# Введение в фотограмметрию
# Построение карт глубины
# Метод **Patch Match**

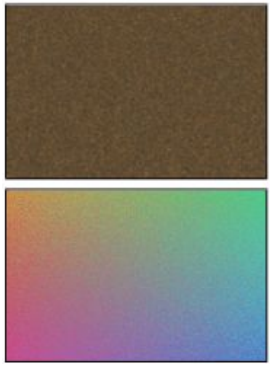
Фотограмметрия. Лекция 13

- **Глубины+Нормали (Patch Match)**
- **Gipuma**
- **Colmap**
- **ACMH/ACMM**

Полярный Николай
polarnick239@gmail.com

# Построение карты глубины методом Patch Match

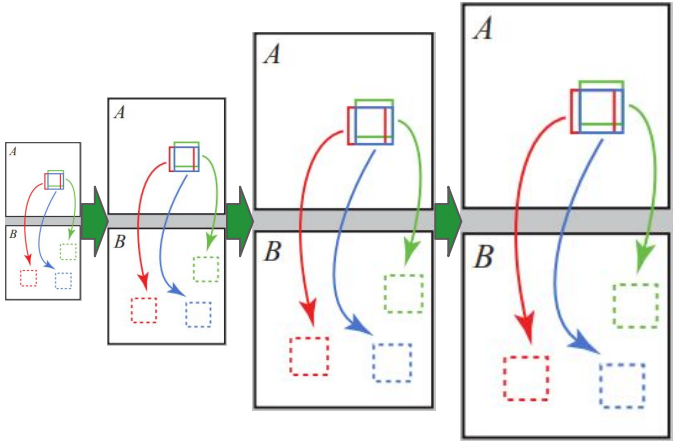
depth_map, normal_map = random()



# Построение карты глубины методом Patch Match

depth_map, normal_map = random()

for level = 0 … N: (Coarse to Fine)

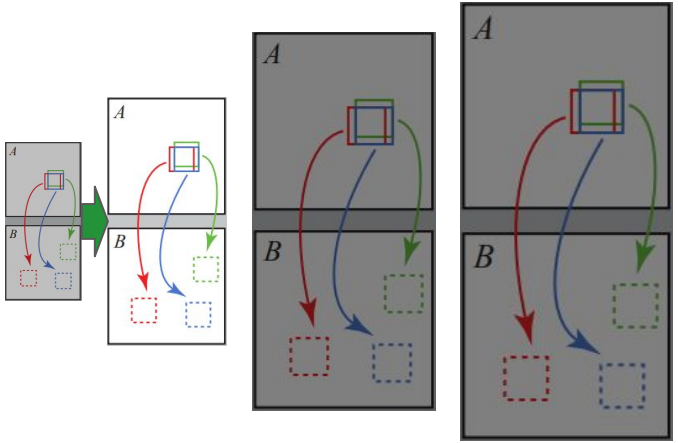


# Построение карты глубины методом Patch Match

depth_map, normal_map = random()

for level = 0 … N:

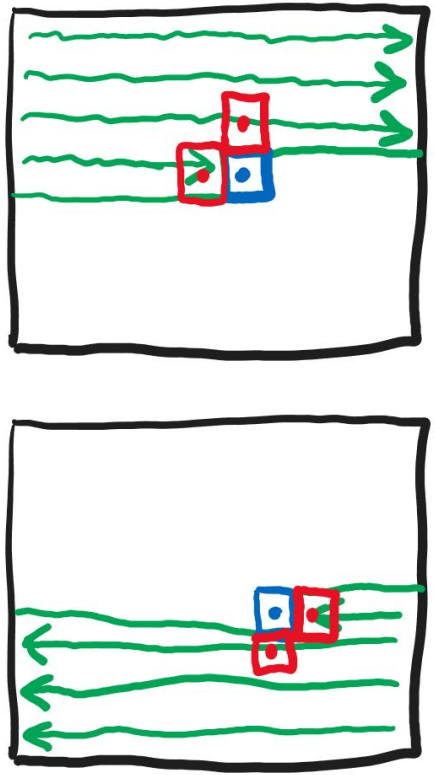
    upscale(depth_map, normal_map)

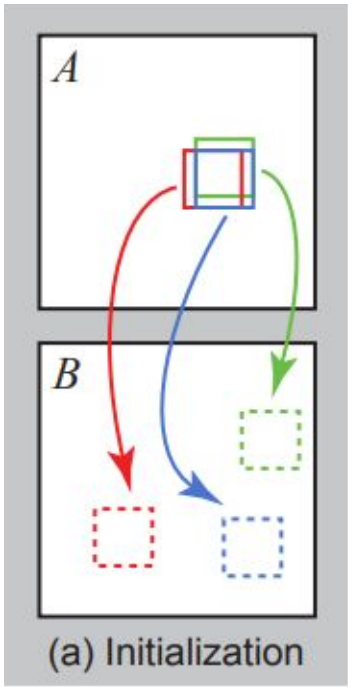


# Построение карты глубины методом Patch Match

```
depth_map, normal_map = random()

for level = 0 … N:
    upscale(depth_map, normal_map)
    for iteration = 0 … 100:
        propagation()
```

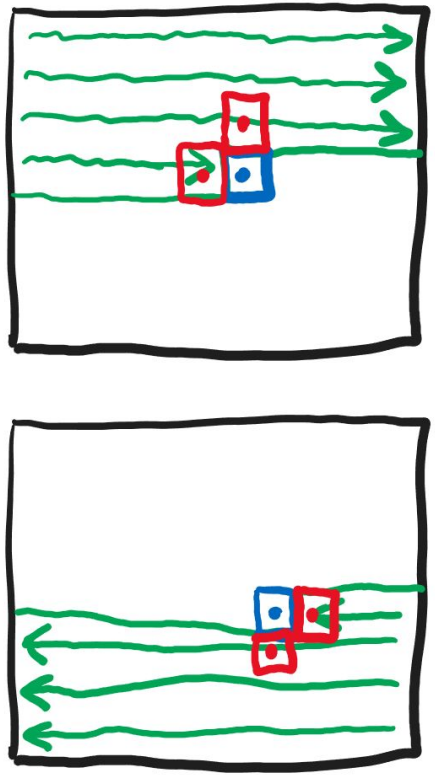


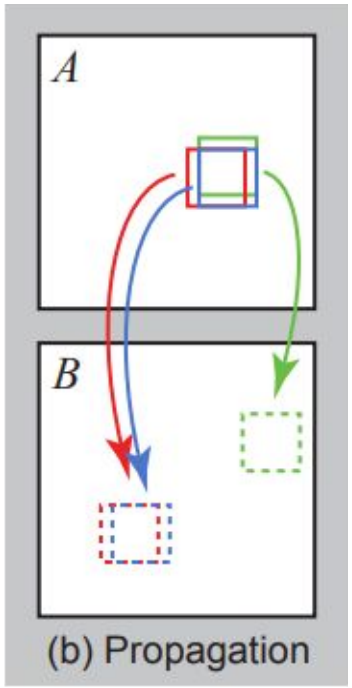


# Построение карты глубины методом Patch Match

```
depth_map, normal_map = random()

for level = 0 … N:
    upscale(depth_map, normal_map)
    for iteration = 0 … 100:
        propagation()
```

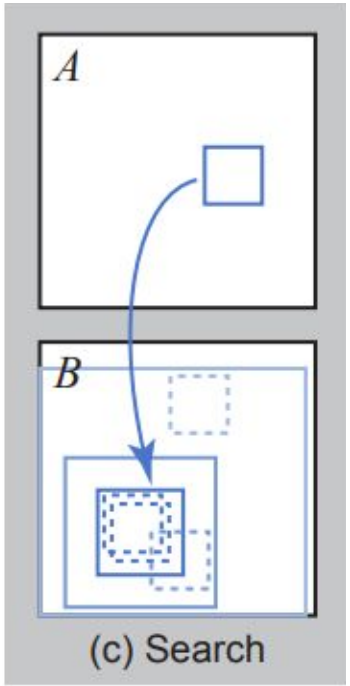




# Построение карты глубины методом Patch Match

```
depth_map, normal_map = random()

for level = 0 … N:
    upscale(depth_map, normal_map)
    for iteration = 0 … 100:
        propagation()
        refinement()
```



# Построение карты глубины методом Patch Match

```
depth_map, normal_map = random()

for level = 0 … N:
    upscale(depth_map, normal_map)
    for iteration = 0 … 100:
        propagation()
        refinement()

return depth_map, normal_map
```

# Построение карты глубины методом Patch Match

Базовые кирпичики:

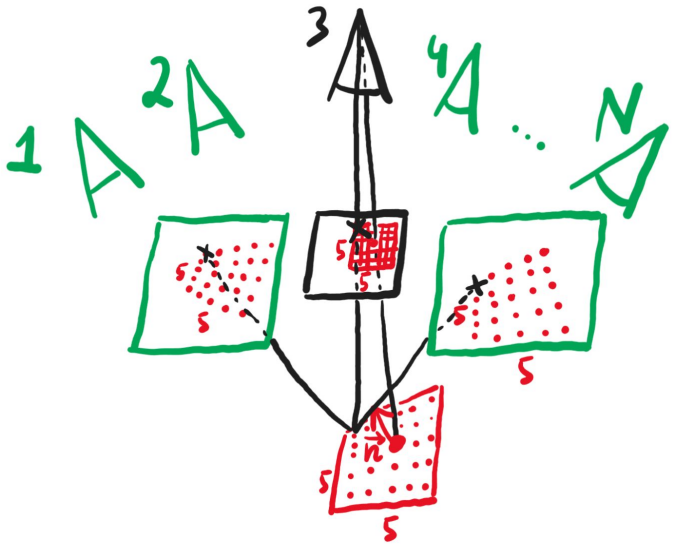
1) Когда мы примеряем на себя **depth + normal** гипотезу…

# Построение карты глубины методом Patch Match

Базовые кирпичики:

1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам.

$$cost_{ref}(i, j, d, \vec{n}) = \frac{\sum_{j \in N_{ref}} ZNCC_{ref \to j}(i, j, d, \vec{n})}{|N_{ref}|}$$

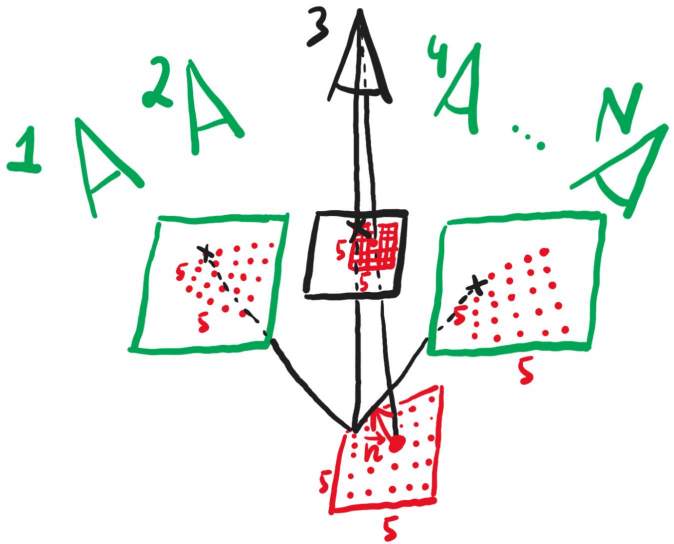


# Построение карты глубины методом Patch Match

Базовые кирпичики:

1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам. Если эта оценка **лучше** чем то что до сих пор у нас было - заменяем нашу гипотезу на то что только что примерили.

$$cost_{ref}(i, j, d, \vec{n}) = \frac{\sum_{j \in N_{ref}} ZNCC_{ref \to j}(i, j, d, \vec{n})}{|N_{ref}|}$$



# Построение карты глубины методом Patch Match

Базовые кирпичики:

1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам. Если эта оценка **лучше** чем то что до сих пор у нас было - заменяем нашу гипотезу на то что только что примерили.

