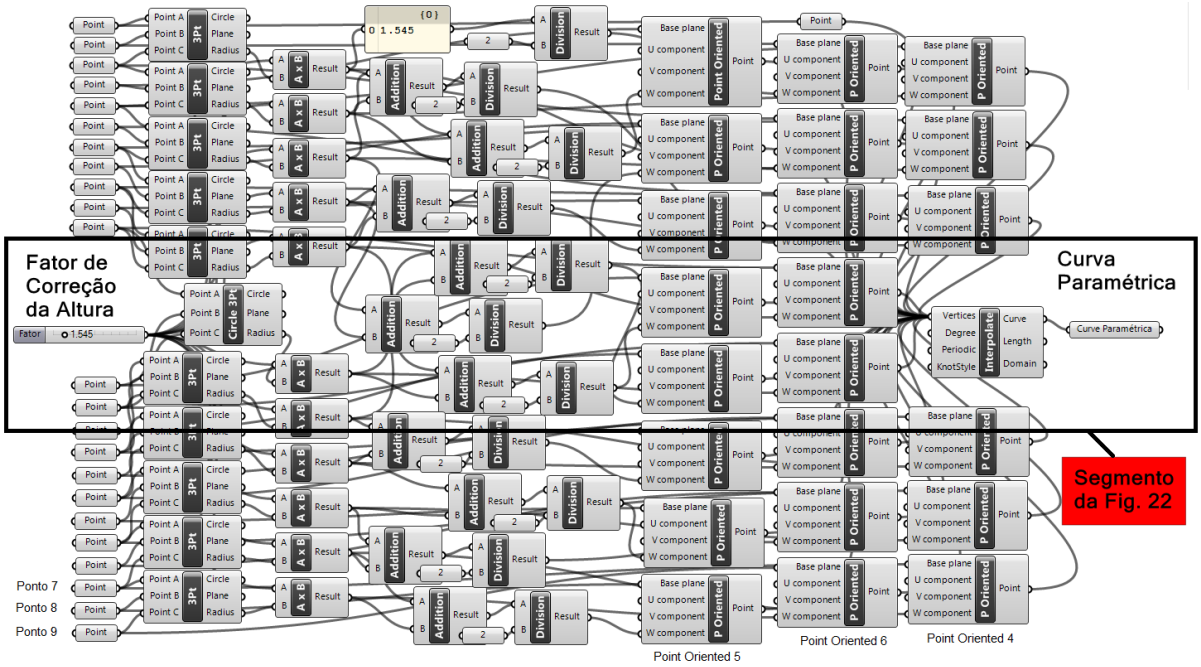
Detalhamento 2 (Figura 25): Programação Visual de um Arco Paramétrico.



Para este método de programação via "*Grasshopper*" foi utilizado uma sequência de comandos descritos pelas seguintes funções:

- *Slider* - Número *Slider*, parâmetro de entrada para definir o fator de correção da altura da projeção da curva espiral.
- *Point* - ponto, através da seleção "*Set Point On*", foi selecionado o ponto sobre o raio da espiral de Fibonacci 2D desenhada no *Rhinoceros*.
- *Circle 3Pt* - Criar um arco ou círculo passando por três pontos. Dados de entrada em que o ponto A (início do arco); ponto B (ponto do meio do arco) e ponto C (ponto final do arco). Dados de Saída em A geometria do Arco resultante, em P plano e R raio do arco.
- *Multiplication, Addition, Division* - Equações matemáticas entre dois números.
- *Point Oriented* - Cria um ponto orientado no espaço a partir das coordenadas (u,v,w).
- *Interpolate Curve* - constrói uma curva interpolada a partir dos pontos de controle.

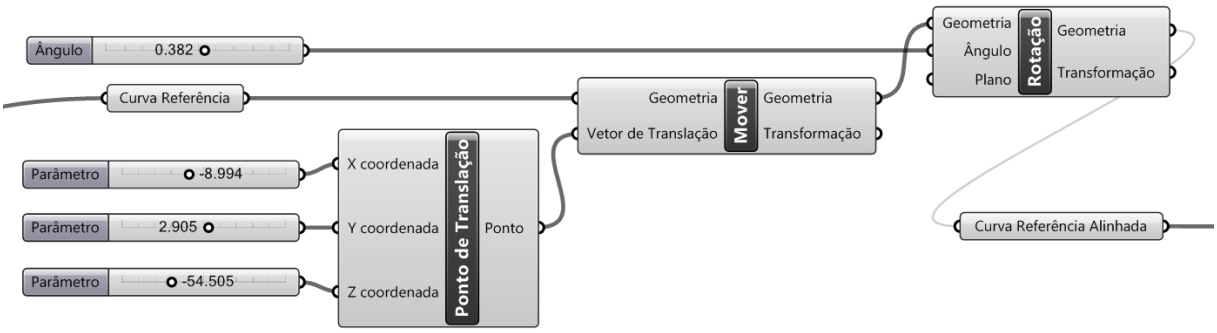
Detalhamento 3 (Figura 27): Programação Visual Completa da “Curva Paramétrica”.



Para este método de programação via “*Grasshopper*” foi utilizado uma sequência de comandos descritos pelas seguintes funções:

- *Slider* - Número *Slider*, parâmetro de entrada para definir o fator de correção da altura da projeção da curva espiral.
- *Point* - ponto, através da seleção “*Set Point On*”, foi selecionado o ponto sobre o raio da espiral de Fibonacci 2D desenhada no *Rhinoceros*.
- *Circle 3Pt* - Criar um arco ou círculo passando por três pontos. Dados de entrada em que o ponto A (início do arco); ponto B (ponto do meio do arco) e ponto C (ponto final do arco). Dados de Saída em A geometria do Arco resultante, em P plano e R raio do arco.
- *Multiplication, Addition, Division* - Equações matemáticas entre dois números.
- *Point Oriented* - Cria um ponto orientado no espaço a partir das coordenadas (u,v,w).
- *Interpolate Curve* - constrói uma curva interpolada a partir dos pontos de controle.

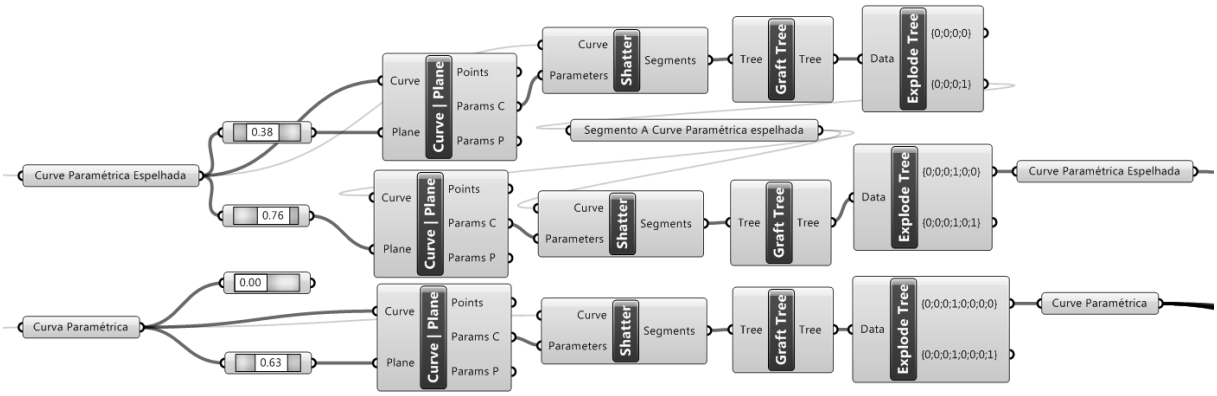
Detalhamento 4 (Figura 30): Programação Visual para os Parâmetros do Alinhamento.



