Uso de métodos computacionais para automatizar a documentação BIM em edificações existentes a partir de documentos 2D

Application of computational methods for automating BIM documentation in existing building from 2D documents

Michelle da Silva Matias
Universidade Federal de Pernambuco | Recife | Brasil | michelle.matias@ufpe.br
Rachel Perez Palha
Universidade Federal de Pernambuco | Recife | Brasil | rachel.palha@ufpe.br

Resumo
O BIM vem revolucionando a indústria da construção, dando suporte para unificar projeto e execução, sendo foco de mandatos BIM no mundo todo. Esta pesquisa tem como objetivo desenvolver uma metodologia que permita automatizar o processo de passar as-builts de obras pré-existentes produzidos em 2D para BIM em 3D. Para tal, está sendo utilizado o formato IFC e o desenvolvimento do código está sendo feito em Python. A ideia aqui é desenvolver o modelo IFC a partir das imagens de plantas baixas com adição de outras informações e gerar os modelos dos edifícios para uso na fase de manutenção.

Palavras-chave: Construções. Automatização. *As Built*. BIM. IFC.

Abstract
BIM has revolutionized the construction industry, providing support to unify design and execution, being the focus of BIM mandates worldwide. This research aims to develop a methodology that allows to automate the process of passing as-builts of pre-existing buildings produced in 2D to BIM in 3D. To this end, the IFC format is being used and the code is being developed in Python. The idea here is to develop the IFC model from the floor plan images with the addition of other information and generate the building models for use in the maintenance phase.

Keywords: Construction. Automation. As Built. BIM. IFC.

INTRODUÇÃO

A tecnologia BIM (*Building Information Modeling*) têm facilitado o processo da produção dos projetos arquitetônicos e de engenharia até sua execução, os tornando integrados de tal forma que resulta em obras de qualidade final superior, tanto na redução dos custos, quanto de prazo de execução [1]. Segundo Doukari e Greenwood [2], com esse avanço na engenharia civil, deu-se a possibilidade de sair da linha a linha, para modelos de realidade 3D, tornou-se possível interligar projetos elétricos,

Como citar:
MATIAS, M. da S.; PALHA, R. P. Uso de métodos computacionais para automatizar a documentação BIM em edificações existentes a partir de documentos 2D. *In*: SIMPÓSIO BRASILEIRO DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO NA CONSTRUÇÃO, 3., 2021, Uberlândia. **Anais [...]**. Porto Alegre: ANTAC, 2021. p. 1-12. Disponível em: https://eventos.antac.org.br/index.php/sbtic/article/view/606. Acesso em: 3 ago. 2021.

hidráulicos e arquitetônicos, por exemplo, de formar rápida e a diminuir custos durante a execução. Apesar de todas as vantagens, o processo de adaptação a esta nova realidade têm sido intenso [1].

Mundialmente, cada país está estabelecendo metas para tornar essa nova tecnologia uma prática nos projetos cotidianos no ramo da construção civil [1][3]. O Reino Unido, por exemplo, é um modelo bem-sucedido no seu plano estabelecido em conformidade com a ISO 19650, e atualmente, está com as metas adiantadas ao que era esperado pelas metas do UK BIM Alliance Chair [4]. No Brasil, esta disseminação tem como força motriz o Mandato BIM-BR, com o intuito de incentivar o setor AECO, reduzir custos nas compras públicas, dar maior transparência aos processos licitatórios e contribuir para melhoria dos processos de manutenção e gerenciamento de ativos [3]. Nesta estratégia foram definidas metas a serem alcançadas, que devem ser ainda mais bem delineadas após o desenvolvimento dos primeiros projetos pilotos.

A Estratégia BIM-BR estabelece três fases de implantação, onde a última trata do uso do BIM para a manutenção e operação de edifícios públicos [3]. Nos novos edifícios, considerando que os mesmos tenham tido seus projetos desenvolvidos já usando BIM, esse não é um problema, afinal, devendo fazer parte do planejamento estratégico dos órgãos públicos. Entretanto, os edifícios já construídos podem ser um problema, pois toda sua informação foi produzida em 2D. Desse modo, este estudo tem como objetivo desenvolver uma metodologia que possa auxiliar a automatizar a geração dos modelos das construções pré-existentes.

O artigo apresenta uma metodologia para transformar o material 2D existente em modelos das edificações pertencentes a Universidade Federal de Pernambuco por meio do desenvolvimento de um processo de automatização utilizando a linguagem de programação Python. Vislumbra-se que o mesmo pode ser estendido a qualquer organização que tenha o interesse de usar seus arquivos *as-builts* para gerar modelos de suas edificações e auxiliar neste processo de gerenciamento das edificações, auxiliando em suas possíveis reformas e na manutenção das instalações. A metodologia aqui desenvolvida tem o intuito de obter um arquivo IFC (*Industry Foundation Classes*) com as informações de projeto, compatível com softwares BIM e desenvolvido a partir de imagens *rasters* 2D.