2) Оценка **cost** сводится к нескольким базовым функциям:

- XYZ = camera.unproject(i, j, d)

# Построение карты глубины методом Patch Match

Базовые кирпичики:

1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам. Если эта оценка **лучше** чем то что до сих пор у нас было - заменяем нашу гипотезу на то что только что примерили.

2) Оценка **cost** сводится к нескольким базовым функциям:

- XYZ = camera.unproject(i, j, d)
- i, j = camera.project(XYZ)

# Построение карты глубины методом Patch Match

Базовые кирпичики:

1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам. Если эта оценка **лучше** чем то что до сих пор у нас было - заменяем нашу гипотезу на то что только что примерили.

2) Оценка **cost** сводится к нескольким базовым функциям:

- XYZ = camera.unproject(i, j, d)
- i, j = camera.project(XYZ)
- XYZ = intersectPlane(ray, Plane(XYZ, normal))

# Gipuma

## Massively Parallel Multiview Stereopsis by Surface Normal Diffusion

Silvano Galliani Katrin Lasinger Konrad Schindler

Photogrammetry and Remote Sensing, ETH Zurich

### Abstract

*We present a new, massively parallel method for high-quality multiview matching. Our work builds on the* Patch-match *idea: starting from randomly generated 3D planes in scene space, the best-fitting planes are iteratively propagated and refined to obtain a 3D depth and normal field per view, such that a robust photo-consistency measure over all images is maximized. Our main novelties are on the one hand to formulate Patchmatch in scene space, which makes it possible to aggregate image similarity across multiple views and obtain more accurate depth maps. And on the other hand a modified, diffusion-like propagation scheme that can be massively parallelized and delivers dense multiview correspondence over ten 1.9-Megapixel images in 3 seconds, on a consumer-grade GPU. Our method uses a slanted support window and thus has no fronto-parallel bias; it is completely local and parallel, such that computation time scales linearly with image size, and inversely proportional to the number of parallel threads. Further-*

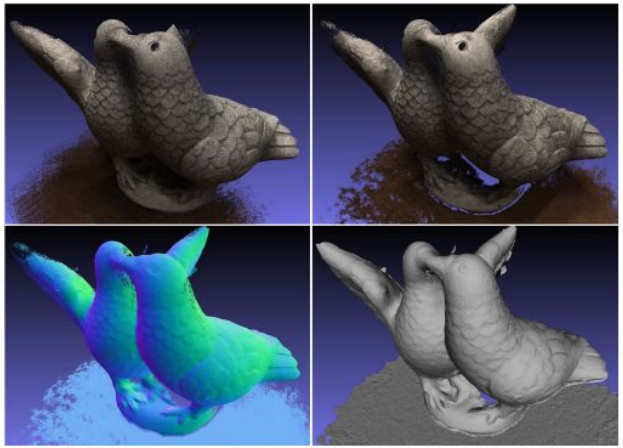
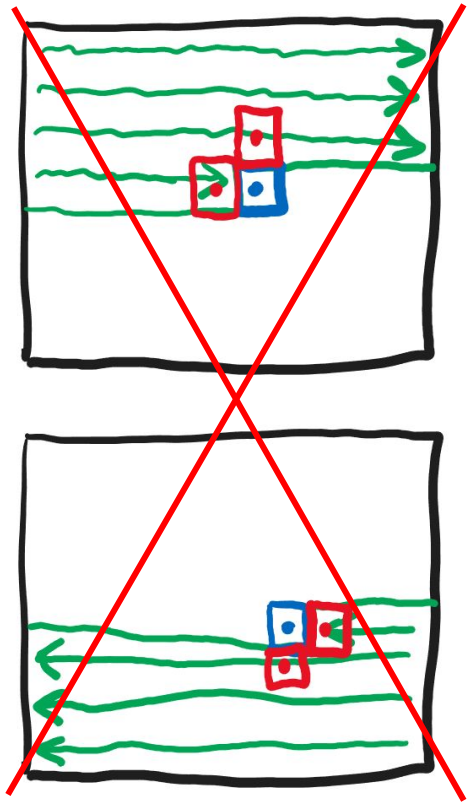


Figure 1: Results on one of the 80 evaluated objects on the DTU benchmark [22]. *Top left*: Ground truth point cloud; *top right*: reconstructed point cloud with texture; *bottom left*: color-coded surface normals; *bottom right*: reconstructed surface.

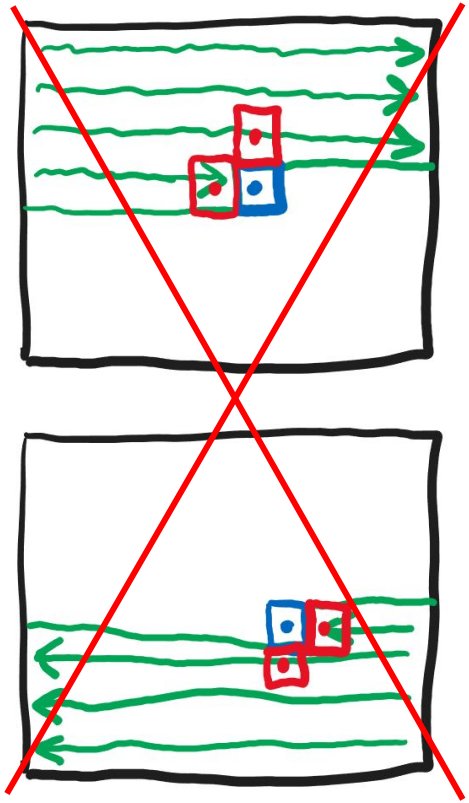
# Gipuma

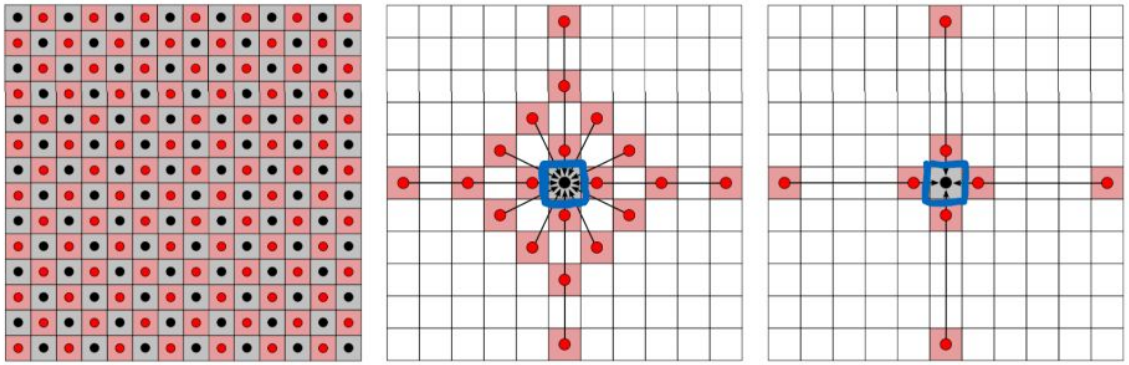
1) **propagation**: мы хотим массовый параллелизм (многоядерные **CPU/GPU**)



# Gipuma

1) **propagation**: мы хотим массовый параллелизм (многоядерные **CPU/GPU**)





(a) (b) (c)

Figure 2: The propagation scheme: (a) Depth and normal are updated in parallel for all red pixels, using black pixels as candidates, and vice versa. (b) Planes from a local neighborhood (red points) serve as candidates to update a given pixel (black). (c) Modified scheme for speed setting, using only inner and outermost pixels of the pattern.

Gipuma

$\mathbf{q}'_{\pi_\mathbf{p}}$ in the other image. Then the matching cost is

$$m(\mathbf{p}, \pi_\mathbf{p}) = \sum_{\mathbf{q} \in W_\mathbf{p}} w(\mathbf{p}, \mathbf{q})\, \rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}). \tag{1}$$

The weight function $w(\mathbf{p}, \mathbf{q}) = e^{-\frac{\|I_\mathbf{p} - I_\mathbf{q}\|}{\gamma}}$ can be seen as a soft segmentation, which decreases the influence of pixels that differ a lot from the central one. We use a fixed setting $\gamma = 10$ in all experiments.

The cost function $\rho$ consists of a weighted combination of absolute color differences and differences in gradient magnitude. More formally, for pixels $\mathbf{q}$ and $\mathbf{q}'_{\pi_\mathbf{p}}$ with colors $I_\mathbf{q}$ and $I_{\mathbf{q}'_{\pi_\mathbf{p}}}$

$$\begin{aligned} \rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}) = (1-\alpha) \cdot \min(\|I_\mathbf{q} - I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{col}) \\ + \alpha \cdot \min(\|\nabla I_\mathbf{q} - \nabla I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{grad}) \ , \end{aligned} \tag{2}$$

where $\alpha$ balances the contribution of the two terms and $\tau_{col}$ and $\tau_{grad}$ are truncation thresholds to robustify the cost against outliers. In all our experiments we set $\alpha = 0.9$, $\tau_{col} = 10$ and $\tau_{grad} = 2$.

Gipuma

$\mathbf{q}'_{\pi_\mathbf{p}}$ in the other image. Then the matching cost is

$$m(\mathbf{p}, \pi_\mathbf{p}) = \sum_{\mathbf{q} \in W_\mathbf{p}} w(\mathbf{p}, \mathbf{q})\, \rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}). \tag{1}$$

The weight function $w(\mathbf{p}, \mathbf{q}) = e^{-\frac{\|I_\mathbf{p} - I_\mathbf{q}\|}{\gamma}}$ can be seen as a soft segmentation, which decreases the influence of pixels that differ a lot from the central one. We use a fixed setting $\gamma = 10$ in all experiments.