Para este método de programação via "*Grasshopper*" foi utilizado uma sequência de comandos descritos pelas seguintes funções:

- *Slider* - Parâmetro de entrada para definir o fator de correção da posição.
- *Point* - ponto de Translação, através da seleção das coordenadas em x, y e z.
- Mover - Transladar um objeto (geometria) através de um vetor.
- Rotação - Rotacionar (girar) um objeto (geometria) por um determinado ângulo referenciado em um ponto e vetor normal ao plano definido (XY).

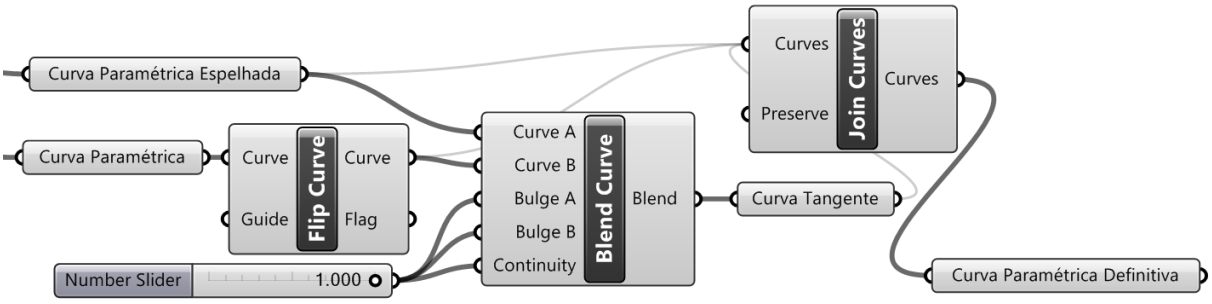
Detalhamento 5 (Figura 33): Programação Visual do Corte das Curvas em Segmentos.



Para este método de programação via “*Grasshopper*” foi utilizado uma sequência de comandos descritos pelas seguintes funções:

- *Slider* - definir o ponto sobre a curva, parâmetro de entrada para definir o local específico da posição do ponto na curva.
- *Intersection - Curve/Plane* - Criar um ponto na intersecção entre a curva e o plano definido por uma posição sobre a curva.
- *Shatter* - Quebrar a curva em partes no ponto de intersecção.
- *Graft Tree* - Fragmentação das listas.
- *Explode Tree* - Divisão das listas em saídas distintas.

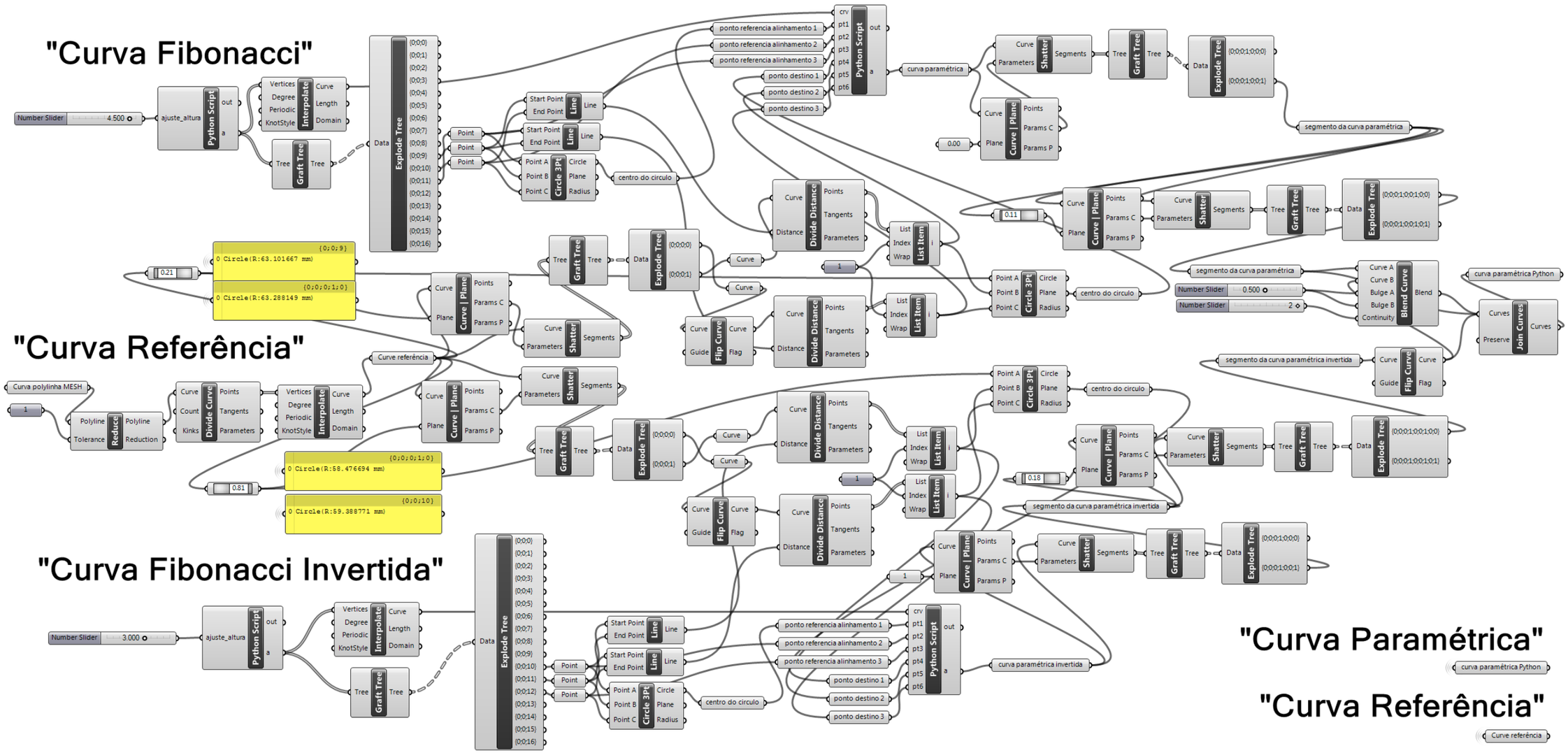
Detalhamento 6 (Figura 34): Programação Visual da “Curva Tangente”.