INDUSTRY FOUNDATION CLASSES - IFC

O IFC é um modelo de produto desenvolvido pela indústria, para projeto e para o ciclo de vida completo de construção [1] ou ainda, os IFCs consistem em uma biblioteca de objetos e definição de propriedades que podem ser usados para representar um empreendimento de construção e suportam a utilização dessa informação da construção para usos particulares [1]. Esse modelo de dados aberto foi elaborado na linguagem *ISO-STEP EXPRESS*, mas totalmente focados em intercâmbio entre softwares na engenharia, possibilitando a interoperabilidade eficaz no desenvolvimento BIM, permitindo trocar informações no desenvolvimento de uma edificação [5].

A *BuildingSMART* é uma organização internacional sem fins lucrativos, neutra e aberta que fornece suporte para desenvolver tecnologias que realce a indústria de projetos construtivos em geral. Esta organização é responsável por desenvolver o IFC original e suas atualizações com foco em desenvolver e melhorar a produtividade, diminuir gastos, ampliar toda a comunicação em as fases de uma obra [6][7].

O IFC funciona como um modelo de dados firmado para orientação de objetos, a qual traz informações de divisões de classes que representaram os objetos tidos em projeto, como formas de elementos, espaços, etc., que se faz necessário mediante o desenvolvimento até a conclusão processo construtivo. Assim, por exemplo, identifica-se uma parede como "*wall*" e nessa há informações contidas com suas características.

O corpo arquitetônico do IFC desenvolvido na linguagem *EXPRESS* é composta por quatro níveis: *domain*, *interoperability*, *core* e *resource*. O nível *domain* está em um patamar que permite a descrição de informações relacionadas a todas as disciplinas envolvidas em todo desenvolvimento do projeto, imaginando a divisão por fase de uma pirâmide, esse seria o topo da mesma nos processos em IFC; a *interoperability* permite o desenvolvimento da interface e câmbio de informações entre os *domain*; o *core* descreve a estrutura e partições do modelo, fornecendo as informações cruciais sobre o objeto, a funcionalidade, relação, tipo, geometria, etc.; e por fim, na base da pirâmide o nível *resource*, o qual contém as definições básicas para detalhar os objetos nos níveis superiores [8]. Com isso, é possível ter um modelo amplo e que possibilite as informações gerais de dos objetos envolvidos a modelos sofisticados, com alto grau de detalhamento.

Para entender mais um pouco sobre a constituição do IFC, é importante saber que como se organiza a linguagem *EXPRESS*, a qual foi desenvolvida para a representação de dados de acordo com entidades, atributos e relacionamentos, viabilizando o uso de medidas globais ou restrições dos mesmos [9]. É importante salientar que não se pode criar programas executáveis com o mesmo, logo não pode ser definida como uma linguagem de programação, ele apenas designa a definição de dados nos quais os programas podem operar, por isso a importância de utilizar uma linguagem de programação como o Python que possa gerar um arquivo com extensão IFC.

METODOLOGIA

Esta é uma pesquisa exploratória, onde o estudo seguiu uma abordagem em fases como acontece em diferentes campos do conhecimento como nas *Design Sciences* [10], na abordagem construtivista [11] e no modelo de racionalidade de Simon [12].

1. Identificação do problema relevante que também é um problema de pesquisa, em que foi desenvolvida uma pesquisa bibliográfica para identificar as diferentes formas de tratar o problema e quais seriam os recursos necessários para desenvolvimento da metodologia;
2. Compreensão do problema e desenvolvimento de conhecimento geral sobre o tópico através de análise de ferramentas e metodologias disponíveis;
3. Refinamento da solução usando exemplos;

4. Implantação da solução usando um Projeto Piloto que será um edifício da Universidade Federal de Pernambuco para construção dos filtros usados no desenvolvimento do modelo;
5. Análise dos resultados do Projeto Piloto e diagnóstico da pesquisa através da identificação de pontos que podem melhorar para replicar a pesquisa para os demais edifícios.

Doukari e Greenwood [2] apresentaram um método para geração automática de modelos de informações de construção a partir da digitalização de imagens do Google Earth e desenvolvimento de modelos baseados em identificação de massa. Esse método se fundamenta em leitura de imagem, conversão da escala de cores *Red, Green, Blue* (RBG) para a escala de tons de cinza, em seguida aplicação computacional do filtro Gaussiano para melhorar a qualidade da imagem obtida, recorrendo juntamente a ferramenta *Canny filter* [13] que aprimora os contornos para em seguida aplicar o *Canny Edge Detection*, podendo dessa forma detectar fragmentos da imagem e reconhecer elementos para extração, avaliação e seleção desses contornos sendo o objetivo final dessa metodologia usada como base estimar a densidade habitacional a partir de imagens de satélites, por exemplo, que não se aplica no que se almeja obter nessa pesquisa [2].

Assim, nossa metodologia trata de adaptar a metodologia proposta por Doukari e Greenwood [2], mas com um objetivo final diferente. O objetivo desta pesquisa é usar a leitura de imagem a partir de documentos 2D para a partir destes desenvolver o modelo das edificações e armazenar os dados em extensão IFC (*Industry Foundation Classes*) produzido pela linguagem de programação e compatível com os programas de tecnologia BIM. Nem todos os projetos da instituição estão em documentos CAD, pois muitos prédios são anteriores à utilização generalizada do CAD, assim, a conversão vetorial dos arquivos CAD não atenderia integralmente à necessidade da instituição, sendo necessário também converter arquivos de imagens desenvolvidas a partir dos projetos em papel da instituição.