The cost function $\rho$ consists of a weighted combination of absolute color differences and differences in gradient magnitude. More formally, for pixels $\mathbf{q}$ and $\mathbf{q}'_{\pi_\mathbf{p}}$ with colors $I_\mathbf{q}$ and $I_{\mathbf{q}'_{\pi_\mathbf{p}}}$

$$\begin{aligned} \rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}) = (1-\alpha) \cdot \min(\|I_\mathbf{q} - I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{col}) \\ + \alpha \cdot \min(\|\nabla I_\mathbf{q} - \nabla I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{grad}) \,, \end{aligned} \tag{2}$$

where $\alpha$ balances the contribution of the two terms and $\tau_{col}$ and $\tau_{grad}$ are truncation thresholds to robustify the cost against outliers. In all our experiments we set $\alpha = 0.9$, $\tau_{col} = 10$ and $\tau_{grad} = 2$.

Gipuma

$\mathbf{q}'_{\pi_\mathbf{p}}$ in the other image. Then the matching cost is

$$m(\mathbf{p}, \pi_\mathbf{p}) = \sum_{\mathbf{q} \in W_\mathbf{p}} w(\mathbf{p}, \mathbf{q})\, \rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}). \quad (1)$$

The weight function $w(\mathbf{p}, \mathbf{q}) = e^{-\frac{\|I_\mathbf{p} - I_\mathbf{q}\|}{\gamma}}$ can be seen as a soft segmentation, which decreases the influence of pixels that differ a lot from the central one. We use a fixed setting $\gamma = 10$ in all experiments.

The cost function $\rho$ consists of a weighted combination of absolute color differences and differences in gradient magnitude. More formally, for pixels $\mathbf{q}$ and $\mathbf{q}'_{\pi_\mathbf{p}}$ with colors $I_\mathbf{q}$ and $I_{\mathbf{q}'_{\pi_\mathbf{p}}}$

$$\rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}) = (1-\alpha) \cdot \min(\|I_\mathbf{q} - I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{col}) + \alpha \cdot \min(\|\nabla I_\mathbf{q} - \nabla I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{grad}), \quad (2)$$

where $\alpha$ balances the contribution of the two terms and $\tau_{col}$ and $\tau_{grad}$ are truncation thresholds to robustify the cost against outliers. In all our experiments we set $\alpha = 0.9$, $\tau_{col} = 10$ and $\tau_{grad} = 2$.

Gipuma

$\mathbf{q}'_{\pi_\mathbf{p}}$ in the other image. Then the matching cost is

$$m(\mathbf{p}, \pi_\mathbf{p}) = \sum_{\mathbf{q} \in W_\mathbf{p}} w(\mathbf{p}, \mathbf{q})\, \rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}). \quad (1)$$

The weight function $w(\mathbf{p}, \mathbf{q}) = e^{-\frac{\|I_\mathbf{p} - I_\mathbf{q}\|}{\gamma}}$ can be seen as a soft segmentation, which decreases the influence of pixels that differ a lot from the central one. We use a fixed setting $\gamma = 10$ in all experiments.

The cost function $\rho$ consists of a weighted combination of absolute color differences and differences in gradient magnitude. More formally, for pixels $\mathbf{q}$ and $\mathbf{q}'_{\pi_\mathbf{p}}$ with colors $I_\mathbf{q}$ and $I_{\mathbf{q}'_{\pi_\mathbf{p}}}$

$$\rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}) = (1-\alpha) \cdot \min(\|I_\mathbf{q} - I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{col}) + \alpha \cdot \min(\|\nabla I_\mathbf{q} - \nabla I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{grad}), \quad (2)$$

where $\alpha$ balances the contribution of the two terms and $\tau_{col}$ and $\tau_{grad}$ are truncation thresholds to robustify the cost against outliers. In all our experiments we set $\alpha = 0.9$, $\tau_{col} = 10$ and $\tau_{grad} = 2$.

Gipuma

$\mathbf{q}'_{\pi_\mathbf{p}}$ in the other image. Then the matching cost is

$$m(\mathbf{p}, \pi_\mathbf{p}) = \sum_{\mathbf{q} \in W_\mathbf{p}} w(\mathbf{p}, \mathbf{q})\, \rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}). \tag{1}$$

The weight function $w(\mathbf{p}, \mathbf{q}) = e^{-\frac{\|I_\mathbf{p} - I_\mathbf{q}\|}{\gamma}}$ can be seen as a soft segmentation, which decreases the influence of pixels that differ a lot from the central one. We use a fixed setting $\gamma = 10$ in all experiments.

The cost function $\rho$ consists of a weighted combination of absolute color differences and differences in gradient magnitude. More formally, for pixels $\mathbf{q}$ and $\mathbf{q}'_{\pi_\mathbf{p}}$ with colors $I_\mathbf{q}$ and $I_{\mathbf{q}'_{\pi_\mathbf{p}}}$

$$\begin{aligned} \rho(\mathbf{q}, \mathbf{q}'_{\pi_\mathbf{p}}) = &(1-\alpha) \cdot \min(\|I_\mathbf{q} - I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{col}) \\ &+ \alpha \cdot \min(\|\nabla I_\mathbf{q} - \nabla I_{\mathbf{q}'_{\pi_\mathbf{p}}}\|, \tau_{grad}), \end{aligned} \tag{2}$$

where $\alpha$ balances the contribution of the two terms and $\tau_{col}$ and $\tau_{grad}$ are truncation thresholds to robustify the cost against outliers. In all our experiments we set $\alpha = 0.9$, $\tau_{col} = 10$ and $\tau_{grad} = 2$.

# Gipuma

1) **propagation**: мы хотим массовый параллелизм (многоядерные **CPU/GPU**)
2) **cost**: вклад зависит от похожести на центральный пиксель

# Gipuma

1) **propagation**: мы хотим массовый параллелизм (многоядерные **CPU/GPU**)
2) **cost**: вклад зависит от похожести на центральный пиксель
3) **cost**: сравниваем не только яркости, но и градиенты

# Gipuma

1) **propagation**: мы хотим массовый параллелизм (многоядерные **CPU/GPU**)
2) **cost**: вклад зависит от похожести на центральный пиксель
3) **cost**: сравниваем не только яркости, но и градиенты

## 3.2. Sparse matching cost

We use a similar matching cost as proposed in the original Patchmatch paper [5]. The only difference is that we consider only intensity rather than color differences. The performance improvement when using RGB is tiny and in our view does not justify a threefold increase in runtime. To further speed up the computation we follow the idea of the so-called Sparse Census Transform [41] and use only every other row and column in the window when evaluating the matching cost, resulting in a $4\times$ gain. Empirically, we do not observe any decrease in matching accuracy with this sparse cost.

## 3.2. Sparse matching cost

We use a similar matching cost as proposed in the original Patchmatch paper [5]. The only difference is that we consider only intensity rather than color differences. The performance improvement when using RGB is tiny and in our view does not justify a threefold increase in runtime. To further speed up the computation we follow the idea of the so-called Sparse Census Transform [41] and use only every other row and column in the window when evaluating the matching cost, resulting in a 4× gain. Empirically, we do not observe any decrease in matching accuracy with this sparse cost.

# Gipuma

1) **propagation**: мы хотим массовый параллелизм (многоядерные **CPU/GPU**)
2) **cost**: вклад зависит от похожести на центральный пиксель
3) **cost**: сравниваем не только яркости, но и градиенты
4) **cost:** grayscale картинки вместо RGB (3x ускорение + экономия памяти)

# Gipuma

1) **propagation**: мы хотим массовый параллелизм (многоядерные **CPU/GPU**)
2) **cost**: вклад зависит от похожести на центральный пиксель
3) **cost**: сравниваем не только яркости, но и градиенты
4) **cost:** grayscale картинки вместо RGB (3x ускорение + экономия памяти)
5) **cost:** разреженный Census (ускорение - пропускаем ряды/столбцы патча)

# Gipuma

1) **propagation**: мы хотим массовый параллелизм (многоядерные **CPU/GPU**)
2) **cost**: вклад зависит от похожести на центральный пиксель
3) **cost**: сравниваем не только яркости, но и градиенты
4) **cost:** grayscale картинки вместо RGB (3x ускорение + экономия памяти)
5) **cost:** разреженный Census (ускорение - пропускаем ряды/столбцы патча)

**Cost aggregation** For a specific plane $\pi$, we obtain a cost value $m_i$ from each of the $N$ comparisons. There are different strategies how to fuse these into a single multiview matching cost.

One possible approach is to accumulate over all $n$ cost values, as proposed by Okutomi and Kanade [30]. However, if objects are occluded in some of the views, these views will return a high cost value even for the correct plane, and thereby blur the objective. In order to robustly handle such cases we follow Kang et al. [24]. They propose to include only the best 50% of all $N$ cost values, assuming that at least half of the images should be valid for a given point. We slightly change this and instead of the fixed 50% introduce a parameter $K$, which specifies the number of individual cost values to be considered,

$$m_{srt} = \text{sort}_{\uparrow}(m_1 \dots m_N) \quad , \quad m_{mv} = \sum_{i=1}^{K} m_i \ . \tag{7}$$

The choice of $K$ depends on different factors: in general, a higher value will increase the redundancy and improve the accuracy of the 3D point, but also the risk of including mismatches and thereby compromising the robustness. Empirically, rather low values tend to work better, in our experiments we use $K = 3$ or less for very sparse datasets.

**Cost aggregation** For a specific plane $\pi$, we obtain a cost value $m_i$ from each of the $N$ comparisons. There are different strategies how to fuse these into a single multiview matching cost.

One possible approach is to accumulate over all $n$ cost values, as proposed by Okutomi and Kanade [30]. However, if objects are occluded in some of the views, these views will return a high cost value even for the correct plane, and thereby blur the objective. In order to robustly handle such cases we follow Kang et al. [24]. They propose to include only the best 50% of all $N$ cost values, assuming that at least half of the images should be valid for a given point. We slightly change this and instead of the fixed 50% introduce a parameter $K$, which specifies the number of individual cost values to be considered,

$$m_{srt} = \text{sort}_{\uparrow}(m_1 \dots m_N) \quad , \quad m_{mv} = \sum_{i=1}^{K} m_i \ . \qquad (7)$$

The choice of $K$ depends on different factors: in general, a higher value will increase the redundancy and improve the accuracy of the 3D point, but also the risk of including mismatches and thereby compromising the robustness. Empirically, rather low values tend to work better, in our experiments we use $K = 3$ or less for very sparse datasets.

**Cost aggregation** For a specific plane $\pi$, we obtain a cost value $m_i$ from each of the $N$ comparisons. There are different strategies how to fuse these into a single multiview matching cost.

One possible approach is to accumulate over all $n$ cost values, as proposed by Okutomi and Kanade [30]. However, if objects are occluded in some of the views, these views will return a high cost value even for the correct plane, and thereby blur the objective. In order to robustly handle such cases we follow Kang et al. [24]. They propose to include only the best 50% of all $N$ cost values, assuming that at least half of the images should be valid for a given point. We slightly change this and instead of the fixed 50% introduce a parameter $K$, which specifies the number of individual cost values to be considered,

$$m_{srt} = \text{sort}_{\uparrow}(m_1 \dots m_N) \quad , \quad m_{mv} = \sum_{i=1}^{K} m_i \ . \tag{7}$$

The choice of $K$ depends on different factors: in general, a higher value will increase the redundancy and improve the accuracy of the 3D point, but also the risk of including mismatches and thereby compromising the robustness. Empirically, rather low values tend to work better, in our experiments we use $K = 3$ or less for very sparse datasets.