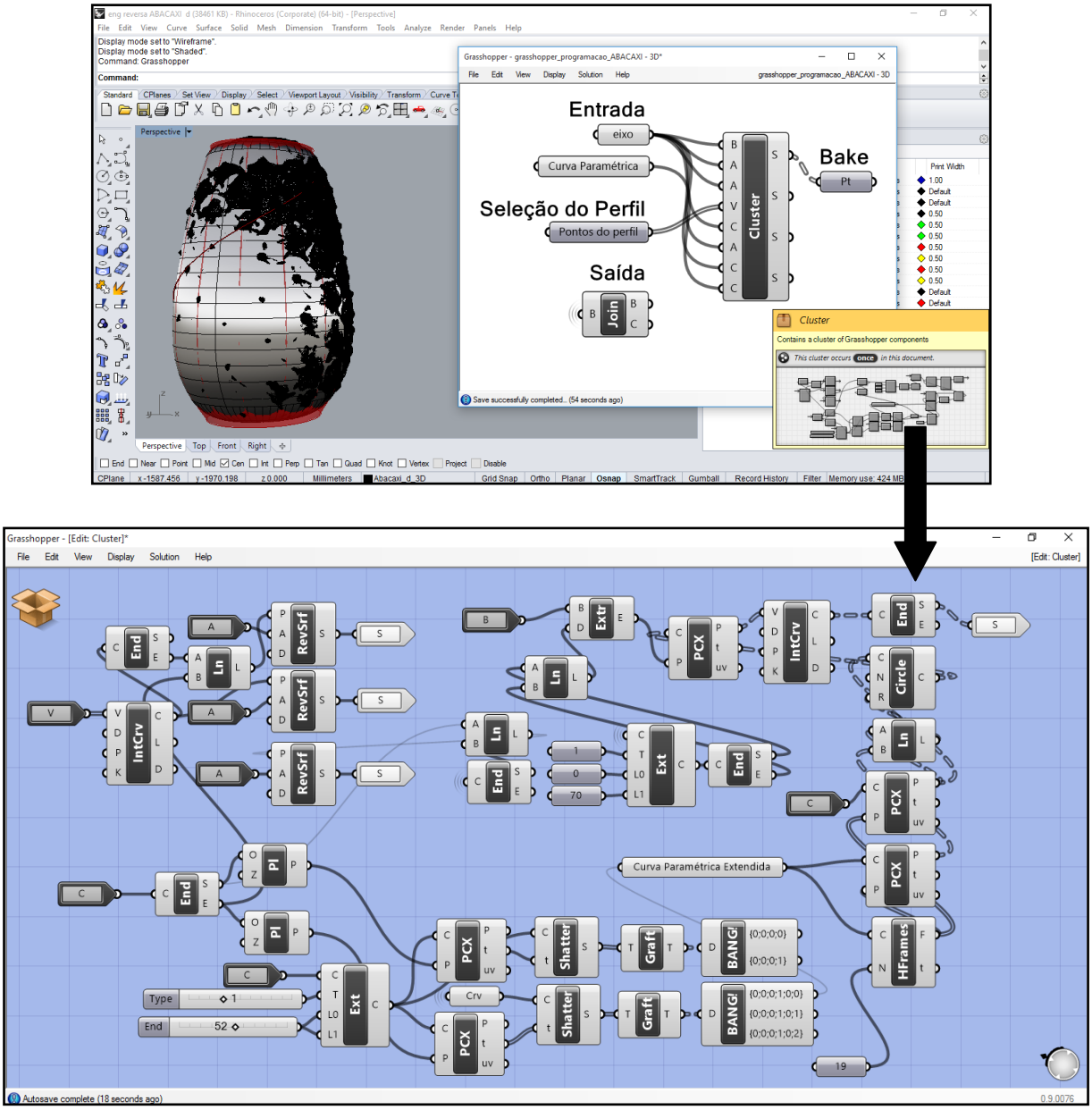
Para este método de programação via “*Grasshopper*” foi utilizado uma sequência de comandos descritos pelas seguintes funções:

- Flip Curve - Virar uma curva usando uma curva-guia.
- *Slider* - parâmetro de entrada para definir o fator de tangência e continuidade da ligação entre as curvas.
- *Blend Curve* - Construir uma curva com continuidade de posição, tangência e curvatura nos pontos de contato dos segmentos selecionados.
- *Join Curves* - Unir diversas curvas, formando uma única curva.

Detalhamento 7 (Ampliação da Figura 43). Modelagem Paramétrica Otimizada.



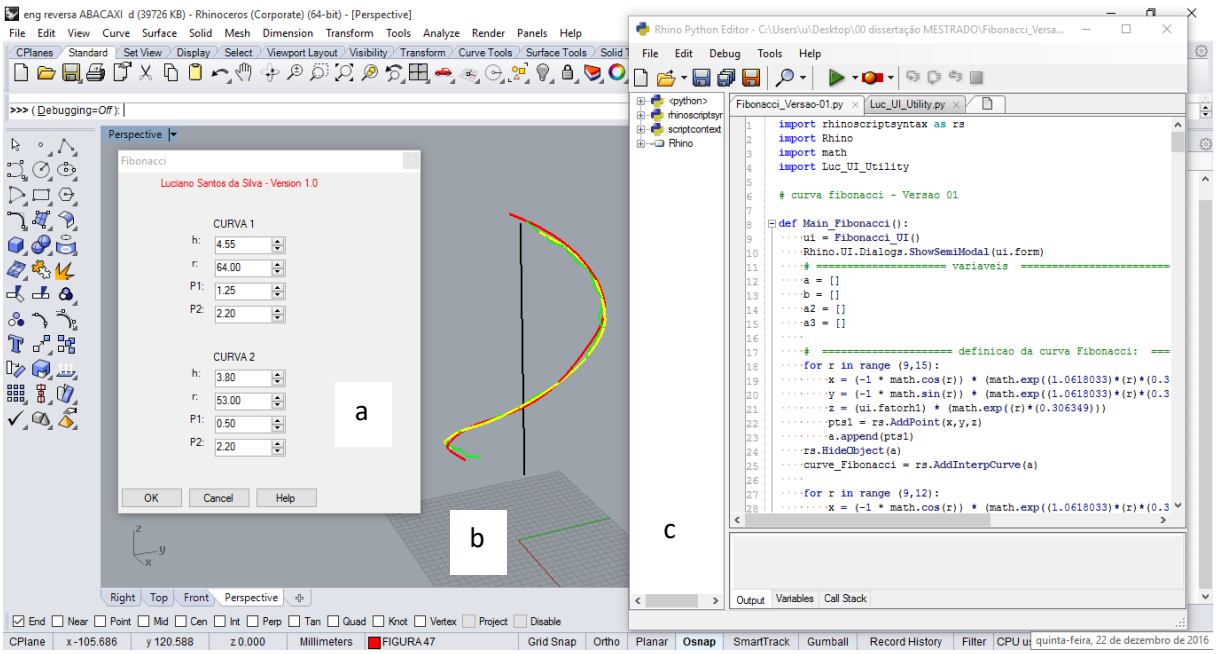
Detalhamento 8 (Figura 53): Geração do Modelo 3D Final.



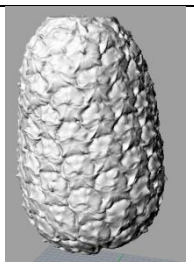
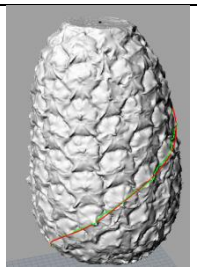
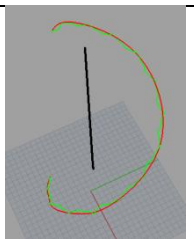
Para este método de programação via "*Grasshopper*" foi utilizada a função da caixa editável, *Cluster.* Esta função possibilita agrupar as funções da modelagem 3D do Abacaxi de maneira editável.