Assim, com uma base teórica partiu-se para o aprendizado das ferramentas necessárias para o desenvolvimento do mesmo, assim como a obtenção dos melhores softwares para se trabalhar com a linguagem de programação mais indicada. A linguagem de programação Python tem sido indicada para se trabalhar com desenvolvimento de inteligência artificial, assim como, bastante simples, se comparadas a outras como as Linguagem C e C++, para iniciantes no ramo da tecnologia. Dessa forma, decidiu-se adotar o Python 3.9 para uso do ambiente de desenvolvimento integrado em programação de computadores, o PyCharm, ambos os softwares possuem licença livre [14].

O programa é possui vários repositórios e bibliotecas, que tem funções especifica para cada tipo de usuário, disponibilizados através da documentação oficial ou pela barra de tarefas do PyCharm. Em cada um dos problemas encontrados, faz-se necessário adquirir uma ou mais bibliotecas que juntas executem a pretensão desejada.

O próximo passo seguiu-se a fase atual de testes para compilar códigos unindo o código aos interesses pretendidos a partir da revisão bibliográficas feitas e dos

objetivos intrínsecos do trabalho. Usando como bases códigos livres em repositórios de compartilhamento como o GitHub para base a produção do projeto. A primeira adaptação de código foi para um algoritmo que reconhece imagens de tal forma a evidenciar os contornos da imagem em escala de cinza, usando como ferramenta principal do código a biblioteca a open-CV que tem como foco resolver métodos computacionais. Consultando o código na figura 1, ao seu lado tem o resultado final obtido a partir do mesmo, um ponto importante é o avanço contínuo no código, inicialmente usando esse programa para reconhecer imagens e torná-las um gradiente de detalhes no preto e branco, possibilitando assim trabalhar com o contorno da forma da imagem, sem as distorções da imagem em cores RGB.

Figura 1: Código de teste 01 – com resultado




```
# Testes IC

import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('C:/Users/miche/Pictures/pha.jpg',0)
edges = cv2.Canny(img,100,200)
#
plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])

plt.show()
```

Fonte: as autoras.

Com a imagem obtida anteriormente, partiu-se para aprimorar a imagem com uma escala gaussiana pelo *Canny Filter* para evidenciar seus elementos, que após tentativas e erros obteve-se a segunda fase de teste do projeto, com objetivo de aplicar a ferramenta *Canny Filter* a imagem escolhida usando as bibliotecas o pacote *Numpy*, que contém bibliotecas como *Matplotlib*, para leitura e geração de dados com grandes números de detalhes. O programa pode ser visto nas figuras 2 e 3, a qual a última mostra o resultado obtido ao final.

Figura 2: Código de teste 02 – parte 1

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage as ndi

from skimage import feature

# Generate noisy image of a square
#im = np.zeros((200, 200))
#im[32:-32, 32:-32] = 1

im = cv2.imread('C:/Users/miche/Pictures/pha.jpg', 0)
# im[32:-32, 32:-32] = 1

im = ndi.rotate(im, 0, mode='constant')
#im = ndi.gaussian_filter(im, 4)
#im += 0.2 * np.random.random(im.shape)

# Compute the Canny filter for two values of sigma
edges1 = feature.canny(im)
edges2 = feature.canny(im, sigma=3)
edges3 = feature.canny(im, sigma=4)
```

Fonte: as autoras.

Figura 3: Código de teste 02 – parte 2 com resultado

```
# display results
fig, (ax1, ax3, ax4) = plt.subplots(nrows=1, ncols=3, figsize=(8, 3), sharex=True, sharey=True)

ax1.imshow(im, cmap=plt.cm.gray)
ax1.axis('off')
ax1.set_title('Noisy Image', fontsize=20)

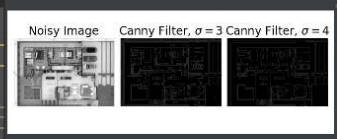
#ax2.imshow(edges1, cmap=plt.cm.gray)
#ax2.axis('off')
#ax2.set_title(r'Canny Filter, $\sigma=1$', fontsize=20)

ax3.imshow(edges2, cmap=plt.cm.gray)
ax3.axis('off')
ax3.set_title(r'Canny Filter, $\sigma=3$', fontsize=20)

ax4.imshow(edges3, cmap=plt.cm.gray)
ax4.axis('off')
ax4.set_title(r'Canny Filter, $\sigma=4$', fontsize=20)

fig.tight_layout()