**Cost aggregation** For a specific plane $\pi$, we obtain a cost value $m_i$ from each of the $N$ comparisons. There are different strategies how to fuse these into a single multiview matching cost.

One possible approach is to accumulate over all $n$ cost values, as proposed by Okutomi and Kanade [30]. However, if objects are occluded in some of the views, these views will return a high cost value even for the correct plane, and thereby blur the objective. In order to robustly handle such cases we follow Kang et al. [24]. They propose to include only the best 50% of all $N$ cost values, assuming that at least half of the images should be valid for a given point. We slightly change this and instead of the fixed 50% introduce a parameter $K$, which specifies the number of individual cost values to be considered,

$$m_{srt} = \text{sort}_{\uparrow}(m_1 \dots m_N) \quad , \quad m_{mv} = \sum_{i=1}^{K} m_i \ . \qquad (7)$$

The choice of $K$ depends on different factors: in general, a higher value will increase the redundancy and improve the accuracy of the 3D point, but also the risk of including mismatches and thereby compromising the robustness. Empirically, rather low values tend to work better, in our experiments we use $K = 3$ or less for very sparse datasets.

**Cost aggregation** For a specific plane $\pi$, we obtain a cost value $m_i$ from each of the $N$ comparisons. There are different strategies how to fuse these into a single multiview matching cost.

One possible approach is to accumulate over all $n$ cost values, as proposed by Okutomi and Kanade [30]. However, if objects are occluded in some of the views, these views will return a high cost value even for the correct plane, and thereby blur the objective. In order to robustly handle such cases we follow Kang et al. [24]. They propose to include only the best 50% of all $N$ cost values, assuming that at least half of the images should be valid for a given point. We slightly change this and instead of the fixed 50% introduce a parameter $K$, which specifies the number of individual cost values to be considered,

$$m_{srt} = \text{sort}_{\uparrow}(m_1 \dots m_N) \quad , \quad m_{mv} = \sum_{i=1}^{K} m_i \ . \qquad (7)$$

The choice of $K$ depends on different factors: in general, a higher value will increase the redundancy and improve the accuracy of the 3D point, but also the risk of including mismatches and thereby compromising the robustness. Empirically, rather low values tend to work better, in our experiments we use $K = 3$ or less for very sparse datasets.

# Gipuma

1) **propagation**: мы хотим массовый параллелизм (многоядерные **CPU/GPU**)
2) **cost**: вклад зависит от похожести на центральный пиксель
3) **cost**: сравниваем не только яркости, но и градиенты
4) **cost:** grayscale картинки вместо RGB (3x ускорение + экономия памяти)
5) **cost:** разреженный Census (ускорение - пропускаем ряды/столбцы патча)
6) **costs aggregation:** усредняем не по всем камерам, а только по **K** лучшим

Colmap

# Pixelwise View Selection for Unstructured Multi-View Stereo

Johannes L. Schönberger[1], Enliang Zheng[2],
Marc Pollefeys[1,3], Jan-Michael Frahm[2]

[1]ETH Zürich, [2]UNC Chapel Hill, [3]Microsoft
{jsch,pomarc}@inf.ethz.ch, {ezheng,jmf}@cs.unc.edu

**Abstract.** This work presents a Multi-View Stereo system for robust and efficient dense modeling from unstructured image collections. Our core contributions are the joint estimation of depth and normal information, pixelwise view selection using photometric and geometric priors, and a multi-view geometric consistency term for the simultaneous refinement and image-based depth and normal fusion. Experiments on benchmarks and large-scale Internet photo collections demonstrate state-of-the-art performance in terms of accuracy, completeness, and efficiency.

**Fig. 1.** Reconstructions for Louvre, Todai-ji, Paris Opera, and Astronomical Clock.

# 3 Review of Joint View Selection and Depth Estimation

This section reviews the framework by Zheng *et al.* [59] to introduce notation and context for our contributions. Since their method processes each row/column independently, we limit the description to a single image row with $l$ as the column index. Their method estimates the depth $\theta_l$ for a pixel in the reference image $X^{\text{ref}}$ from a set of unstructured source images $\boldsymbol{X}^{\text{src}} = \{X^m \mid m = 1 \dots M\}$. The estimate $\theta_l$ maximizes the color similarity between a patch $X_l^{\text{ref}}$ in the reference image and homography-warped patches $X_l^m$ in non-occluded source images. The binary indicator variable $Z_l^m \in \{0, 1\}$ defines the set of non-occluded source images as $\bar{\boldsymbol{X}}_l^{\text{src}} = \{X^m \mid Z_l^m = 1\}$. To sample $\bar{\boldsymbol{X}}_l^{\text{src}}$, they infer the probability that the reference patch $X_l^{\text{ref}}$ at depth $\theta_l$ is visible at the source patch $X_l^m$ using

$$P(X_l^m | Z_l^m, \theta_l) = \begin{cases} \frac{1}{NA} \exp\left(-\frac{(1-\rho_l^m(\theta_l))^2}{2\sigma_\rho^2}\right) & \text{if } Z_l^m = 1 \\ \frac{1}{N}\mathcal{U} & \text{if } Z_l^m = 0, \end{cases} \tag{1}$$

# 3 Review of Joint View Selection and Depth Estimation

This section reviews the framework by Zheng *et al.* [59] to introduce notation and context for our contributions. Since their method processes each row/column independently, we limit the description to a single image row with $l$ as the column index. Their method estimates the depth $\theta_l$ for a pixel in the reference image $X^{\text{ref}}$ from a set of unstructured source images $\boldsymbol{X}^{\text{src}} = \{X^m \mid m = 1 \dots M\}$. The estimate $\theta_l$ maximizes the color similarity between a patch $X_l^{\text{ref}}$ in the reference image and homography-warped patches $X_l^m$ in non-occluded source images. The binary indicator variable $Z_l^m \in \{0, 1\}$ defines the set of non-occluded source images as $\bar{\boldsymbol{X}}_l^{\text{src}} = \{X^m \mid Z_l^m = 1\}$. To sample $\bar{\boldsymbol{X}}_l^{\text{src}}$, they infer the probability that the reference patch $X_l^{\text{ref}}$ at depth $\theta_l$ is visible at the source patch $X_l^m$ using

$$P(X_l^m | Z_l^m, \theta_l) = \begin{cases} \frac{1}{NA} \exp\left(-\frac{(1-\rho_l^m(\theta_l))^2}{2\sigma_\rho^2}\right) & \text{if } Z_l^m = 1 \\ \frac{1}{N}\mathcal{U} & \text{if } Z_l^m = 0, \end{cases} \tag{1}$$

ence. In the case of occlusion, the color distributions of the two patches are unrelated and follow the uniform distribution $\mathcal{U}$ in the range $[-1, 1]$ with probability density 0.5. Otherwise, $\rho_l^m$ describes the color similarity between the reference and source patch based on normalized cross-correlation (NCC) using fronto-parallel homography warping. The variable $\sigma_\rho$ determines a soft threshold for $\rho_l^m$ on the reference patch being visible in the source image. The state-transition matrix from the preceding pixel $l-1$ to the current pixel $l$ is $P(Z_l^m|Z_{l-1}^m) = \begin{pmatrix} \gamma & 1-\gamma \\ 1-\gamma & \gamma \end{pmatrix}$ and encourages spatially smooth occlusion indicators, where a larger $\gamma$ enforces neighboring pixels to have more similar indicators. Given reference and source images $\boldsymbol{X} = \{X^{\text{ref}}, \boldsymbol{X}^{\text{src}}\}$, the inference problem then boils down to recover, for all $L$ pixels in the reference image, the depths $\boldsymbol{\theta} = \{\theta_l \mid l = 1 \dots L\}$ and the occlusion indicators $\boldsymbol{Z} = \{Z_l^m \mid l = 1 \dots L, m = 1 \dots M\}$ from the posterior distribution $P(\boldsymbol{Z}, \boldsymbol{\theta}|\boldsymbol{X})$ with a uniform prior $P(\boldsymbol{\theta})$. To solve the computationally infeasible Bayesian approach of first computing the joint probability

$$P(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\theta}) = \prod_{l=1}^{L} \prod_{m=1}^{M} [P(Z_l^m|Z_{l-1}^m) P(X_l^m|Z_l^m, \theta_l)] \tag{2}$$

and then normalizing over $P(\boldsymbol{X})$, Zheng *et al.* use variational inference the-

ence. In the case of occlusion, the color distributions of the two patches are unrelated and follow the uniform distribution $\mathcal{U}$ in the range $[-1, 1]$ with probability density 0.5. Otherwise, $\rho_l^m$ describes the color similarity between the reference and source patch based on normalized cross-correlation (NCC) using fronto-parallel homography warping. The variable $\sigma_\rho$ determines a soft threshold for $\rho_l^m$ on the reference patch being visible in the source image. The state-transition matrix from the preceding pixel $l-1$ to the current pixel $l$ is $P(Z_l^m|Z_{l-1}^m) = \begin{pmatrix} \gamma & 1-\gamma \\ 1-\gamma & \gamma \end{pmatrix}$ and encourages spatially smooth occlusion indicators, where a larger $\gamma$ enforces neighboring pixels to have more similar indicators. Given reference and source images $\boldsymbol{X} = \{X^{\text{ref}}, \boldsymbol{X}^{\text{src}}\}$, the inference problem then boils down to recover, for all $L$ pixels in the reference image, the depths $\boldsymbol{\theta} = \{\theta_l \mid l = 1 \dots L\}$ and the occlusion indicators $\boldsymbol{Z} = \{Z_l^m \mid l = 1 \dots L, m = 1 \dots M\}$ from the posterior distribution $P(\boldsymbol{Z}, \boldsymbol{\theta}|\boldsymbol{X})$ with a uniform prior $P(\boldsymbol{\theta})$. To solve the computationally infeasible Bayesian approach of first computing the joint probability

$$P(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\theta}) = \prod_{l=1}^{L} \prod_{m=1}^{M} [P(Z_l^m|Z_{l-1}^m) P(X_l^m|Z_l^m, \theta_l)] \tag{2}$$

and then normalizing over $P(\boldsymbol{X})$, Zheng *et al.* use variational inference the-

# Colmap

1) **costs aggregation**: оптимизируем какие камеры считаются inliers

# Colmap

1) **costs aggregation**: оптимизируем какие камеры считаются inliers
2) **costs**: bilateral взвешивание ради лучшего результата на границах

15]), we generate a small set of additional plane hypotheses at each propagation step. We observe that the current best depth and normal parameters can have the following states: neither of them, one of them, or both of them have the optimal solution or are close to it. By combining random and perturbed depths with current best normals and vice versa, we increase the chance of sampling the correct solution. More formally, at each step in PatchMatch, we choose the current best estimate for pixel $l$ according to Eq. (4) from the set of hypotheses

$$\{(\theta_l, \boldsymbol{n}_l), (\theta_{l-1}^{\text{prp}}, \boldsymbol{n}_{l-1}), (\theta_l^{\text{rnd}}, \boldsymbol{n}_l), (\theta_l, \boldsymbol{n}_l^{\text{rnd}}), (\theta_l^{\text{rnd}}, \boldsymbol{n}_l^{\text{rnd}}), (\theta_l^{\text{prt}}, \boldsymbol{n}_l), (\theta_l, \boldsymbol{n}_l^{\text{prt}})\}, \tag{6}$$

where $\theta_l^{\text{rnd}}$ and $\boldsymbol{n}_l^{\text{rnd}}$ denote randomly generated samples. To refine the current parameters when they are close to the optimal solution, we perturb the current estimate as $\theta_l^{\text{prt}} = (1 \pm \epsilon)\theta_l$ and $\boldsymbol{n}_l^{\text{prt}} = \boldsymbol{R}_\epsilon \boldsymbol{n}_l$. The variable $\epsilon$ describes a small depth perturbation, and the rotation matrix $\boldsymbol{R}_\epsilon \in SO(3)$ perturbs the normal direction by a small angle subject to $\boldsymbol{p}_l^T \boldsymbol{n}_l^{\text{prt}} < 0$. Normal estimation improves both the reconstruction completeness and accuracy, while the new sampling scheme leads to both fast convergence and more accurate estimates (Section 5).