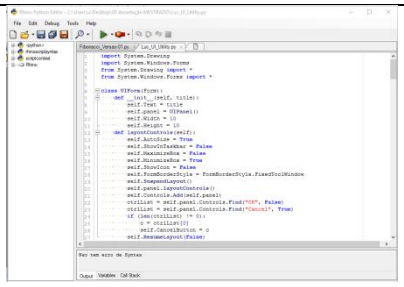
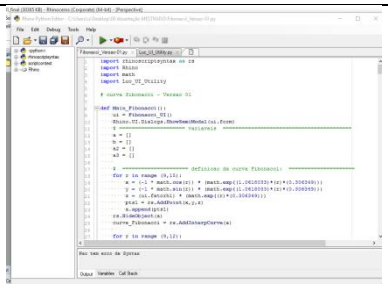
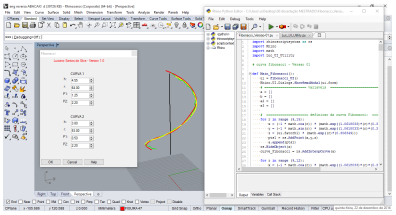
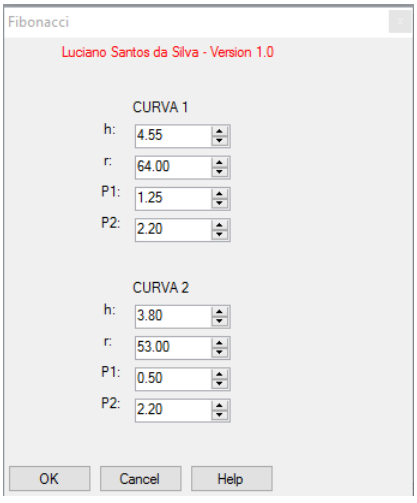
APÊNDICE B

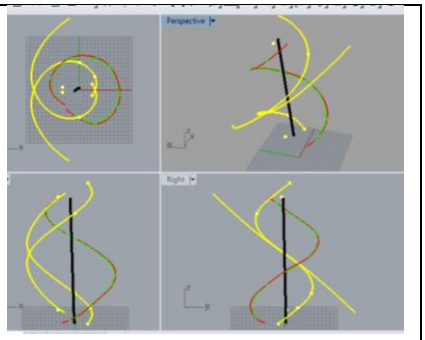
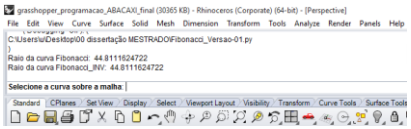
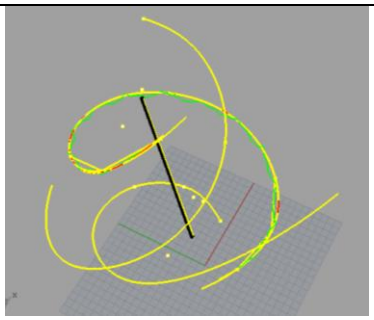
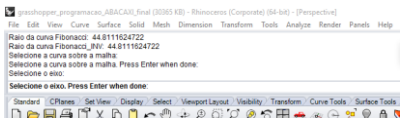
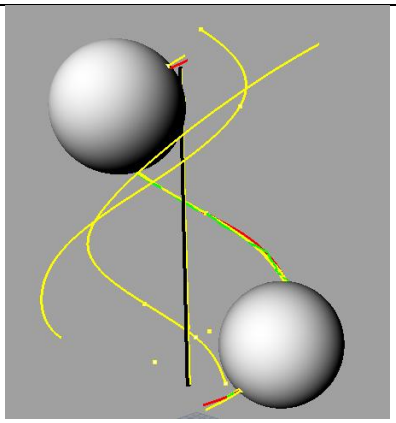
Detalhamento 9 (Figura 47). Interface e programação no *Rhino Python Editor.*

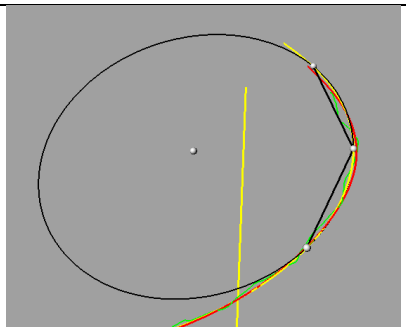
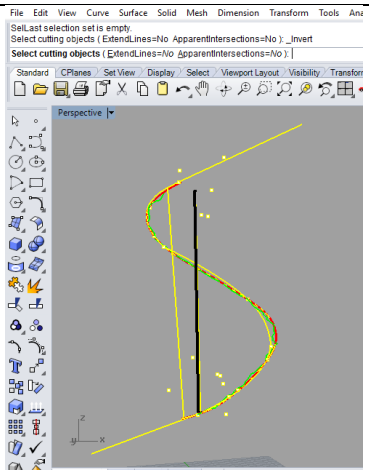
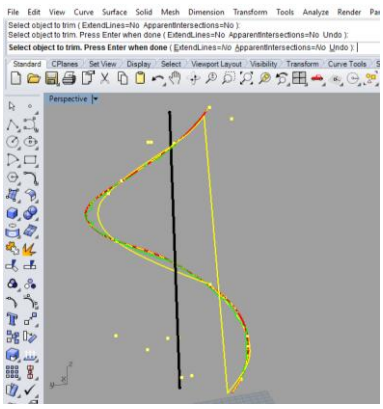
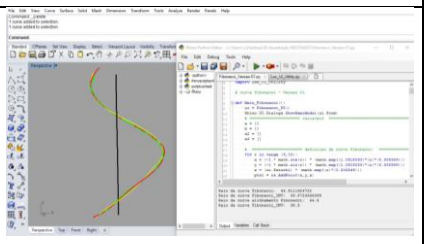


Sequência dos comandos para construção da curva Fibonacci sobre os frutículos do abacaxi, utilizando o software Rhinoceros, Grasshopper e Python.

No Rhinoceros construir uma *layer* com nome stl e a partir do comando *Import*, importar a malha da digitalização do abacaxi.	
Construir uma *layer* Curva Mesh e a partir do comando do Rhino *Curve/Polyline/OnMesh* - construir uma curva sobre o *mesh* sobre os frutículos do abacaxi.	
Comando do Rhino *Curve/Polyline/OnMesh* - construir uma linha de eixo sobre o *mesh* do abacaxi.	