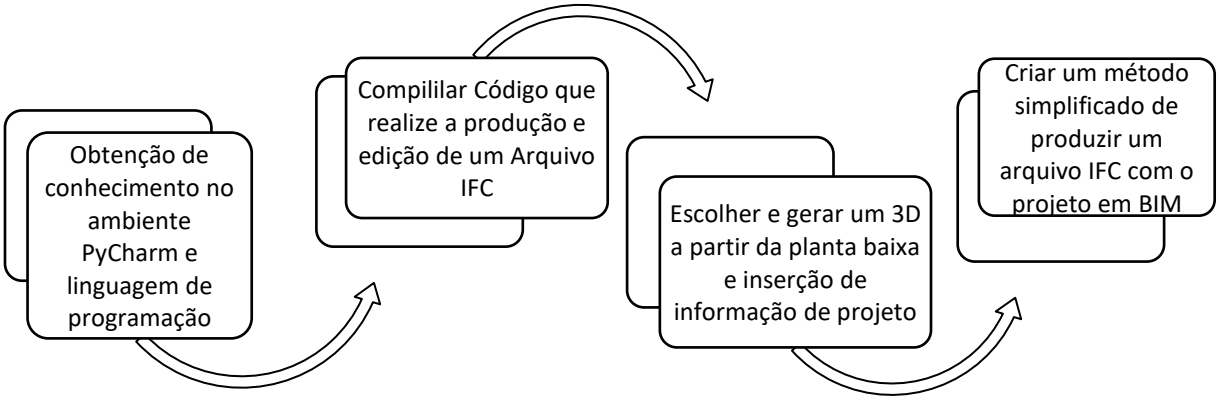
plt.show()
```



Fonte: as autoras.

Assim, o fluxograma a seguir (figura 4), de forma resumida, descreve a soma de problemas e resultados que se somam para obter o objetivo final do trabalho.

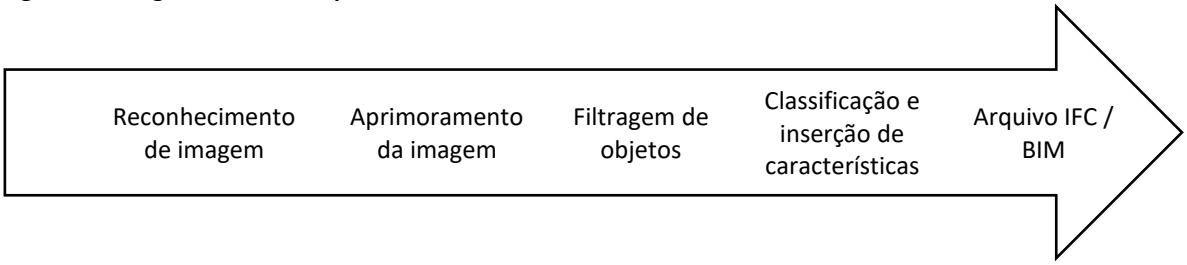
Figura 4: Código de teste 02 – parte 1



Fonte: as autoras.

Deste modo, continua-se com o desenvolvimento do algoritmo para continuidade do fluxo sendo ajustado o seu desenvolvimento para aprimorar o reconhecimento de imagens, filtragem e classificação dos objetos, conforme fluxo apresentado na Figura 5. Com estes refinamentos, é possível aprimorar a imagem e melhorar a sensibilidade para a filtragem de objetos e sua criação no modelo através do uso de informações associadas às informações geométricas extraídas das figuras. Diante dessas informações é desenvolvido cada objeto em IFC. Assim, a ideia é que de posse da planta baixa, transformá-la em imagem, seja possível reproduzir o projeto em um modelo em formato IFC para que ele seja aberto no Revit, por exemplo, de maneira automática, incorporando rapidez e tecnologia ao processo de transformação de projetos em 2D para modelos BIM a partir do uso de algoritmos, já que não é possível atualmente abrir um arquivo CAD e 2D em BIM com a geração de uma realidade 3D que é fundamental para a utilização dos recursos das ferramentas BIM.

Figura 5: Código de teste 02 – parte 1



Fonte: as autoras.

RESULTADOS E DISCUSSÕES

Após identificar um método eficaz em situação semelhantes, a estratégia de estudar uma linguagem de programação mais acessível e versátil que possibilita aos usuários com conhecimento mínimo de programação conseguir desfrutar dos futuros resultados.

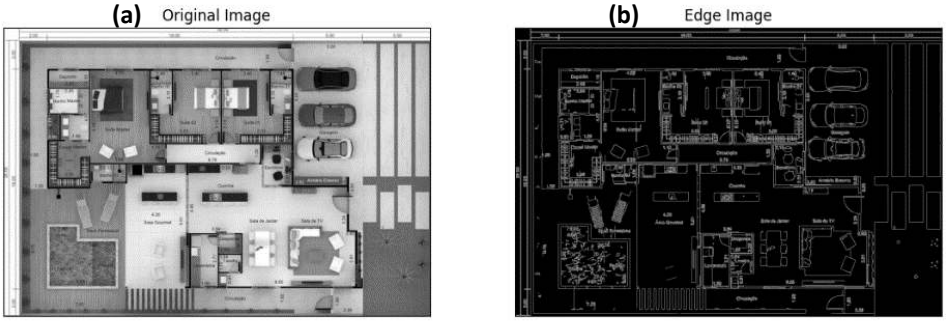
Inicialmente foi dado foco em obter a imagem em escala de cinza, a partir de uma planta baixa encontrada no site Pinterest em forma de imagem no sistema RGB de cores, como pode ser visto na figura 6. Analisando o nível de detalhamento que se tem nessa imagem, como pode ser constatado na figura 8 a eficácia em utilizar o programa, o qual apresenta fragmentos de contorno dos objetos