15]), we generate a small set of additional plane hypotheses at each propagation step. We observe that the current best depth and normal parameters can have the following states: neither of them, one of them, or both of them have the optimal solution or are close to it. By combining random and perturbed depths with current best normals and vice versa, we increase the chance of sampling the correct solution. More formally, at each step in PatchMatch, we choose the current best estimate for pixel $l$ according to Eq. (4) from the set of hypotheses

$$\{(\theta_l, \boldsymbol{n}_l), (\theta_{l-1}^{\text{prp}}, \boldsymbol{n}_{l-1}), (\theta_l^{\text{rnd}}, \boldsymbol{n}_l), (\theta_l, \boldsymbol{n}_l^{\text{rnd}}), (\theta_l^{\text{rnd}}, \boldsymbol{n}_l^{\text{rnd}}), (\theta_l^{\text{prt}}, \boldsymbol{n}_l), (\theta_l, \boldsymbol{n}_l^{\text{prt}})\}, \tag{6}$$

where $\theta_l^{\text{rnd}}$ and $\boldsymbol{n}_l^{\text{rnd}}$ denote randomly generated samples. To refine the current parameters when they are close to the optimal solution, we perturb the current estimate as $\theta_l^{\text{prt}} = (1 \pm \epsilon)\theta_l$ and $\boldsymbol{n}_l^{\text{prt}} = \boldsymbol{R}_\epsilon \boldsymbol{n}_l$. The variable $\epsilon$ describes a small depth perturbation, and the rotation matrix $\boldsymbol{R}_\epsilon \in SO(3)$ perturbs the normal direction by a small angle subject to $\boldsymbol{p}_l^T \boldsymbol{n}_l^{\text{prt}} < 0$. Normal estimation improves both the reconstruction completeness and accuracy, while the new sampling scheme leads to both fast convergence and more accurate estimates (Section 5).

15]), we generate a small set of additional plane hypotheses at each propagation step. We observe that the current best depth and normal parameters can have the following states: neither of them, one of them, or both of them have the optimal solution or are close to it. By combining random and perturbed depths with current best normals and vice versa, we increase the chance of sampling the correct solution. More formally, at each step in PatchMatch, we choose the current best estimate for pixel $l$ according to Eq. (4) from the set of hypotheses

$$\{(\theta_l, \boldsymbol{n}_l), (\theta_{l-1}^{\text{prp}}, \boldsymbol{n}_{l-1}), (\theta_l^{\text{rnd}}, \boldsymbol{n}_l), (\theta_l, \boldsymbol{n}_l^{\text{rnd}}), (\theta_l^{\text{rnd}}, \boldsymbol{n}_l^{\text{rnd}}), (\theta_l^{\text{prt}}, \boldsymbol{n}_l), (\theta_l, \boldsymbol{n}_l^{\text{prt}})\}, \tag{6}$$

where $\theta_l^{\text{rnd}}$ and $\boldsymbol{n}_l^{\text{rnd}}$ denote randomly generated samples. To refine the current parameters when they are close to the optimal solution, we perturb the current estimate as $\theta_l^{\text{prt}} = (1 \pm \epsilon)\theta_l$ and $\boldsymbol{n}_l^{\text{prt}} = \boldsymbol{R}_\epsilon \boldsymbol{n}_l$. The variable $\epsilon$ describes a small depth perturbation, and the rotation matrix $\boldsymbol{R}_\epsilon \in SO(3)$ perturbs the normal direction by a small angle subject to $\boldsymbol{p}_l^T \boldsymbol{n}_l^{\text{prt}} < 0$. Normal estimation improves both the reconstruction completeness and accuracy, while the new sampling scheme leads to both fast convergence and more accurate estimates (Section 5).[46]

15]), we generate a small set of additional plane hypotheses at each propagation step. We observe that the current best depth and normal parameters can have the following states: neither of them, one of them, or both of them have the optimal solution or are close to it. By combining random and perturbed depths with current best normals and vice versa, we increase the chance of sampling the correct solution. More formally, at each step in PatchMatch, we choose the current best estimate for pixel $l$ according to Eq. (4) from the set of hypotheses

$$\{(\theta_l, \boldsymbol{n}_l), (\theta_{l-1}^{\text{prp}}, \boldsymbol{n}_{l-1}), (\theta_l^{\text{rnd}}, \boldsymbol{n}_l), (\theta_l, \boldsymbol{n}_l^{\text{rnd}}), (\theta_l^{\text{rnd}}, \boldsymbol{n}_l^{\text{rnd}}), (\theta_l^{\text{prt}}, \boldsymbol{n}_l), (\theta_l, \boldsymbol{n}_l^{\text{prt}})\}, \tag{6}$$

where $\theta_l^{\text{rnd}}$ and $\boldsymbol{n}_l^{\text{rnd}}$ denote randomly generated samples. To refine the current parameters when they are close to the optimal solution, we perturb the current estimate as $\theta_l^{\text{prt}} = (1 \pm \epsilon)\theta_l$ and $\boldsymbol{n}_l^{\text{prt}} = \boldsymbol{R}_\epsilon \boldsymbol{n}_l$. The variable $\epsilon$ describes a small depth perturbation, and the rotation matrix $\boldsymbol{R}_\epsilon \in SO(3)$ perturbs the normal direction by a small angle subject to $\boldsymbol{p}_l^T \boldsymbol{n}_l^{\text{prt}} < 0$. Normal estimation improves both the reconstruction completeness and accuracy, while the new sampling scheme leads to both fast convergence and more accurate estimates (Section 5).[47]

# Colmap

1) **costs aggregation**: оптимизируем какие камеры считаются inliers
2) **costs**: bilateral взвешивание ради лучшего результата на границах
3) **propagation**: рассматривать кандидатами парные-мутации глубин и нормалей (из текущей лучшей, из пертурбации, из соседа, из рандома)

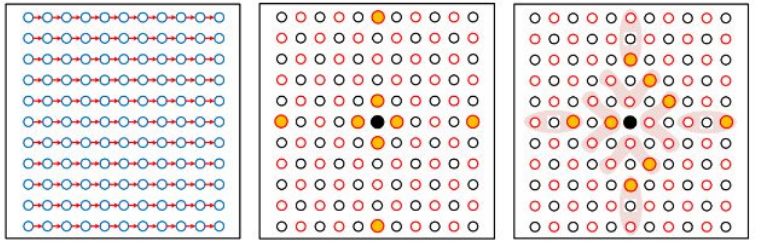
# Multi-Scale Geometric Consistency Guided Multi-View Stereo

Qingshan Xu and Wenbing Tao*
National Key Laboratory of Science and Technology on Multispectral Information Processing
School of Artifical Intelligence and Automation, Huazhong University of Science and Technology, China
{qingshanxu, wenbingtao}@hust.edu.cn



## Abstract

*In this paper, we propose an efficient multi-scale geometric consistency guided multi-view stereo method for accurate and complete depth map estimation. We first present our basic multi-view stereo method with Adaptive Checkerboard sampling and Multi-Hypothesis joint view selection (ACMH). It leverages structured region information to sample better candidate hypotheses for propagation and infer the aggregation view subset at each pixel. For the depth estimation of low-textured areas, we further propose to combine ACMH with multi-scale geometric consistency guidance (ACMM) to obtain the reliable depth estimates for low-textured areas at coarser scales and guarantee that they can be propagated to finer scales. To correct the erroneous estimates propagated from the coarser scales, we present a novel detail restorer. Experiments on extensive datasets show our method achieves state-of-the-art performance, re-*

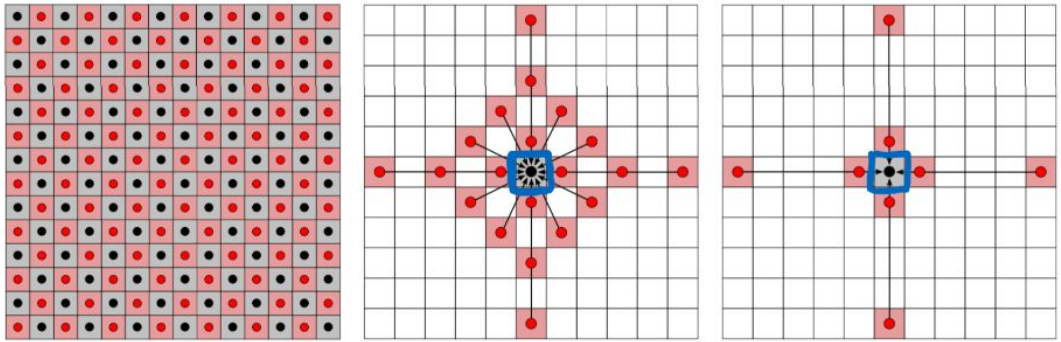


(a) (b) (c)

Figure 1. Propagation scheme. (a) Sequential propagation. (b) Symmetric checkerboard propagation. (c) Adaptive checkerboard propagation. The light red areas in (c) show sampling regions. The solid yellow circles in (b) and (c) show the sampled points.

initialization, propagation, view selection and refinement. In this pipeline, propagation and view selection are two key steps to PatchMatch Stereo methods. The former is important to efficiency while the latter is critical to accuracy.