Comando do Rhino *Tools/Python Script/Edit*, no *Rhino Python Editor* selecionar o arquivo Luc_UI_Utility.py e rodar a programação pela função *Debug/StartDebugging* (F5).	
Construir uma layer Curva Equação. A partir do comando do *Rhino Tools/Python Script/Edit*, no *Rhino Python Editor* selecionar o arquivo Fibonacci_Versão-01.py e rodar a programação pela função *Debug/StartDebugging* (F5).	
Na área gráfica so Rhino vai abrir uma caixa com a programação Fibonacci, ajustar os parâmetros: Para a construção da “Curva Equação” é necessário o ajuste de alguns parâmetros na caixa de comandos, conforme apresentado na Figura 47. Os parâmetros apresentados são para construção da “Curva Fibonacci” (curva 1), da “Curva Fibonacci Invertida” (curva 2) e para a “Curva Tangência” (calculada automaticamente). No caso das curvas 1 e 2 o parâmetro “h” corresponde ao Fator da Altura (Ajuste da Altura da Espiral Áurea). O parâmetro “P1” corresponde ao ponto de tangência do círculo de alinhamento com a “Curva Equação” e o “P2” corresponde ao ponto final da curva. O parâmetro “r” corresponde ao valor do raio para construção do círculo e dos pontos de referência para alinhamento da “Curva Equação” (1) ou “Curva Equação Invertida” (2) com a “Curva Referência”. Conforme estabelecido na programação, a “Curva Tangência” será construída automaticamente interpolando de modo contínuo e tangente, ligando a “Curva Fibonacci” e a “Curva Fibonacci Invertida” passando pelos pontos “P2” (curva 1) e “P2” (curva 2).	 

Opção Ok da janela - A programação será executada construindo as curvas equação positiva e invertida.	
Na Prompt será informado o Raio da curva Fibonacci: 44.81... (raio do diâmetro da curva equação) e o Raio da curva Fibonacci_INV: 44.81... (raio do diâmetro da curva equação invertida).	
Na Prompt será solicitado para o usuário. Selecione a curva sobre a malha: a ação é de selecionar com o mouse a curva construida com a opção 2 - construir uma curva sobre o mesh sobre os frutículos do abacaxi. Teclar “espaço” para finalizar a seleção.	
Na Prompt será solicitado para o usuário. Selecione o eixo: a ação é de selecionar com o mouse o eixo construida com a opção 3 - construir uma linha de eixo sobre o mesh do abacaxi. Teclar “espaço” para finalizar a seleção.	
A programação vai seguir com a construção das curvas, esferas e trimagem das curvas.	

O alinhamento das curvas será realizado pela orientação por três pontos (centro do círculo, ponto intersecção do círculo com a corda A e ponto da intersecção do círculo com a corda B).	
Na Prompt será solicitado para o usuário. Select cutting objects (ExtendLines=No ApparentIntersections=No): a ação é de selecionar com o mouse o eixo construida com a opção 3, a seguir teclar “espaço” para finalizar a seleção. Na sequencia será solicitado - Select object to trim. Press Enter When done (ExtendLines=No ApparentIntersections=No Undo): selecionando o excesso das pontas da curva para serem aparadas. Teclar “enter” para finalizar a seleção.	
Na Prompt será solicitado para o usuário. Selecione acurva final: a ação será solicitada para finalizar a curva equação Fibonacci completa, será informado os raios das esferas utilizadas para o alinhamento das curvas. Posicao c1: 44.50 (parâmetro para ajuste do diâmetro da curva equação positiva). Posicao c2: 44.50 (parâmetro para ajuste do diâmetro da curva equação invertida). Estes valores devem ser próximos aos valores informados no item 8 - Raio da curva Fibonacci: 44.81116 (raio do diâmetro da curva equação) e o Raio da curva Fibonacci_INV: 44.81116 (raio do diâmetro da curva equação invertida).	
Finalização com a Curva equação Fibonacci completa construida no Layer selecionado.	