mostrados na imagem original como forma de redesenhar figuras que estão em blocos, simplificando a imagem para a escala de preto e branco, possibilitando assim reutilizar as informações e futuramente gerar novos projetos. A imagem JPG que deu origem ao que se pretendia foi retirada da internet, com a intenção de realizar os testes, verificando assim se estava tendo um funcionamento ideal do programa, a qual veio colorida e com grande densidade de informação a serem lidas e contornadas pelo desenvolvedor.

Figura 6: Imagem gerada pelo desenvolvedor



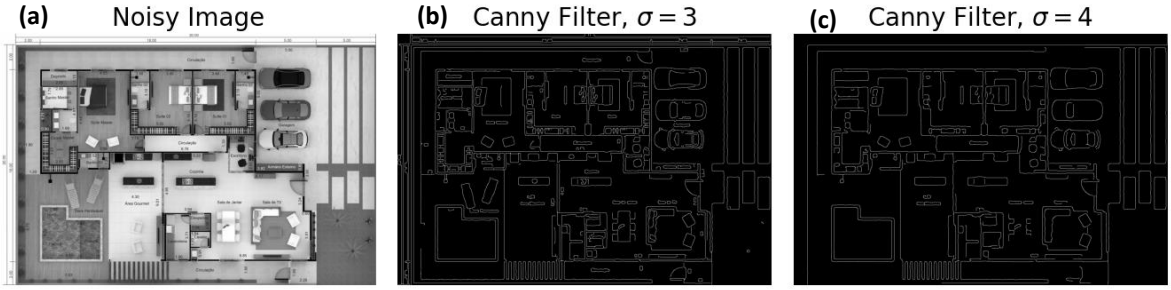
Fonte: https://www.pinterest.es/pin/49539664643222674/ .

Figura 7: Imagem gerada pelo desenvolvedor



Fonte: as autoras.

Figura 8: Imagem gerada pelo desenvolvedor aplicando o Canny Filter

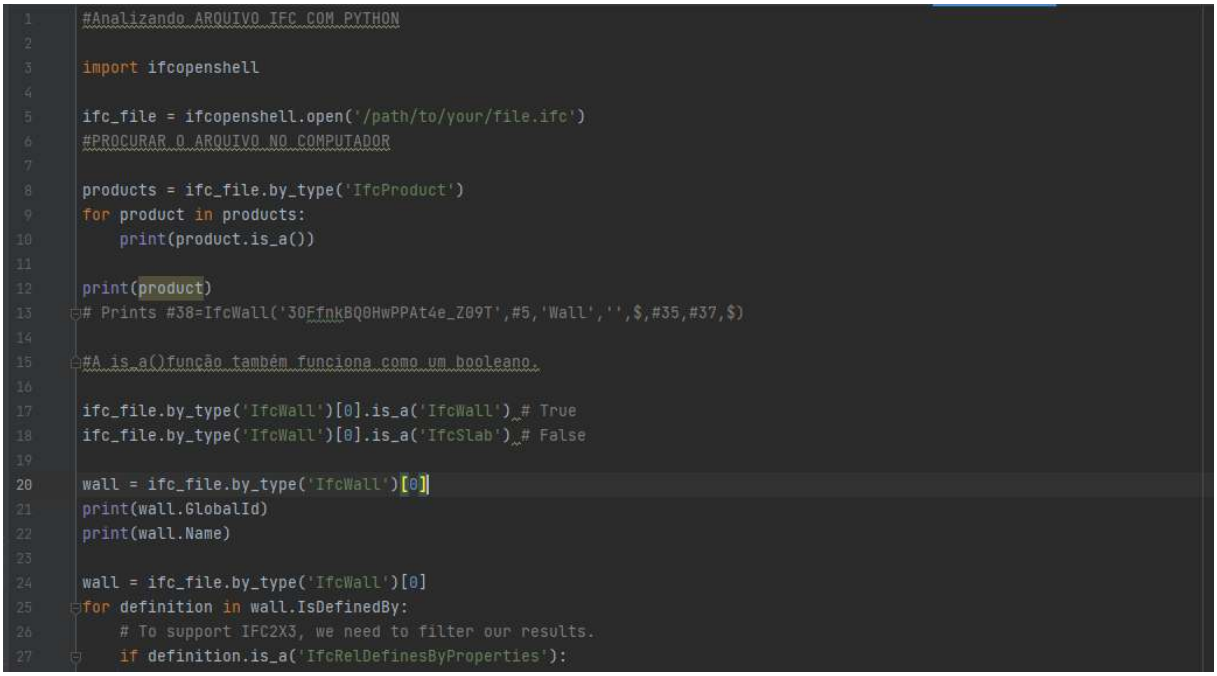


Fonte: as autoras.

Outro ponto importante, deu-se em implementar a ferramenta *Canny Filter* no Python, onde se consegue selecionar os objetos mais significativos para construção em meio a tantos detalhes quanto essa foto possui, para na próxima fase de identificação de objeto, seja possível que o próprio Python consiga filtrar os elementos como itens separados da imagem. O σ atribuído, melhora o filtro e consequentemente a qualidade da imagem. Por exemplo, a imagem apresentada na figura 7 (b) tem um $\sigma = 1$, onde podemos ver emaranhado de informação, na Figura 8 (b) e 8 (c) temos valores de $\sigma = 3$, e $\sigma = 4$, respectivamente, onde é possível que aumenta a distinção entre os objetos, reduzindo os espaços onde não é possível determinar que tipo de objeto está sendo analisado.

Em seguida, foi verificado a compatibilidade entre a linguagem de programação usada e o arquivo que se tem a intenção de gerar (figuras 9, 10 e 11), um arquivo IFC, com as informações que devem ser retiradas da planta baixa fornecida, como no ambiente consegue-se abrir arquivos IFC e até mesmo ler mostrando informação como das paredes e portas existentes no arquivo, o qual estava já previamente pronto apenas para ser usando como teste.

Figura 9: Imagem gerada através do uso do IFC Open Shell – Parte 1