# ACMH / ACMM



# ACMH / ACMM

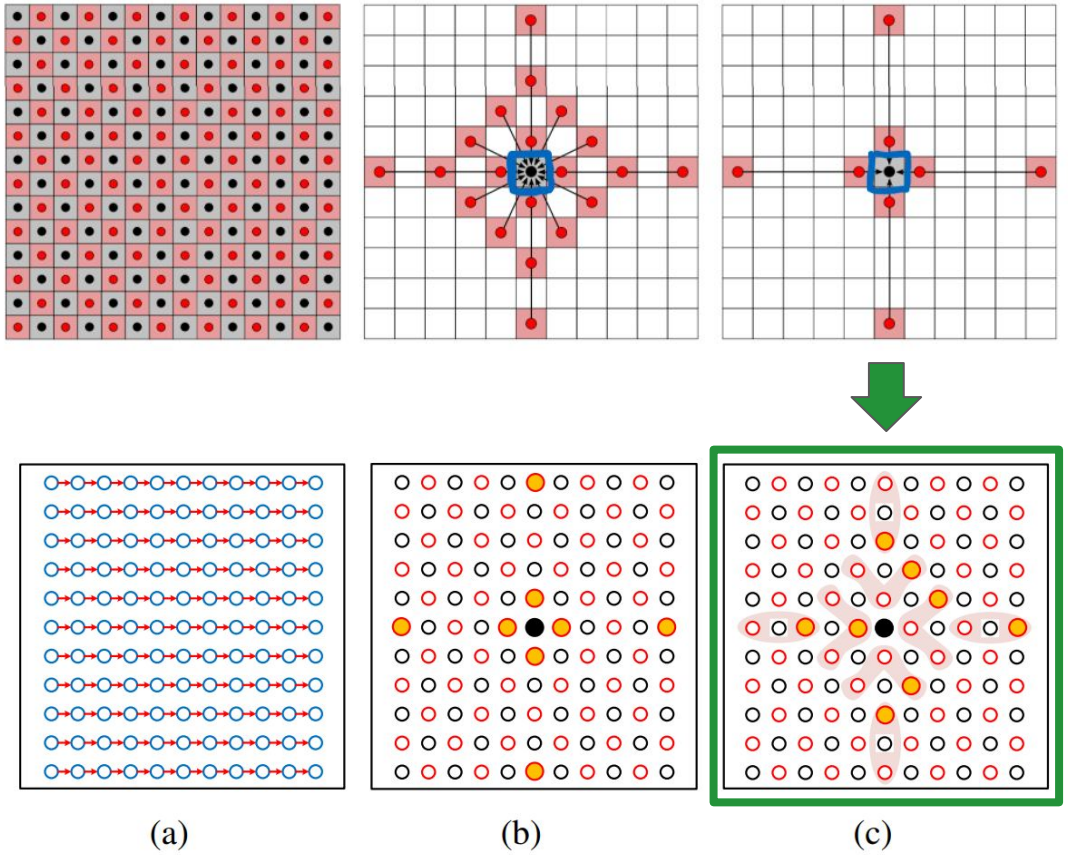



Figure 1. Propagation scheme. (a) Sequential propagation. (b) Symmetric checkerboard propagation. (c) Adaptive checkerboard propagation. The light red areas in (c) show sampling regions. The solid yellow circles in (b) and (c) show the sampled points.

# ACMH / ACMM

1) **propagation**: V-паттерн соседей с локальным выбором победителя

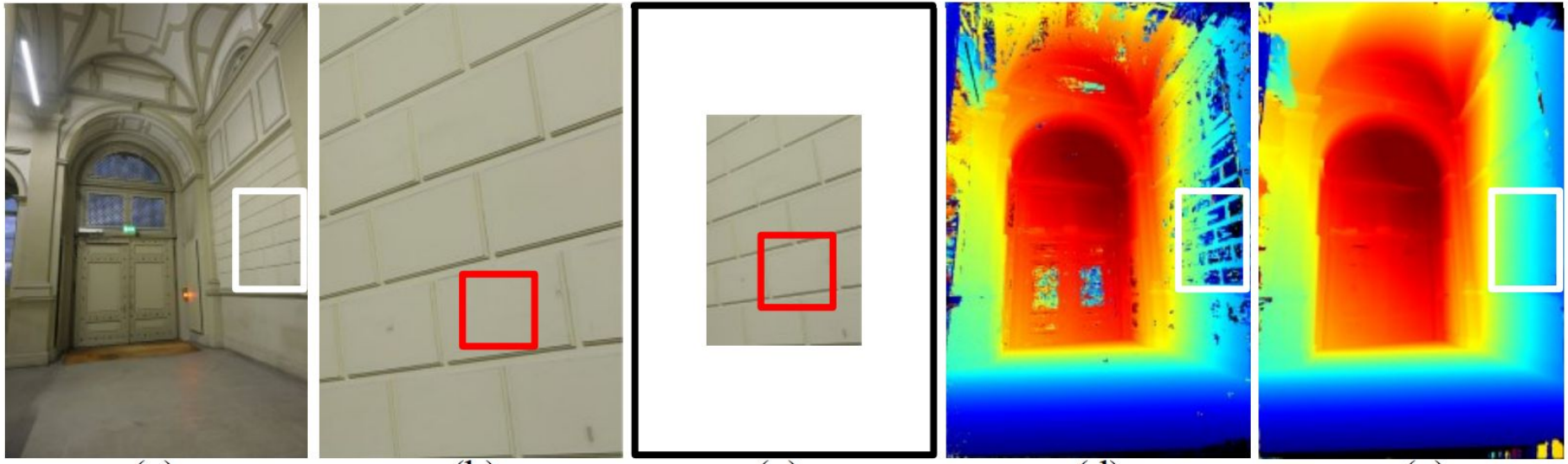


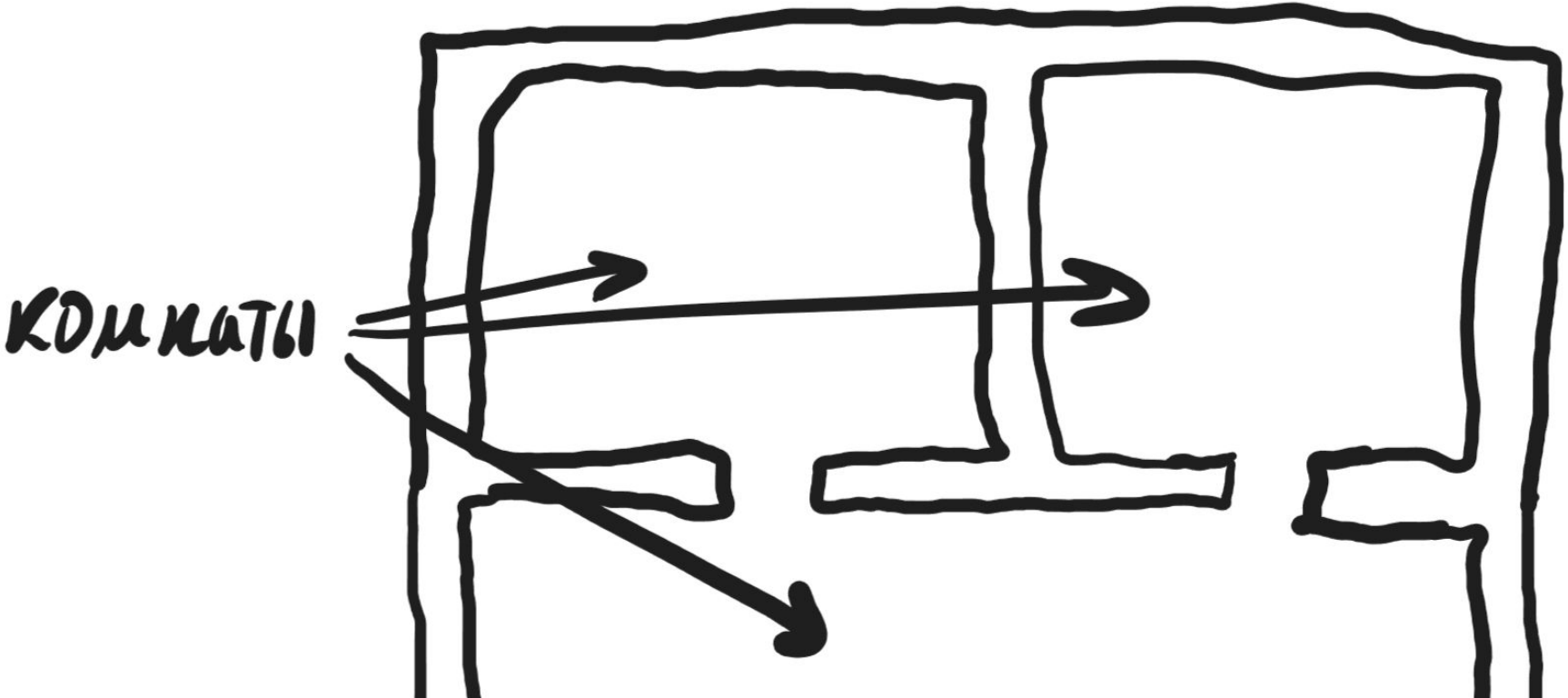

Figure 2. Texture richness for different scales. (a) Original Image. (b) The zoomed-in version of the white box in (a). (c) The down-sampled version of (b). (d) Depth map obtained with the original scale. (e) Depth map obtained with the multi-scale scheme. The patch windows in red are kept the same size.