Script utilizado na programação via *Rhino Python Editor*.

```
import rhinoscriptsyntax as rs
import Rhino
import math
import Luc_UI_Utility
# curva fibonacci - Versao 01
def Main_Fibonacci():
    ui = Fibonacci_UI()
    Rhino.UI.Dialogs.ShowSemiModal(ui.form)

# ====== variaveis ======================

    a = []
    b = []
    a2 = []
    a3 = []

#  ===== definicao da curva Fibonacci:  ========

    for r in range (9,15):
        x = (-1 * math.cos(r)) * (math.exp((1.0618033)*(r)*(0.306349)))
        y = (-1 * math.sin(r)) * (math.exp((1.0618033)*(r)*(0.306349)))
        z = (ui.fatorh1) * (math.exp((r)*(0.306349)))
        pts1 = rs.AddPoint(x,y,z)
        a.append(pts1)
    rs.HideObject(a)
    curve_Fibonacci = rs.AddInterpCurve(a)
    for r in range (9,12):
        x = (-1 * math.cos(r)) * (math.exp((1.0618033)*(r)*(0.306349)))
        y = (-1 * math.sin(r)) * (math.exp((1.0618033)*(r)*(0.306349)))
        z = (ui.fatorh1) * (math.exp((r)*(0.306349)))
        p2 = rs.AddPoint(x,y,z)
        a2.append(p2)
        rs.HideObject(a2)
    curve_c_alinhamento = rs.AddInterpCurve(a2)
    start_point2 = rs.CurveStartPoint(curve_c_alinhamento)
    rs.AddPoint(start_point2)
    mid_point2 = rs.CurveMidPoint(curve_c_alinhamento)
    rs.AddPoint(mid_point2)
    end_point2 = rs.CurveEndPoint(curve_c_alinhamento)
    rs.AddPoint(end_point2)
    rs.HideObject(curve_c_alinhamento)
    circle2 = rs.AddCircle3Pt(start_point2, mid_point2, end_point2)
    rs.HideObject (circle2)
    ct_point2 = rs.CircleCenterPoint(circle2)
    rs.AddPoint( ct_point2 )
    line1 = rs.AddLine(ct_point2,mid_point2)
    #valor_raio_Fib = rs.DimensionValue(line1
    line2 = rs.AddLine(start_point2, mid_point2)
    line3 = rs.AddLine(mid_point2,end_point2)
    print “Raio da curva Fibonacci: “, rs.Distance(ct_point2, mid_point2)
    rs.HideObject(line1)
    rs.HideObject(line2)
    rs.HideObject(line3)

    curve = rs.AddInterpCurve(a)
    if rs.IsCurve(curve):
```

```
        domain = rs.CurveDomain(curve)
    domain[1] /= 1.75

#  ==== definicao da curva Fibonacci_INV: =====
    for r in range (9,15):

        x = (-1 * math.cos(r)) * (math.exp((1.0618033)*(r)*(0.306349)))
        y = (1 * math.sin(r)) * (math.exp((1.0618033)*(r)*(0.306349)))
        z = (-ui.fatorh2) * (math.exp((r)*(0.306349)))
        pts2 = rs.AddPoint(x,y,z)
        b.append(pts2)
        rs.HideObject(b)
        pl = (280)
        rs.MoveObject(pts2,(pl*0,pl*0,pl*1))
    curve_Fibonacci_INV = rs.AddInterpCurve(b)
    for r in range (9,12):
        x = (-1 * math.cos(r)) * (math.exp((1.0618033)*(r)*(0.306349)))
        y = (1 * math.sin(r)) * (math.exp((1.0618033)*(r)*(0.306349)))
        z = (-ui.fatorh2) * (math.exp((r)*(0.306349)))
        p3 = rs.AddPoint(x,y,z)
        a3.append(p3)
        rs.HideObject(a3)
        rs.MoveObject(p3,(pl*0,pl*0,pl*1))
    curve_c_alinhamento = rs.AddInterpCurve(a3)
    start_point3 = rs.CurveStartPoint(curve_c_alinhamento)
    rs.AddPoint(start_point3)
    mid_point3 = rs.CurveMidPoint(curve_c_alinhamento)
    rs.AddPoint(mid_point2)
    end_point3 = rs.CurveEndPoint(curve_c_alinhamento)
    rs.AddPoint(end_point3)
    rs.HideObject(curve_c_alinhamento)
    circle3 = rs.AddCircle3Pt(start_point3, mid_point3, end_point3)
    rs.HideObject (circle3)
    center_point3 = rs.CircleCenterPoint(circle3)
    rs.AddPoint( center_point3 )
    line11 = rs.AddLine(center_point3,mid_point3)
    line21 = rs.AddLine(start_point3, mid_point3)
    line31 = rs.AddLine(mid_point3,end_point3)
    print “Raio da curva Fibonacci_INV: “, rs.Distance(center_point3, mid_point3)
    rs.HideObject(line11)
    rs.HideObject(line21)
    rs.HideObject(line31)
    curve2 = rs.AddInterpCurve(b)
    if rs.IsCurve(curve2):
        domain = rs.CurveDomain(curve2)
    domain[1] /= 1.75

#  === definicao da curva sobre a malha da digitalizacao: ===

    crv_mesh= rs.GetObjects(“Selecione a curva sobre a malha”,rs.filter.curve)
    if crv_mesh: rs.IsCurve(crv_mesh)
    points = rs.DivideCurve (crv_mesh,40,False,True)
    polyline = rs.AddInterpCurve(points, degree=1, knotstyle=2, start_tangent=None, end_tangent=None)
    Crv_ext_mesh  = rs.ExtendCurveLength(polyline,2, 2, 12 )
    points2 = rs.DivideCurve (Crv_ext_mesh,10,False,True)
    Curva_Interpolada = rs.AddInterpCurve(points2, degree=5, knotstyle=5, start_tangent=None,
end_tangent=None)
    mid_point_alinhamento = rs.CurveMidPoint(Curva_Interpolada)
```

```
    rs.AddPoint(mid_point_alinhamento)
    rs.HideObject(Crv_ext_mesh)
    crv_int_mesh = rs.CopyObject(crv_mesh)
    points22_mesh = rs.DivideCurve (crv_int_mesh,10,False,True)
    Curva_Interpolada_mesh = rs.AddInterpCurve(points22_mesh, degree=5, knotstyle=5, start_tangent=None,
end_tangent=None)

#  ========= definicao do eixo a malha da digitalizacao: ==========

    eixo_mesh = rs.GetObjects("Selecione o eixo",rs.filter.curve)
    p1 = rs.CurveStartPoint(eixo_mesh)
    #rs.AddPoint(p1)
    p2 = rs.CurveEndPoint(eixo_mesh)
    #rs.AddPoint(p2)
    eixo1 = rs.AddLine(p1,p2)
    vetor1 = rs.VectorCreate(p1, p2)
    plane1 = rs.PlaneFromNormal(p1, vetor1)
    plane2 = rs.PlaneFromNormal(p2, vetor1)

#  === definicao dos pontos para alinhamento sobre a curva da malha da digitalizacao: ======

    Curva_Int2 = rs.CopyObject(Curva_Interpolada)
    crv_ref_fibonacci = (Curva_Int2)
    if crv_ref_fibonacci: rs.IsCurve(crv_ref_fibonacci)
    domain1 = rs.CurveDomain(crv_ref_fibonacci)
    domain1[1] /= 1.7 #parametro da posicao circulo da curva Fibonacci
    Curva_trimada_Fibonacci = rs.TrimCurve( crv_ref_fibonacci, domain1 )
    Curva_trimada_Fibonacci2 = rs.CopyObject(Curva_trimada_Fibonacci)
    Curva_trimada = rs.CopyObject(Curva_trimada_Fibonacci) #curva FIBONACCI
    curv2 = (Curva_trimada_Fibonacci)
    if curv2: rs.IsCurve(curv2)
    domain2 = rs.CurveDomain(Curva_trimada_Fibonacci)
    domain2[1] /= (ui.posicao1) #parametro da posicao circulo da curva Fibonacci
    curv3 = rs.TrimCurve( Curva_trimada_Fibonacci, domain2)
    point_curv3 = rs.CurveEndPoint(curv3)
    rs.AddPoint(point_curv3)
    if rs.IsCurve(curv3): rs.ReverseCurve(curv3)
    curv3inv = curv3
    if curv3: rs.IsCurve(curv3)
    esfera1 = rs.AddSphere(point_curv3, 32)
    p10 = rs.AddPoint(0,0,0)
    p11 = rs.AddPoint(0,0,0.1)
    eixo2 = rs.AddLine((p10),(p11))
    extrude1 = rs.ExtrudeCurve(curv3,eixo2)
    line = rs.TrimBrep(extrude1,esfera1)
    rs.HideObject(esfera1)