```
#Analizando ARQUIVO IFC COM PYTHON

import ifcopenshell

ifc_file = ifcopenshell.open('/path/to/your/file.ifc')
#PROCURAR O ARQUIVO NO COMPUTADOR

products = ifc_file.by_type('IfcProduct')
for product in products:
    print(product.is_a())

print(product)
# Prints #38=IfcWall('3OFfnkBQ0HwPPAt4e_Z09T',#5,'Wall','',$,#35,#37,$)

#A is_a()função também funciona como um booleano.

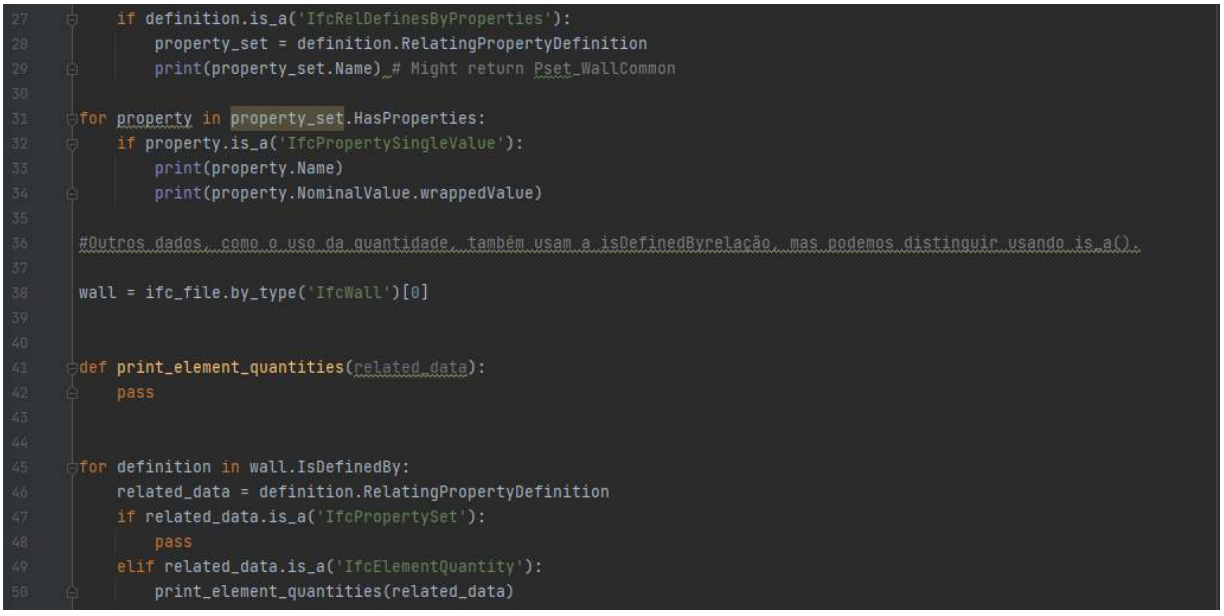
ifc_file.by_type('IfcWall')[0].is_a('IfcWall') # True
ifc_file.by_type('IfcWall')[0].is_a('IfcSlab') # False

wall = ifc_file.by_type('IfcWall')[0]
print(wall.GlobalId)
print(wall.Name)

wall = ifc_file.by_type('IfcWall')[0]
for definition in wall.IsDefinedBy:
    # To support IFC2X3, we need to filter our results.
    if definition.is_a('IfcRelDefinesByProperties'):
```

Fonte: as autoras.

Figura 10: Imagem gerada através do uso do IFC Open Shell – Parte 2



```
    if definition.is_a('IfcRelDefinesByProperties'):
        property_set = definition.RelatingPropertyDefinition
        print(property_set.Name) # Might return Pset_WallCommon

for property in property_set.HasProperties:
    if property.is_a('IfcPropertySingleValue'):
        print(property.Name)
        print(property.NominalValue.wrappedValue)

#Outros dados, como o uso da quantidade, também usam a isDefinedByrelação, mas podemos distinguir usando is_a().