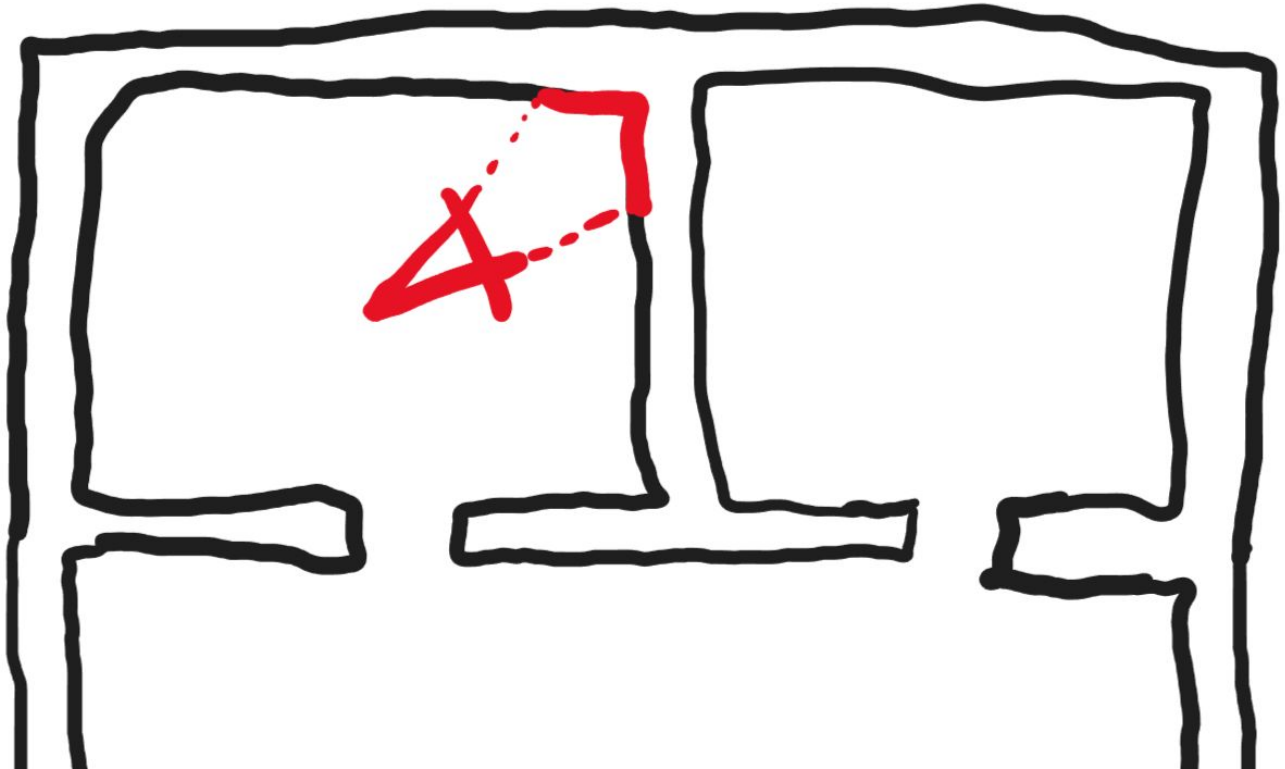
# ACMH / ACMM

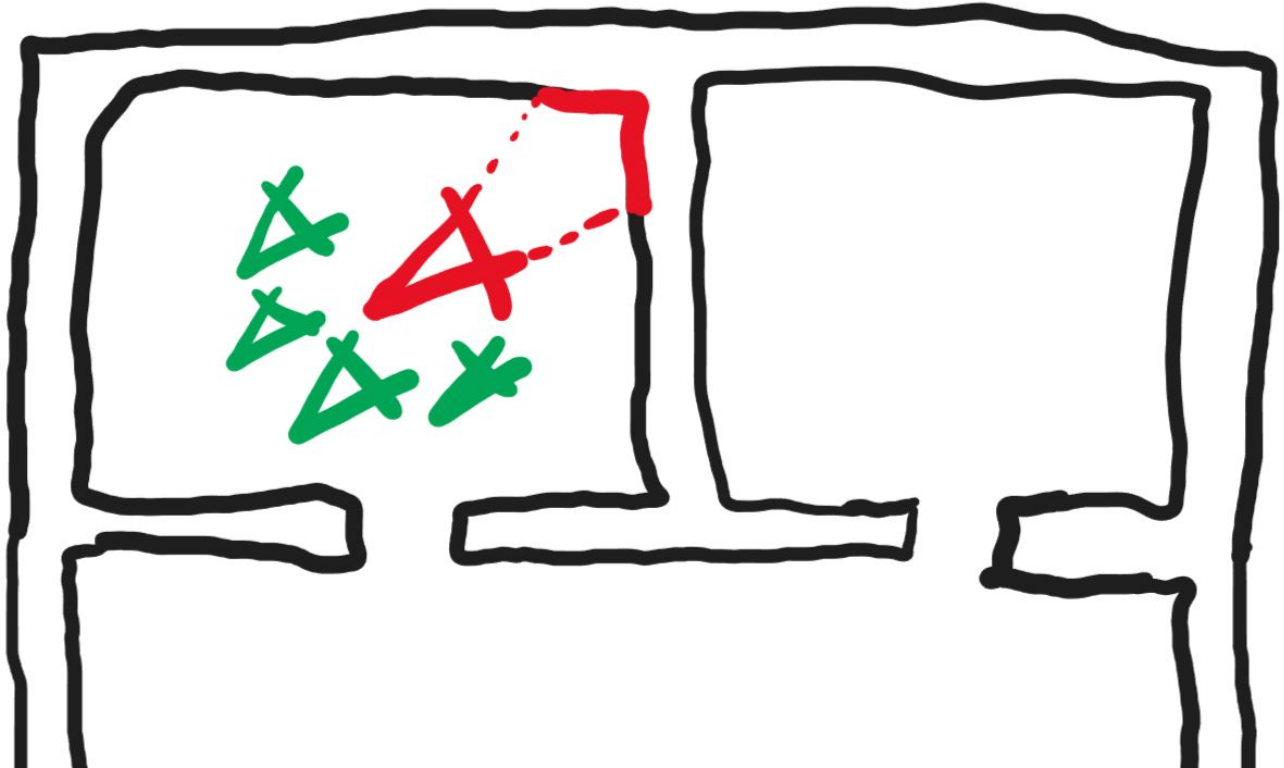
1) **propagation**: V-паттерн соседей с локальным выбором победителя
2) **coarse-to-fine**: слабо-текстурированные поверхности реконструируются на низких уровнях пирамиды, а значит на детальных можно либо к ним тяготеть, либо прямо их глубины/нормали оставить

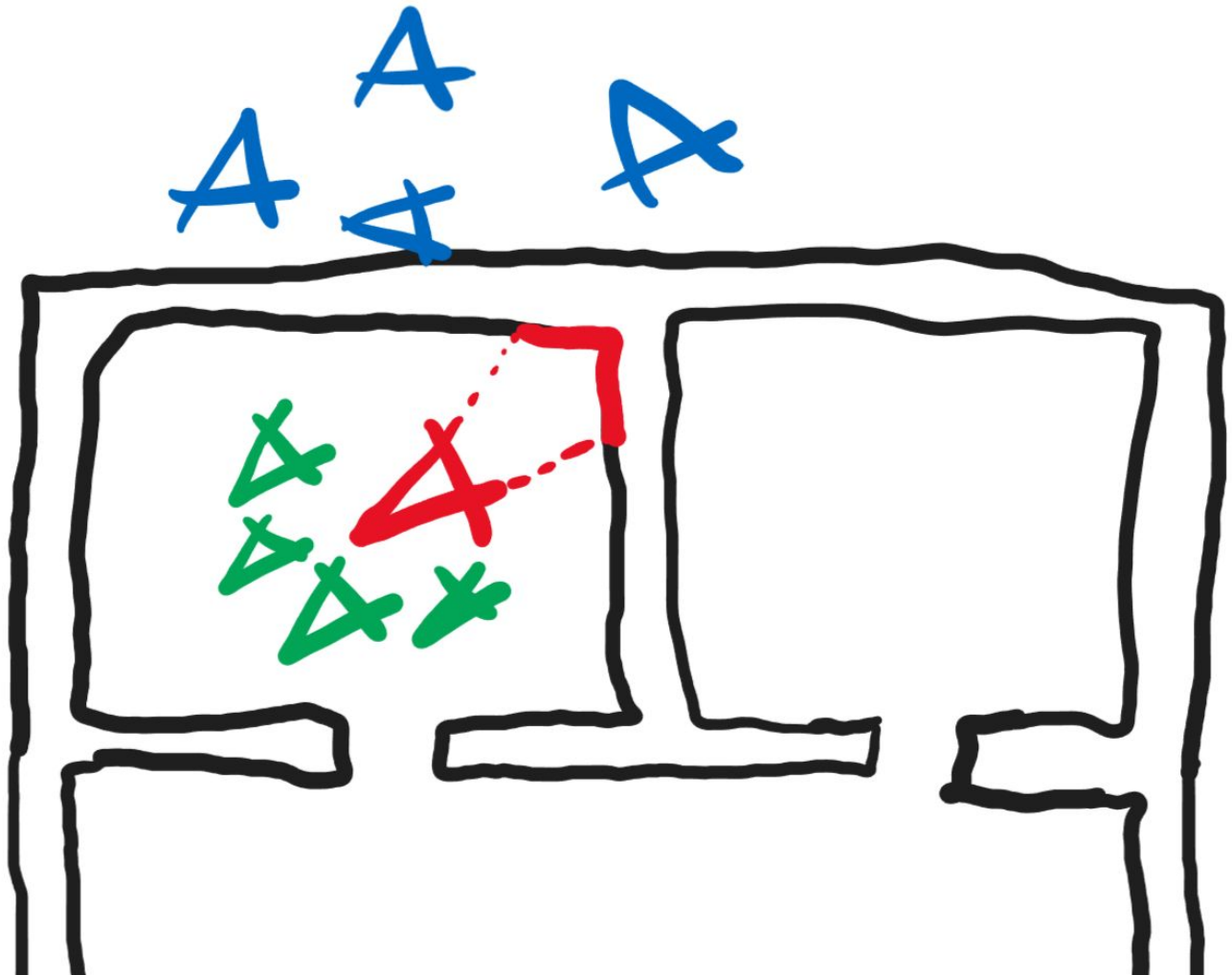
# Мысли

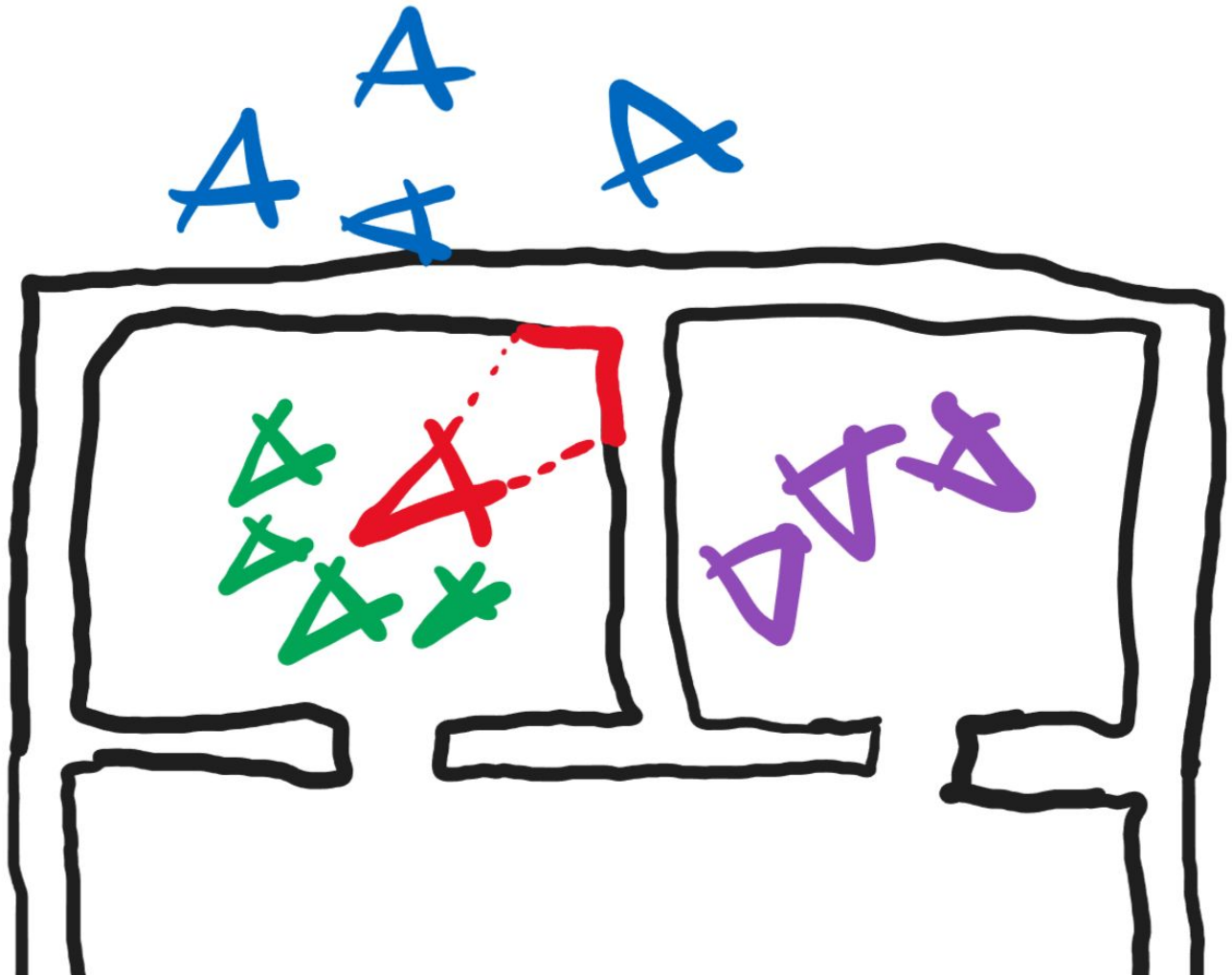
1) Какие соседние фотографии учитывать?