    rs.HideObject(extrude1)
    rs.HideObject(curv3)
    intersec = rs.IntersectBreps( esfera1, extrude1)
    point12 = rs.CurveStartPoint(intersec)
    rs.AddPoint(point12)
    linea = rs.AddLine(point_curv3,point12)
    rs.HideObject(p10)
    rs.HideObject(p11)
    rs.HideObject(eixo2)
    rs.HideObject(intersec)
    if rs.IsCurve(Curva_trimada_Fibonacci2): rs.ReverseCurve(Curva_trimada_Fibonacci2)
    curv22 = (Curva_trimada_Fibonacci2)
```

```
curv_ref2 = curv22
if curv_ref2: rs.IsCurve(curv_ref2)
domain22 = rs.CurveDomain(curv_ref2)
domain22[1] /= 0.7783
curv33 = rs.TrimCurve( curv_ref2, domain22)
point_curv33 = rs.CurveEndPoint(curv33)
rs.AddPoint(point_curv33)
if curv33: rs.IsCurve(curv33)
esfera11  = rs.AddSphere(point_curv3, 32)
extrude11 = rs.ExtrudeCurve(curv33,eixo2)
lineb = rs.TrimBrep(extrude11,esfera11)
rs.HideObject(esfera11)
rs.HideObject(extrude11)
rs.HideObject(curv33)
intersec2 = rs.IntersectBreps( esfera11, extrude11)
point122 = rs.CurveStartPoint(intersec2)
rs.AddPoint(point122)
linec = rs.AddLine(point_curv3,point122)
rs.HideObject(intersec2)
circle_Fibonacci = rs.AddCircle3Pt((point122),(point_curv3),(point12))
rs.HideObject (circle_Fibonacci)
center_point22 = rs.CircleCenterPoint(circle_Fibonacci)
rs.AddPoint( center_point22 )
line11 = rs.AddLine(point_curv3, center_point22)
esfera111  = rs.AddSphere(point_curv3, (ui.r_esf111))
extrude111 = rs.ExtrudeCurve(line11,eixo2)
line = rs.TrimBrep(extrude111,esfera111)
rs.HideObject(esfera111)
rs.HideObject(extrude111)
rs.HideObject(line11)
intersec2 = rs.IntersectBreps( esfera111, extrude111)
pt_ct_al1  = rs.CurveEndPoint(intersec2)
rs.AddPoint( pt_ct_al1 )
print “Raio da curva alinhamento Fibonacci: “, rs.Distance((pt_ct_al1), (point_curv3))
Curva_Intinv = rs.CopyObject(Curva_Interpolada)
crv_rev = (Curva_Intinv)
if rs.IsCurve(crv_rev): rs.ReverseCurve(crv_rev)
crv_ref = crv_rev
if crv_ref: rs.IsCurve(crv_ref)
domain4 = rs.CurveDomain(crv_ref)
domain4[1] /= (ui.posicao2) #parametro da posicao circulo da curva Fibonacci_INV
ref2 = rs.TrimCurve( crv_ref, domain4 )
curv_ref2_inv = rs.CopyObject(ref2)
curv_ref2_inv2 = rs.CopyObject(ref2)
Curva_trimada_INV = rs.CopyObject(ref2) #curva FIBONACCI_INV
rs.HideObject(ref2)
if curv_ref2_inv: rs.IsCurve(curv_ref2_inv)
domain5 = rs.CurveDomain(curv_ref2_inv)
domain5[1] /= 0.86
curv5 = rs.TrimCurve(curv_ref2_inv, domain5)
curv5_end_point = rs.CurveEndPoint(curv5)
rs.AddPoint(curv5_end_point)
esfera22  = rs.AddSphere(curv5_end_point, 32)
extrude22 = rs.ExtrudeCurve(curv5,eixo2)
line33 = rs.TrimBrep(extrude22,esfera22)
rs.HideObject(esfera22)
rs.HideObject(extrude22)
intersec22 = rs.IntersectBreps( esfera22, extrude22)
```

```
    point_intersec22 = rs.CurveStartPoint(intersec22)
    rs.AddPoint(point_intersec22)
    line43 = rs.AddLine(curv5_end_point,point_intersec22)
    rs.HideObject(intersec22)
    if curv_ref2_inv2: rs.IsCurve(curv_ref2_inv2)
    if rs.IsCurve(curv_ref2_inv2): rs.ReverseCurve(curv_ref2_inv2)
    curv_ref2_inv22 = curv_ref2_inv2
    if curv_ref2_inv22: rs.IsCurve(curv_ref2_inv22)
    domain6 = rs.CurveDomain(curv_ref2_inv22)
    domain6[1] /= 1.0965
    curv6 = rs.TrimCurve(curv_ref2_inv22, domain6)
    curv6_end_point = rs.CurveEndPoint(curv6)
    rs.AddPoint(curv6_end_point)
    esfera33  = rs.AddSphere(curv6_end_point, 32)
    extrude33 = rs.ExtrudeCurve(curv6,eixo2)
    line53 = rs.TrimBrep(extrude33,esfera33)
    rs.HideObject(esfera33)
    rs.HideObject(extrude33)
    intersec33 = rs.IntersectBreps( esfera33, extrude33)
    point_intersec33 = rs.CurveStartPoint(intersec33)
    rs.AddPoint(point_intersec33)
    line63 = rs.AddLine(curv6_end_point,point_intersec33)
    rs.HideObject(intersec33)
    rs.HideObject(curv5)
    rs.HideObject(curv6)
    circle_Fibonacci_inv = rs.AddCircle3Pt((point_intersec22),(curv5_end_point),(point_intersec33))
    rs.HideObject (circle_Fibonacci_inv)
    center_point22_inv = rs.CircleCenterPoint(circle_Fibonacci_inv)
    rs.AddPoint( center_point22_inv )
    line73 = rs.AddLine(center_point22_inv,curv5_end_point)
    line83 = rs.AddLine(curv5_end_point,point_intersec33)
    esfera222  = rs.AddSphere(curv5_end_point, (ui.r_esf222))
    extrude222 = rs.ExtrudeCurve(line73,eixo2)
    line93 = rs.TrimBrep(extrude222,esfera222)
    rs.HideObject(esfera222)
    rs.HideObject(extrude222)
    rs.HideObject(line93)
    rs.HideObject(line73)
    intersec2_inv = rs.IntersectBreps( esfera222, extrude222)
    pt_ct_al2  = rs.CurveStartPoint(intersec2_inv)
    rs.AddPoint( pt_ct_al2 )
    print “Raio da curva Fibonacci_INV: “, rs.Distance(pt_ct_al2,curv5_end_point)

#  == alinhamento curva Fibonacci Fibonacci: ===

    crv_alinhada_Fibonacci =
rs.OrientObject(curve_Fibonacci,((ct_point2),(end_point2),(start_point2)),((pt_ct_al1),(point122),(point12)),1)

#  === alinhamento curva Fibonacci Fibonacci_INV: ====

    crv_alinhada_Fibonacci_INV =
rs.OrientObject(curve_Fibonacci_INV,((center_point3),(end_point3),(start_point3)),((pt_ct_al2),(point_intersec33),
(point_intersec22)),1)

#  ====== extender as curva Fibonacci: =======

    Crv_ext1 = rs.ExtendCurveLength(crv_alinhada_Fibonacci, 0, 0, 100 )
    Crv_ext2 = rs.ExtendCurveLength(crv_alinhada_Fibonacci_INV, 0, 0, 100 )
```

```
# ==================== hide objetos ====================
   rs.HideObject (curve_Fibonacci)
   rs.HideObject (curve_Fibonacci_INV)
   rs.HideObject(curve_c_alinhamento)
   rs.HideObject (curv2)
   rs.HideObject (curve)
   rs.HideObject (curve2)
   rs.HideObject (Curva_Intinv)
   rs.HideObject(line11)
   rs.HideObject(line21)
   rs.HideObject(line31)
   rs.HideObject(line33)
   rs.HideObject(line43)
   rs.HideObject(line53)
   rs.HideObject(line63)
   rs.HideObject(line73)
   rs.HideObject(line83)
   rs.HideObject(line93)
   rs.HideObject(linea)
   rs.HideObject(lineb)
   rs.HideObject(linec)
   rs.HideObject (Curva_trimada)
   rs.HideObject (Curva_trimada_INV)

# ==== trimagem das curvas ==========

   crv_corte1 = (Crv_ext1)
   if crv_corte1: rs.IsCurve(crv_corte1)
   domain77 = rs.CurveDomain(Crv_ext1)
   domain77 [1] /= (ui.corte1) #parametro da trimagem da curva Fibonacci
   crv_corte1_res = rs.TrimCurve( Crv_ext1, domain77)
   point_corte1_res = rs.CurveEndPoint(crv_corte1_res)
   rs.AddPoint(point_corte1_res)
   crv_corte2_INV = (Crv_ext2)
   if crv_corte2_INV: rs.IsCurve(crv_corte2_INV)
   domain88 = rs.CurveDomain(Crv_ext2)
   domain88 [1] /= (ui.corte2) #parametro da trimagem da curva Fibonacci_INV