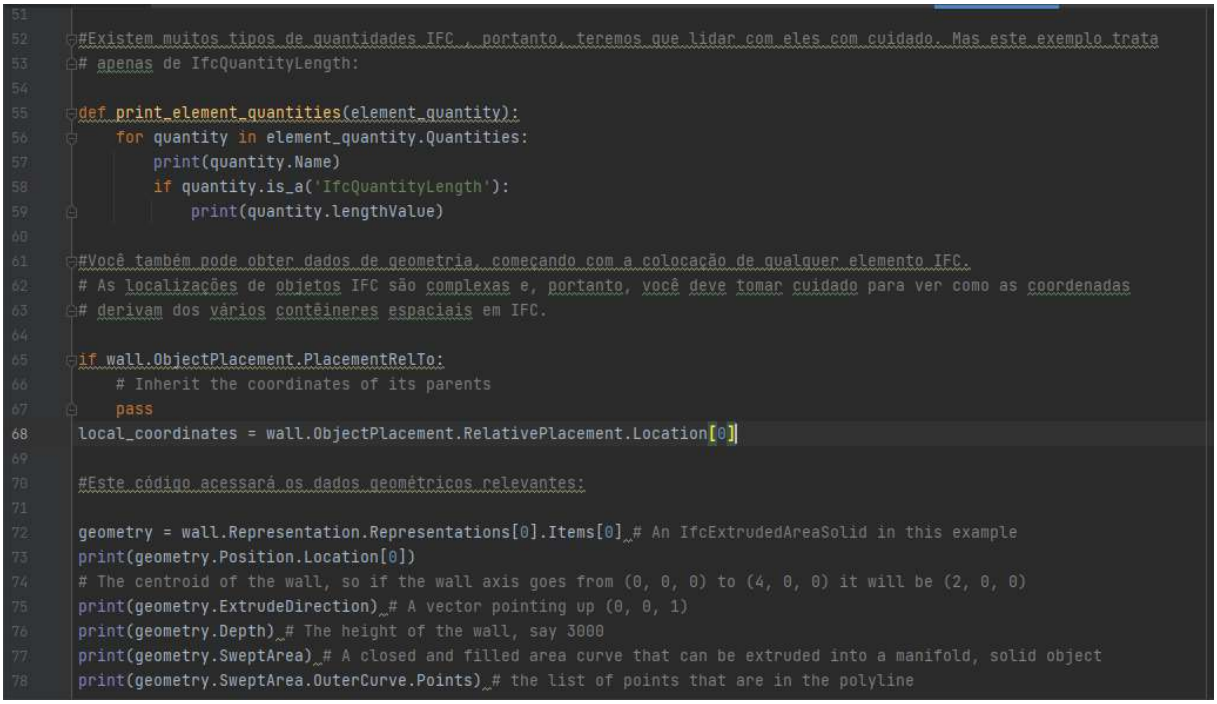
wall = ifc_file.by_type('IfcWall')[0]


def print_element_quantities(related_data):
    pass


for definition in wall.IsDefinedBy:
    related_data = definition.RelatingPropertyDefinition
    if related_data.is_a('IfcPropertySet'):
        pass
    elif related_data.is_a('IfcElementQuantity'):
        print_element_quantities(related_data)
```

Fonte: as autoras.

Figura 11: Imagem gerada através do uso do IFC Open Shell – Parte 3



```
#Existem muitos tipos de quantidades IFC , portanto, teremos que lidar com eles com cuidado. Mas este exemplo trata
# apenas de IfcQuantityLength:

def print_element_quantities(element_quantity):
    for quantity in element_quantity.Quantities:
        print(quantity.Name)
        if quantity.is_a('IfcQuantityLength'):
            print(quantity.lengthValue)

#Você também pode obter dados de geometria, começando com a colocação de qualquer elemento IFC.
# As localizações de objetos IFC são complexas e, portanto, você deve tomar cuidado para ver como as coordenadas
# derivam dos vários contêineres espaciais em IFC.

if wall.ObjectPlacement.PlacementRelTo:
    # Inherit the coordinates of its parents
    pass
local_coordinates = wall.ObjectPlacement.RelativePlacement.Location[0]

#Este código acessará os dados geométricos relevantes:

geometry = wall.Representation.Representations[0].Items[0]# An IfcExtrudedAreaSolid in this example
print(geometry.Position.Location[0])
# The centroid of the wall, so if the wall axis goes from (0, 0, 0) to (4, 0, 0) it will be (2, 0, 0)
print(geometry.ExtrudeDirection)# A vector pointing up (0, 0, 1)
print(geometry.Depth)# The height of the wall, say 3000
print(geometry.SweptArea)# A closed and filled area curve that can be extruded into a manifold, solid object
print(geometry.SweptArea.OuterCurve.Points)# the list of points that are in the polyline
```

Fonte: as autoras.

Para o uso desse recurso visto nas imagens 10, 11 e 12, foi indispensável uma ferramenta extra de plugin no PyCharm, o *IFC Open Shell*, que tem como umas das funções ler, abrir e escrever arquivos nesse formato, também obtido gratuitamente, com a obtenção do mesmo chegou em resultados esperados, na leitura e abertura dos arquivos [15].

Esta é uma pesquisa em desenvolvimento, de modo que se espera, conciliar todas essas ferramentas em um único código associada à um sistema de decisão baseado em regras “se-então”, que permita desenvolver essa análise simultânea e que os estudos de abertura desse tipo de arquivo, assim como a inserção de informações cruciais, sejam viáveis. A identificação de objetos usando as ferramentas mostradas, por mais que a linguagem usada seja fácil em comparações as demais, tem-se a carência de falta de documentação para implementação de novos recursos sobre o software, logo é uma das maiores dificuldade encontrada neste estudo, uma vez que é preciso identificar o tipo de objeto e associá-lo às dimensões e características do tipo de objeto (tais como laje, de viga, janela) de acordo com aquele preconizado no IFC.

CONCLUSÕES

Dado a grande importância da implementação do BIM em todos os setores, tem-se assim que automatizar os processos de geração de *As builts* das edificações já existentes dentro da Universidade Federal de Pernambuco e de outros órgãos e empresas, que com o início da implantação da Estratégia Bim BR e sem infraestruturas suficientes dentro dos órgãos públicos, por exemplo, simplificar o processo evitará grande retrabalho com parte da equipe técnica. Com a presente pesquisa e sua revisão bibliográfica concluiu-se que é possível a obtenção de outros métodos por

mecanismos computacionais para automatização de projetos 2D de obras já existentes para 3D. Assim como, com a utilização da linguagem de programação, Python, é evidente que o método pode ser obtido. Até então, o trabalho apresentou outros resultados com relação ao próprio ambiente de desenvolvimento, que com aplicação da metodologia, foi-se capaz de implantar a geração de imagens em preto e branco e uma gaussiana simplificada com o *Canny filter* para melhorar a imagem. Evidenciando ainda que a abertura e leitura de arquivos IFC também foi implementada. Deixando como sugestão para trabalho futuros a unificação dos códigos, para o melhor gerenciamento do projeto, e o desenvolvimento de funcionalidades que abra um arquivo IFC, anexe a imagem requerida e logo em seguida com a filtragem de fragmentos, seja possível identificá-los e adicionar informações.

REFERÊNCIAS

[1] SACKS, Rafael *et al.* **BIM Handbook - A Guide to Building Information Modeling for Owners, Designers, Engineers, Contractors, and Facility Managers**. 3ª Editioned. [*S. l.: s. n.*], 2018.

[2] DOUKARI, Omar; GREENWOOD, David. Automatic generation of building information models from digitized plans. **Automation in Construction**, [*s. l.*], v. 113, n. December 2019, p. 103129, 2020. DOI: https://doi.org/10.1016/j.autcon.2020.103129.

[3] MDIC. **Estratégia Nacional de Disseminação do Building Information Modelling - BIM**. [*s. l.*], p. 1–33, 2018. Disponível em: http://www.mdic.gov.br/images/REPOSITORIO/sdci/CGMO/26-11-2018-estrategia-BIM-BR-2.pdf.

[4] UK BIM ALLIANCE. Information management according to BS EN ISO 19650 - Guidance Part 2: Processes for Project Delivery. **UK BIM Alliance**, [*s. l.*], p. 1–159, 2020. Disponível em: https://www.ukbimalliance.org/stories/information-management-according-to-bs-en-iso-19650/.

[5] EASTMAN, Charles M. *et al.* Exchange Model and Exchange Object Concepts for Implementation of National BIM Standards. **Journal of Computing in Civil Engineering**, [*s. l.*], v. 24, n. 1, p. 25–34, 2010. DOI: https://doi.org/10.1061/(ASCE)0887-3801(2010)24:1(25).

[6] BUILDINGSMART. [*S. l.*], [*s. d.*]. Disponível em: https://www.buildingsmart.org/. Acesso em: 18 jun. 2021.

[7] TEO, Tee Ann; CHO, Kuan Hsun. BIM-oriented indoor network model for indoor and outdoor combined route planning. **Advanced Engineering Informatics**, [*s. l.*], v. 30, n. 3, p. 268–282, 2016. DOI: https://doi.org/10.1016/j.aei.2016.04.007.

[8] AYRES FILHO, Cervantes; SCHEER, Sérgio. Accessing ifc model through compiled classes and suggestions for creating higher-level access classes. **Gestão & Tecnologia de Projetos**, [*s. l.*], v. 4, n. 2, p. 112–138, 2009. DOI: https://doi.org/10.4237/gtp.v4i2.112.

[9] FOWLER, Julian. **STEP for data management, exchange and sharing**. [*s. l.*], p. 226, 1995. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.2051&rep=rep1&type=pdf.

[10] AKEN, Joan E. van. Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-Tested and Grounded Technological Rules. **Journal of Management Studies**, [*s. l.*], v. 41, n. 2, p. 219–246, 2004. DOI: https://doi.org/10.1111/j.1467-6486.2004.00430.x.

[11] BERKES, Fikret; DAVIDSON-HUNT, Iain J. Communities and social enterprises in the age of globalization. **Journal of Enterprising Communities: People and Places in the Global Economy**, [*s. l.*], v. 1, n. 3, p. 209–221, 2007. DOI: https://doi.org/10.1108/17506200710779521.

[12] SIMON, Herbert A. **Models of Bounded Rationality**. Economic Aed. Cambridge, MA: MIT Press, 1982.

[13] CANNY, John. A Computational Approach to Edge Detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [*s. l.*], v. PAMI-8, n. 6, p. 679–698, 1986. DOI: https://doi.org/10.1109/TPAMI.1986.4767851.

[14] SILVA, Igor Rodrigues Sousa; SILVA, Rogério Oliveira da. Linguagem De Programação Python Python Programming Language. **Revista Tecnologias em Projeção**, [*s. l.*], v. 10, n. 1, p. 55–71, 2019.

[15] IFC OPEN SHELL. **IFC Open Shell**. [*S. l.*], [*s. d.*]. Disponível em: http://ifcopenshell.org/. Acesso em: 7 mar. 2021.