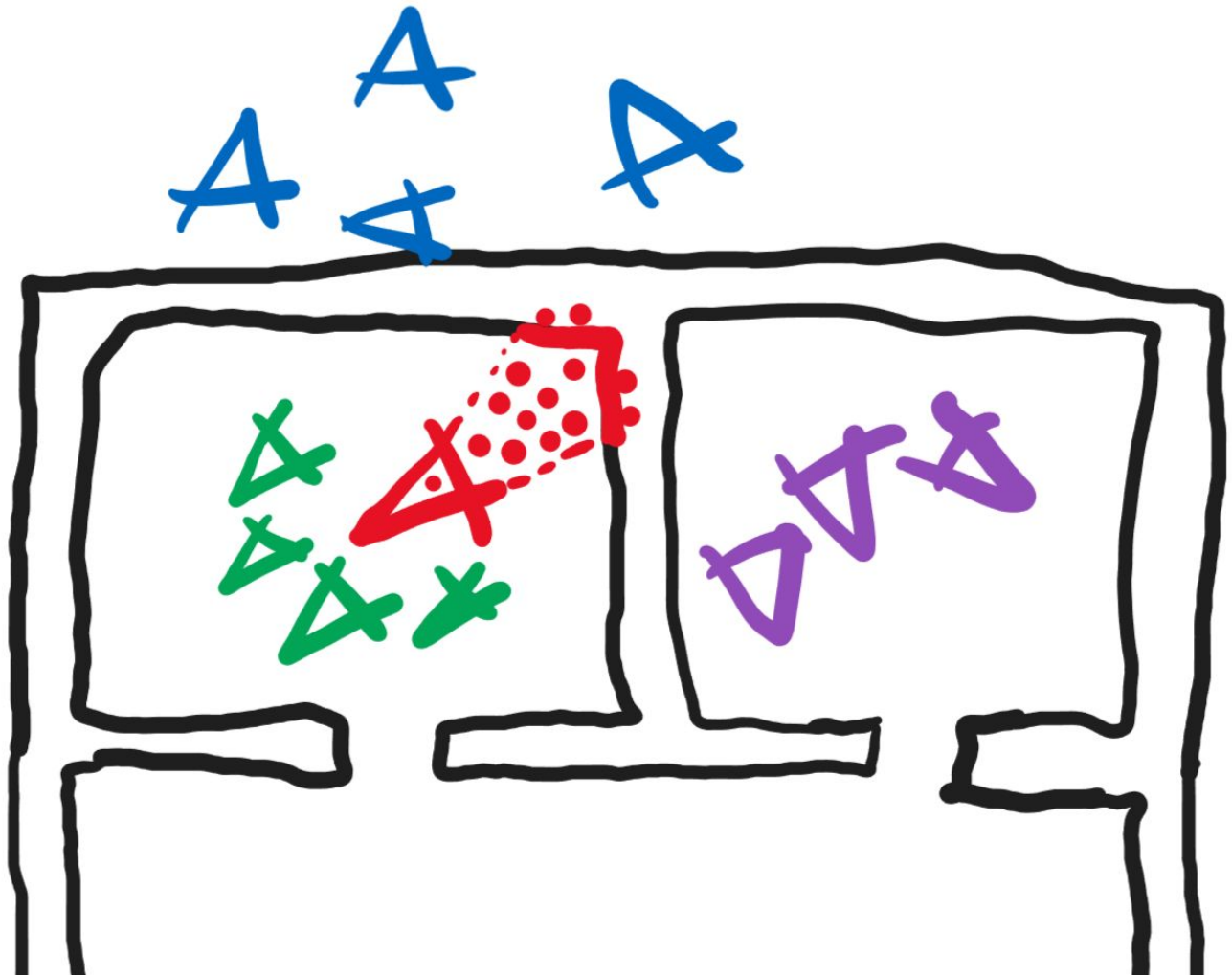

КОМНАТЫ











# Мысли

1) Соседние камеры выбирать на базе ключевых точек

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Как не упасть по памяти если соседних камер много?

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Как пофильтровать выбросы/occlusions?

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.

# Мысли

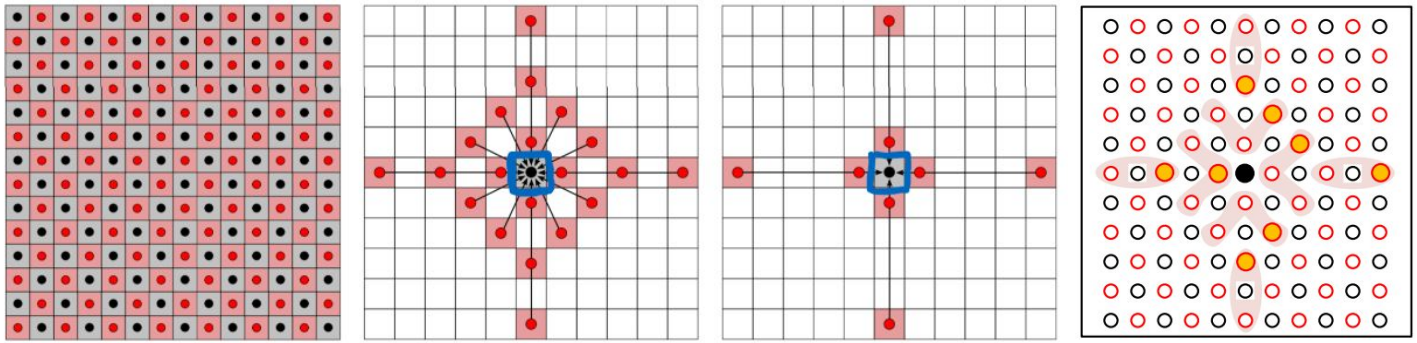
1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.
5) Откуда взять массовый параллелизм для GPU?

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.
5) Массовый параллелизм (для GPU) есть за счет шахматного propagation.



# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.
5) Массовый параллелизм (для GPU) есть за счет шахматного propagation.
6) Как добавить субпиксельную точность глубины?

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.
5) Массовый параллелизм (для GPU) есть за счет шахматного propagation.
6) Субпиксельная точность глубины достигается за счет локальной пертурбации

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.
5) Массовый параллелизм (для GPU) есть за счет шахматного propagation.
6) Субпиксельная точность глубины достигается за счет локальной пертурбации
7) Что если 3x3 окно ведет себя нестабильно а очень хочется 3x3?

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.
5) Массовый параллелизм (для GPU) есть за счет шахматного propagation.
6) Субпиксельная точность глубины достигается за счет локальной пертурбации
7) Можно оценивать cost окном 5x5, а в конце - делать локальные пертурбации оценивая cost 3x3 - для субпиксельного уточнения.

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.
5) Массовый параллелизм (для GPU) есть за счет шахматного propagation.
6) Субпиксельная точность глубины достигается за счет локальной пертурбации
7) Можно оценивать cost окном 5x5, а в конце - делать локальные пертурбации оценивая cost 3x3 - для субпиксельного уточнения.
8) Как более умно выбирать не-occlusion камеры соседи?

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.
5) Массовый параллелизм (для GPU) есть за счет шахматного propagation.
6) Субпиксельная точность глубины достигается за счет локальной пертурбации
7) Можно оценивать cost окном 5x5, а в конце - делать локальные пертурбации оценивая cost 3x3 - для субпиксельного уточнения.
8) Occlusions фильтруем выбором min(K, <1.5*min_cost) лучших по cost.

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.
5) Массовый параллелизм (для GPU) есть за счет шахматного propagation.
6) Субпиксельная точность глубины достигается за счет локальной пертурбации
7) Можно оценивать cost окном 5x5, а в конце - делать локальные пертурбации оценивая cost 3x3 - для субпиксельного уточнения.
8) Occlusions фильтруем выбором min(K, <1.5*min_cost) лучших по cost.
9) Что если мы хотим поощрять большое число камер inlier-ов?

# Мысли

1) Соседние камеры выбирать на базе ключевых точек
2) Берем не больше чем N соседей (лучших по числу общих точек)
3) Occlusions фильтруем выбором K лучших по cost камер-соседей.
4) Выбросы фильтровать будем потом - на этапе объединения карт глубины.
5) Массовый параллелизм (для GPU) есть за счет шахматного propagation.
6) Субпиксельная точность глубины достигается за счет локальной пертурбации
7) Можно оценивать cost окном 5x5, а в конце - делать локальные пертурбации оценивая cost 3x3 - для субпиксельного уточнения.
8) Occlusions фильтруем выбором min(K, <1.5*min_cost) лучших по cost.
9) Уменьшать агрегированный cost из-за большого числа inliers опасно - есть риск что мы будем предпочитать ошибочную гипотезу которая популярна.

# Ссылки

Benchmarks (удобно находить новые хорошие статьи):

- **ETH3D**: High-resolution multi-view benchmark
- **Tanks And Temples** Benchmark

Методы (удобно находить новые хорошие статьи через цитирования уже известных хороших):

- **Gipuma**: Massively Parallel Multiview Stereopsis by Surface Normal Diffusion, Galliani et. al., 2015
- **Colmap**: Pixelwise View Selection for Unstructured Multi-View Stereo, Schonberger et. al., 2016
- **ACMH/ACMM**: Multi-Scale Geometric Consistency Guided Multi-View Stereo, Xu Q, Tao W, 2019

Исходные коды (**осторожно с лицензиями!**):

- **ACMH/ACMM**: оригинальные ACMH (MIT), ACMM (MIT),
  сторонняя реализация AMCH-ACMM (MIT)
  сторонняя реализация в OpenMVS / ACPMH (**AGPL**)
- **Colmap:** Colmap (BSD)
- **Gipuma:** Gipuma (**GPL**)
- **OpenMVS:** OpenMVS / PatchMatch (**AGPL**)

Разное:

- PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing, Barnes et. al., 2009
- Записки с обзором методов и идеями

# Вопросы?

Полярный Николай
polarnick239@gmail.com