   crv_corte2_INV_res = rs.TrimCurve( Crv_ext2, domain88)
   point_corte2_res = rs.CurveEndPoint(crv_corte2_INV_res)
   rs.AddPoint(point_corte2_res)
   mid_point_crv_liga = (mid_point_alinhamento)
   points = (point_corte1_res),(mid_point_crv_liga),(point_corte2_res)
   crv_liga = rs.AddInterpCurve(points, degree=3, knotstyle=5, start_tangent=None, end_tangent=None)

# ===== trimagem das curvas ===========

   Circle1 = rs.AddCircle(plane1,100)
   Circle2 = rs.AddCircle(plane2,100)
   surface1 = rs.AddPlanarMesh (Circle1)
   surface2 = rs.AddPlanarMesh (Circle2)
   cmx1 = rs.CurveMeshIntersection(crv_corte1_res, surface1, True)
   if cmx1:
      for element in cmx1:
         """print element[0], ", Face index = ", element[1]"""
   pt_int1 = rs.AddPoint(element[0])
   cmx1 = rs.CurveMeshIntersection(crv_corte2_INV_res, surface2, True)
   if cmx1:
```

```
        for element in cmx1:
            """print element[0], ", Face index = ", element[1]"""
    pt_int2 = rs.AddPoint(element[0])
    linetrim = rs.AddLine (pt_int1,pt_int2)
    rs.HideObject (Circle1)
    rs.HideObject (Circle2)
    rs.HideObject (surface1)
    rs.HideObject (surface2)
    rs.HideObject (pt_int1)
    rs.HideObject (pt_int2)
    rs.HideObject(Crv_ext_mesh)
    rs.HideObject (Curva_Interpolada)
    objs = ((crv_corte1_res),(crv_corte2_INV_res),(crv_liga))
    rs.HideObject (objs)
    if objs: rs.JoinCurves(objs)
    crv100 = (objs,rs.filter.curve)
    crv_trim = rs.Command("_Trim _SelLast _Invert")
    crv102 = (crv_trim,rs.filter.curve)
    rs.HideObject (linetrim)
    crv104 = (crv102,rs.filter.curve)
    crv106 = rs.GetObjects("Selecione a curva final",rs.filter.curve)
    if crv106: rs.IsCurve(crv106)
    points2 = rs.DivideCurve(crv106,10,False,True)
    Fibonacci_Abacaxi = rs.AddInterpCurve(points2, degree=5, knotstyle=5, start_tangent=None,
end_tangent=None)
    rs.HideObject (crv106)
    #rs.HideObject (eixo1)

#  ===== Caixa de Parametros ===========

class Fibonacci_UI():
    def __init__(self):
        updnWidth = 80

        self.fatorh1 = 4.55
        self.fatorh2 = 3.80
        self.r_esf111 = 64.00
        self.r_esf222 = 53.00
        self.posicao1 = 1.25
        self.posicao2 = 0.50
        self.corte1 = 2.20
        self.corte2 = 2.20

        self.form = Luc_UI_Utility.UIForm("Fibonacci")
        self.form.panel.addLabel("", "            Luciano Santos da Silva - Version 1.0", (255, 0, 0), True)
        self.form.panel.addLabel("", "", None, True)

        self.form.panel.addLabel("", "                        CURVA 1", (0, 0, 0), True)
        self.form.panel.addLabel("", "                    h:", None, False)
        self.form.panel.addNumericUpDown("", 0.00, 6, 0.50, 2, self.fatorh1, updnWidth, True,
self.Fatorh1_OnValueChange)
        self.form.panel.addLabel("", "                    r:", None, False)
        self.form.panel.addNumericUpDown("", 38.50, 72.00, 6, 2, self.r_esf111, updnWidth, True,
self.R_esf111_OnValueChange)
        self.form.panel.addLabel("", "                    P1:", None, False)
        self.form.panel.addNumericUpDown("", 1.20, 1.50, 0.001, 2, self.posicao1, updnWidth, True,
self.Posicao1_OnValueChange)
        self.form.panel.addLabel("", "                    P2:", None, False)
```

```
        self.form.panel.addNumericUpDown("", 1.50, 2.50, 0.01, 2, self.corte1, updnWidth, True,
self.Corte1_OnValueChange)
            self.fm.panel.addLabel("", "", None, True)
            self.form.panel.addLabel("", "                          CURVA 2", (0, 0, 0), True)
        self.form.panel.addLabel("", "                    h:", None, False)
        self.form.panel.addNumericUpDown("", 0.00, 6, 0.50, 2, self.fatorh2, updnWidth, True,
self.Fatorh2_OnValueChange)
        self.form.panel.addLabel("", "                    r:", None, False)
        self.form.panel.addNumericUpDown("", 38.50, 72.00, 6, 2, self.r_esf222, updnWidth, True,
self.R_esf222_OnValueChange)
        self.form.panel.addLabel("", "                    P1:", None, False)
        self.form.panel.addNumericUpDown("", 0.400, 0.5, 0.001, 2, self.posicao2, updnWidth, True,
self.Posicao2_OnValueChange)
        self.form.panel.addLabel("", "                    P2:", None, False)
        self.form.panel.addNumericUpDown("", 1.50, 2.50, 0.01, 2, self.corte2, updnWidth, True,
self.Corte2_OnValueChange)
        self.form.panel.addLabel("", "", None, True)
        self.addControls()
        self.form.layoutControls()
         def addControls(self):
        buttonWidth = 70
        p = self.form.panel
        p.addButton("OK", "OK", buttonWidth, False, None)
        p.addButton("Cancel", "Cancel", buttonWidth, False, None)
        p.addButton("Help", "Help", buttonWidth, False, self.Help_OnButtonPress)
            def Help_OnButtonPress(self, sender, e):
        rs.MessageBox("Os parametros apresentados sao para construcao da Curva Fibonacci curva 1. Para Curva
Fibonacci Invertida curva 2. Para a Curva Tangencia calculada automaticamente. No caso das curvas 1 e 2 o
parametro h corresponde ao Fator da Altura ou seja Ajuste da Altura da Espiral Aurea. O parametro P1
corresponde ao ponto de tangência do círculo de alinhamento com a “Curva Equação” e o P2 corresponde ao
ponto final da curva. O parametro r corresponde ao valor do raio para construcao do circulo e dos pontos de
referencia para alinhamento da Curva Equacao 1 ou Curva Equacao Invertida 2 com a Curva Referencia.
Conforme estabelecido na programacao, a Curva Tangencia sera construida automaticamente interpolando de
modo continuo e tangente, ligando a Curva Fibonacci e a Curva Fibonacci Invertida passando pelos pontos P2
curva 1 e P2 curva 2.", 0, "Help")

    def Fatorh1_OnValueChange(self, sender, e):
        self.fatorh1 = sender.Value
    def Fatorh2_OnValueChange(self, sender, e):
        self.fatorh2 = sender.Value

    def R_esf111_OnValueChange(self, sender, e):
        self.r_esf111 = sender.Value
    def R_esf222_OnValueChange(self, sender, e):
        self.r_esf222 = sender.Value

    def Posicao1_OnValueChange(self, sender, e):
        self.posicao1 = sender.Value
    def Posicao2_OnValueChange(self, sender, e):
        self.posicao2 = sender.Value

    def Corte1_OnValueChange(self, sender, e):
        self.corte1 = sender.Value
    def Corte2_OnValueChange(self, sender, e):
        self.corte2 = sender.Value

if( __name__ == "__main__" ):
    Main_Fibonacci()
```